Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Investigation of 6-DOF Robot Path Planning Using Evolutionary Algorithms

Mohamed Gamal Ebrahem AbdelJawad

mohamed.abdelgwad@feng.bu.edu.eg

Benha University

Abdel-Kader Abdel-Karim Ibrahim

Benha University

Tarek Mohamed Tawfeek

Benha University

Research Article

Keywords: Multi-objective optimization, Trajectory optimization, 6-DOF manipulator, Evolutionary algorithms, MOPSO algorithm, NSGA-II algorithm

Posted Date: April 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8693518/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

An Investigation of 6-DOF Robot Path Planning Using Evolutionary Algorithms

M.G.E.AbdelJawad[1], A.A.Ibrahim[1,2] and T.M.Tawfeek[1,3]

Abstract

Multi-objective optimization (MOO) plays a vital role in robotics, where several conflicting objectives—such as minimizing energy consumption, execution time, and motion jerk—must be simultaneously satisfied. Traditional single-objective approaches fail to represent the trade-offs among such competing criteria, motivating the adoption of population-based evolutionary and swarm intelligence algorithms. This study presents a novel comparative investigation of two leading multi-objective optimization methods, MOPSO and NSGA-II, for 6-DOF robotic trajectory planning. The study also lays the foundation for a hybrid framework that sequentially combines both algorithms to leverage their complementary strengths. Both algorithms are applied to the multi-objective trajectory optimization of the AR4 robotic manipulator, aiming to achieve optimal motion performance under kinematic and dynamic constraints. The study evaluates each algorithm's convergence behavior, diversity preservation, and computational efficiency using standard performance metrics—Generational Distance (GD), Spread (SP), and Hypervolume (HV). Results demonstrate that MOPSO exhibits rapid convergence and superior exploration capabilities, while NSGA-II achieves finer convergence accuracy and better Pareto front uniformity. Building on this complementarity, the paper proposes a future hybrid framework combining MOPSO's exploratory strength with NSGA-II's refinement and stability. The envisioned MOPSO→NSGA-II hybrid approach seeks to balance exploration and exploitation dynamically, achieving dense, smooth, and well-converged Pareto fronts. The findings provide a foundation for developing robust hybrid metaheuristics in robotic trajectory optimization, enhancing performance and adaptability in complex multi-objective environments.

Keywords Multi-objective optimization · Trajectory optimization · 6-DOF manipulator · Evolutionary algorithms · MOPSO algorithm · NSGA-II algorithm.

1 Introduction

Trajectory planning of robotic manipulators plays a critical role in achieving smooth, efficient, and reliable motion between predefined waypoints. In industrial and service robotics, trajectory quality directly affects execution time, energy consumption, mechanical stress, and motion smoothness. As robotic systems are increasingly deployed in complex and high-performance environments, trajectory planning has evolved from simple motion generation toward optimization-based approaches capable of handling multiple, often conflicting, performance objectives (Francisco et al., 2021).

Manipulator trajectory planning can generally be performed in either Cartesian space or joint space. Although Cartesian space planning provides intuitive path representation, it often involves complex kinematic computations and may lead to singularities. Consequently, joint space trajectory planning is commonly preferred for robotic manipulators due to its computational simplicity and stability. Among the various curve interpolation methods, B-spline curves have gained widespread application in joint space trajectory planning because of their strong local controllability and ability to generate smooth, continuous profiles with adjustable flexibility (Hu et al., 2024).

✉ **T.M.Tawfeek**

TAREK.BAKHIT@feng.bu.edu.eg

✉ **M.G.E.AbdelJawad** (corresponding author)

mohamed.abdelgwad@feng.bu.edu.eg

✉ **A.A.Ibrahim**

ABDELKADER.IBRAHIM@feng.bu.edu.eg

[1] Mech. Eng. Dept, Shoubra Faculty of Engineering, Banha University, Cairo, Egypt

[2] Higher Technological Institute 10th of Ramadan, Egypt.

[3] Mech. Eng. Dept., Faculty of Engineering, Sinai University, Egypt.

Multi-objective trajectory optimization aims to simultaneously minimize competing criteria such as execution time, energy consumption, and motion jerk. Improving one objective often degrades another, making single-objective formulations insufficient for practical robotic applications. Consequently, population-based multi-objective optimization algorithms, which generate a set of Pareto-optimal solutions instead of a single solution, have become widely adopted in robotic trajectory planning (Cui et al., 2019; Francisco et al., 2021).

Among these methods, evolutionary and swarm intelligence algorithms—particularly the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Multi-Objective Particle Swarm Optimization (MOPSO)—have been extensively applied to manipulator trajectory optimization.

Numerous studies have demonstrated their effectiveness in optimizing time, energy, and smoothness using spline-based joint-space trajectories. Other works have proposed improved or hybrid variants of these algorithms to enhance convergence speed or solution diversity. However, such studies often focus on algorithm modification rather than providing a clear understanding of the intrinsic performance characteristics of the original algorithms.

Despite their widespread use, a systematic and fair comparison between standard NSGA-II and MOPSO under identical experimental conditions remains limited, particularly for 6-DOF robotic manipulators. In many cases, differences in problem formulation, parameter settings, or evaluation metrics make it difficult to objectively assess their relative strengths and weaknesses in practical trajectory optimization tasks.
Motivated by this gap, this paper presents a comprehensive comparative study of NSGA-II and MOPSO for multi-objective trajectory optimization of a 6-DOF AR4 robotic manipulator. Quintic B-spline trajectories are employed in joint space to ensure continuous velocity and acceleration profiles. The algorithms are evaluated using standard performance metrics : Generational Distance (GD), Spacing (SP), and Hypervolume (HV) to analyze convergence behavior, diversity preservation, and Pareto front quality.

Several researchers have investigated the use of optimization algorithms in manipulator trajectory planning. (Ramabalan et al., 2009; Saravanan et al., 2009) utilized third-order uniform B-spline curves combined with the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) and Multi-Objective Differential Evolution (MODE) to optimize time, energy, and obstacle avoidance. However, third-order B-splines cannot ensure acceleration continuity, which may result in unwanted vibrations during motion.

To address this limitation, higher-order spline curves such as quintic B-splines have been adopted in later studies to achieve continuous acceleration and smoother transitions.

These studies often employed metaheuristic algorithms such as NSGA-II and Multi-Objective Particle Swarm Optimization (MOPSO) to optimize one or more performance indices including execution time, energy, and motion smoothness (Huang et al., 2018; Lan et al., 2020; Shi et al., 2016, 2018).

Different multi-objective optimization algorithms exhibit distinct convergence and diversity behaviors, which may significantly affect their suitability for practical robotic trajectory planning problems (Guo & Lai, 2020).

Several studies have proposed improved or modified variants of NSGA-II and MOPSO to enhance specific aspects such as convergence speed or diversity. However, these works primarily focus on algorithmic modification rather than comparative evaluation of the baseline algorithms.
The main contributions of this work are as follows:

- the comparative analysis of two widely used multi-objective optimization algorithms NSGA-II and MOPSO for the trajectory planning of a 6-DOF AR4 manipulator.
- Quintic B-spline interpolation is employed in joint space to ensure smooth motion profiles with continuous acceleration and velocity.
- The optimization objectives include minimizing total execution time, reducing energy consumption, and minimizing motion pulsation.
- Each algorithm is implemented and evaluated using MATLAB to assess its convergence behavior, diversity maintenance, and Pareto front quality.
- Quantitative analysis of convergence, diversity, and stability using benchmark test functions and a real AR4 robot case study, leading to the identification of complementary strengths of NSGA-II and MOPSO and motivating a sequential hybrid framework as future work.

2 Trajectory planning

2.1 B-spline

The expression of B-spline is as follows.

$$P(u) = \sum_{i=0}^{n} Q_i \, N_{i,k}(u) \tag{1}$$

where k donates the degree of the B-spline, n + 1 is the number of control points, Q_i (i = 0, 1, …, m) is the ith control points of the B-spline, $N_{i,k}$ (u) (i = 0, 1, …, m) is the i_{th} basis function of the k_{th}order B-spline, u is the normalized knot vector variable and p(u) is the joint position at the time u. $N_{i,k}$ (u) can be derived by means of the Cox-de Boor recursion formula (De Boor, 1978; Hu et al., 2024; K. F. Wang et al., 2023).

$$\begin{cases} N_{i,0}(u) = \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & \text{elsewhere} \end{cases} \\ N_{i,k}(u) = \dfrac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \dfrac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \end{cases} \tag{2}$$

To derive Eq. (1), we can obtain the r_{th}order derivative of the B-spline.

$$P(u) = \sum_{i=0}^{n} Q_i N_{i,k}(u) \tag{3}$$

$$Q_j^r = \begin{cases} Q_j, s = 0 \\ (k+1-s)\frac{Q_j^{x-1} - Q_{j-1}^{x-1}}{u_{j+k+1-x} - u_j}, \quad s = 1,2,\dots,r \end{cases} \tag{4}$$

It can be used to calculate velocity, acceleration, and jerk.

For the reverse calculation of control points, let's assume that the B -spline curve needs to pass through a set of path points, denoted as P_i(i = 0,1, … , n). We can obtain $n + 1$ equations that satisfy the constraint conditions. However, we require $n + k$ control points, so we need to add $k - 1$ additional equations. In the case of using a quintic B-spline curve ($k = 5$), we need to add four additional constraint conditions. Let's denote the initial point's velocity and acceleration as v_0 and a_0, respectively, and the final point's velocity and acceleration as v_f and a_f, respectively. This leads to a total of $n + 5$ equations.

$$P(u_{i+5}) = \sum_{i=i}^{i+4} Q_j N_{j,5}(u_{i+5}) \triangleq P_i, u_{i+5} = [u_5, u_{n+5}]\ i = 0,1,\dots,n \tag{5}$$

$$\begin{cases} v_0 = P'(u_n) & v_f = P'(u_{n+5}) \\ a_0 = P''(u_5) & a_f = P''(u_{n+5}) \end{cases} \tag{6}$$

According to Eqs. (5) and (6), we can derive a matrix equation.

$$A_N Q = P \tag{7}$$

where,

$$Q = [Q_0, Q_1, \dots, Q_n, Q_{n+1}, Q_{n+2}, Q_{n+3}, Q_{n+4}]^{\mathrm{T}};$$

$$P\left[P_0, P_1, \dots, P_n, v_0, v_f\ a_0, a_f\right]^{\mathrm{T}}$$

Therefore, we can use Eq. (7) to solve the control points.

$$Q = A_N^{-1} P \tag{8}$$

2.2 Objective Function Construction

We define the multi-objective function for the manipulator as follows:

$$S_1 = min\left(\sum_{i=1}^{n-1} \Delta t_i\right) = min\left[\sum_{i=0}^{n-1} (t_{i+1} - t_i)\right] \tag{9}$$

$$S_2 = min\left[\sum_{j=0}^{N} \sqrt{\frac{\int_0^T \left(a_j(t)\right)^2 \mathrm{d}t}{T}}\right] \tag{10}$$

$$S_3 = min\left[\sum_{j=0}^{N} \sqrt{\frac{\int_0^T \left(j_j(t)\right)^2 \mathrm{d}t}{T}}\right] \tag{11}$$

In the equation, S_1, S_2, and S_3 represent the operational time, energy consumption, and pulsation impact of the manipulator, respectively. n is the number of path points, i is the index of the path point, N is the number of joints, j is the index of the joint, $a(t)$ represents the joint acceleration, and $j(t)$ represents the joint jerk.

The kinematic constraints of the manipulator are as follows:

$$\begin{cases} \left|v_j(t)\right| = \left|p'(t)\right| \le V_{j\max}, & j = 1,2,\dots,N. \\ \left|a_j(t)\right| = \left|p''(t)\right| \le A_{j\max}, & j = 1,2,\dots,N \\ \left|j_j(t)\right| = \left|p'''(t)\right| \le J_{j\max}, & j = 1,2,\dots,N. \end{cases} \tag{12}$$

where, $V_{j\max}$, $A_{j\max}$, and $J_{j\max}$ represent the maximum velocity, acceleration, and jerk of j th joint, respectively.

3 MOO Algorithms

3.1 MOPSO Algorithm

The Multi-Objective Particle Swarm Optimization (MOPSO) algorithm is a swarm intelligence-based metaheuristic inspired by the coordinated motion of flocks of birds or schools of fish. It extends the classical Particle Swarm Optimization (PSO) to address problems involving multiple conflicting objectives by producing a **set of Pareto-optimal solutions** instead of a single global optimum. Through cooperative interactions among particles, MOPSO efficiently explores the search space and identifies trade-offs among competing objectives (Kennedy & Eberhart, 1995; Reyes-Sierra & Coello, 2006).

Each particle represents a potential solution characterized by its position vector (decision variables) and velocity vector (rate of change). During the optimization process, particles navigate the search space by updating their velocities and positions based on both personal experience and collective swarm intelligence.

The velocity and position update equations are expressed as:

$$v_i = wv_i + c_1 r_1 (p_{\text{best}} - x_i) + c_2 r_2 (g_{\text{best}} - x_i) \quad (13)$$

$$x_i = x_i + v_i \quad (14)$$

where:w is the inertia weight controlling the influence of previous velocities,c_1 and c_2 are the cognitive and social learning coefficients,r_1, r_2 are random numbers uniformly distributed in $[0,1]$,p_{best} denotes the particle's personal best position, and g_{best} represents the global (leader) best selected from an external repository.

This formulation maintains a balance between **exploration** (global search through inertia and randomness) and **exploitation** (local refinement through attraction to best-known positions), enabling the swarm to converge efficiently toward promising regions in the objective space.

MOPSO typically employs an **external archive** to store non-dominated solutions and a **leader selection strategy** (e.g., based on crowding distance or grid-based methods) to preserve diversity along the Pareto front. These mechanisms ensure the algorithm's capability to produce well-distributed and high-quality trade-off solutions.(Coello et al., 2004; Reyes-Sierra & Coello, 2006).

In practical applications, MOPSO is employed to search for the best compromise among several performance criteria. The optimization process begins with defining the **decision variables** (such as time intervals, control parameters, or design dimensions) and formulating the **objective functions** to be minimized or maximized (for example, minimizing energy, time, or jerk).

Next, **constraints** are imposed to ensure solution feasibility—such as physical limits, joint boundaries, or maximum allowable velocities and accelerations in robotic systems. The swarm is then initialized with random particles distributed across the feasible search space, and the algorithm iteratively updates each particle's velocity and position while maintaining an archive of the best non-dominated solutions.

The process continues until a predefined **termination condition** (typically the maximum number of iterations) is reached. The final content of the archive represents the **approximate Pareto front**, providing a diverse set of optimal trade-off solutions among the objectives.

In summary, MOPSO serves as a powerful optimization framework capable of balancing exploration and convergence, making it suitable for complex, multi-objective engineering problems where multiple performance measures must be optimized simultaneously.

3.2 MOPSO Flowchart

The Multi-Objective Particle Swarm Optimization (MOPSO) algorithm extends the classical PSO framework to handle multiple conflicting objectives by generating a set of Pareto-optimal solutions instead of a single global optimum.

MOPSO integrates **swarm intelligence** with **Pareto-based selection** and **diversity preservation** mechanisms to achieve a well-distributed set of optimal solutions. The overall procedure is illustrated in **Fig. 1,** and its steps are summarized below:

Algorithm Steps:

1. **Initialize Swarm:** A swarm of particles is randomly generated with initial positions and velocities within the defined search domain. Each particle represents a candidate solution vector. Random initialization ensures coverage of diverse regions in the design space, enhancing global exploration capability.
2. **Update External Repository:** All non-dominated solutions are stored in an external repository representing the evolving Pareto front. Dominated particles are removed to maintain elitism. This repository ensures preservation of the best solutions across iterations.
3. **Select Leader (Guide):** A leader (guide) is selected from the repository using the **crowding distance** measure. Leaders from less crowded regions are preferred, promoting a uniform distribution of solutions along the Pareto front.
4. **Update Velocity and Position:** Each particle updates its velocity based on three terms: its previous velocity, its personal best position, and the selected leader's position. The updated position balances exploration and exploitation to achieve convergence toward the Pareto front.
5. **Apply Mutation (Optional):** Random perturbations (mutations) can be applied to selected particles to enhance diversity and prevent premature convergence. This step ensures exploration of unexplored regions in the objective space.
6. **Update and Prune Repository:** The repository is updated by adding new non-dominated particles. If it exceeds the maximum allowed size, a pruning mechanism based on crowding distance is applied to remove densely packed solutions while keeping the best spread of Pareto-optimal points.
7. **Termination and Output:** The iterative process continues until the termination condition (typically the

maximum number of iterations) is reached. The final repository then represents the estimated Pareto front — the set of optimal trade-off solutions for the given problem.

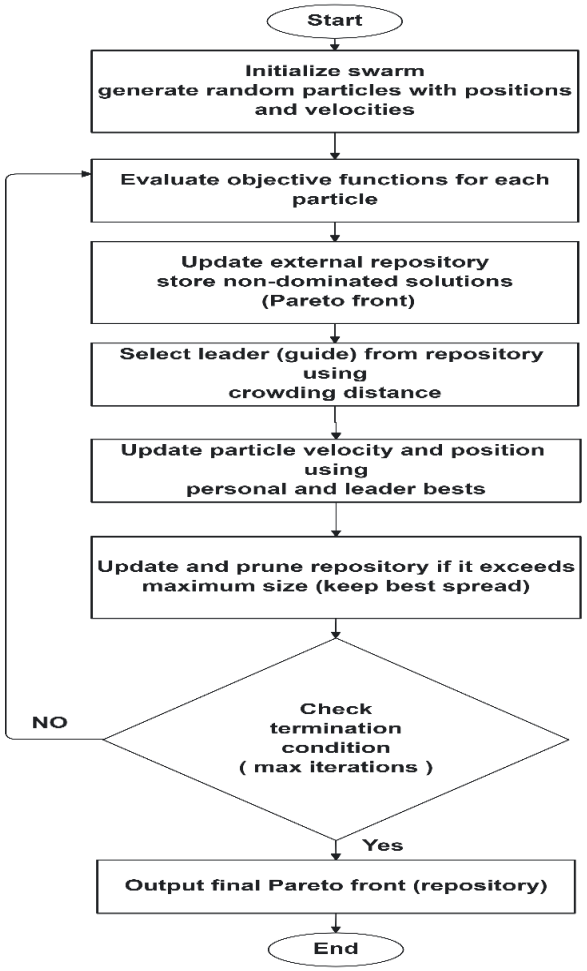


Fig. 1Flow chart of the MOPSO algorithm**.**

3.3. NSGA-II Algorithm

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a widely used evolutionary algorithm for solving multi-objective optimization problems. Unlike single-objective methods, NSGA-II generates a set of Pareto-optimal solutions that represent trade-offs among conflicting objectives, allowing designers to explore different optimal alternatives.(Deb, 2001; Deb et al., 2002).

NSGA-II operates based on genetic algorithm principles, simulating natural evolution through selection, crossover, and mutation. Its key features include:

- **Fast Non-Dominated Sorting:** Individuals are ranked into Pareto fronts based on dominance relationships, ensuring that non-dominated solutions are identified and prioritized.
- **Crowding Distance:** Diversity is maintained by evaluating how isolated each individual is in the objective space, promoting a well-spread Pareto front.
- **Elitism:** Best solutions from both the parent and offspring populations are retained across generations, enhancing convergence and solution quality.

These mechanisms allow NSGA-II to efficiently balance convergence toward optimal solutions while preserving diversity, making it suitable for complex multi-objective engineering problems.

3.4. NSGA-II Flowchart

The NSGA-II implementation follows a population-based evolutionary approach, integrating non-dominated sorting and diversity preservation to generate high-quality Pareto-optimal solutions. The overall procedure is illustrated in **Fig. 2,** and its steps are summarized below:

1. **Initialize Population:** A population of candidate solutions is randomly generated within the defined search space. This ensures broad coverage and exploration of potential solutions.

2. **Evaluate Objective Functions:** Each individual in the population is evaluated against the problem's objectives. The results are used to determine Pareto dominance among solutions.

3. **Non-Dominated Sorting and Crowding Distance Calculation:** Individuals are classified into Pareto fronts based on non-dominated sorting. Within each front, crowding distances are calculated to measure density and maintain diversity along the Pareto front.

4. **Selection, Crossover, and Mutation:** Parent solutions are selected using a tournament or rank-based strategy. Crossover combines parents to generate offspring, while mutation introduces small variations, allowing exploration of the search space.

5. **Combine and Sort Populations:** Parent and offspring populations are merged and re-sorted using non-dominated sorting and crowding distance. Only the best solutions are retained to maintain the original population size.

6. **Iterate Until Termination:** The evolutionary cycle repeats until the stopping criteria are met (e.g., maximum generations). The final population represents the Pareto front, providing a set of trade-off solutions among the objectives.

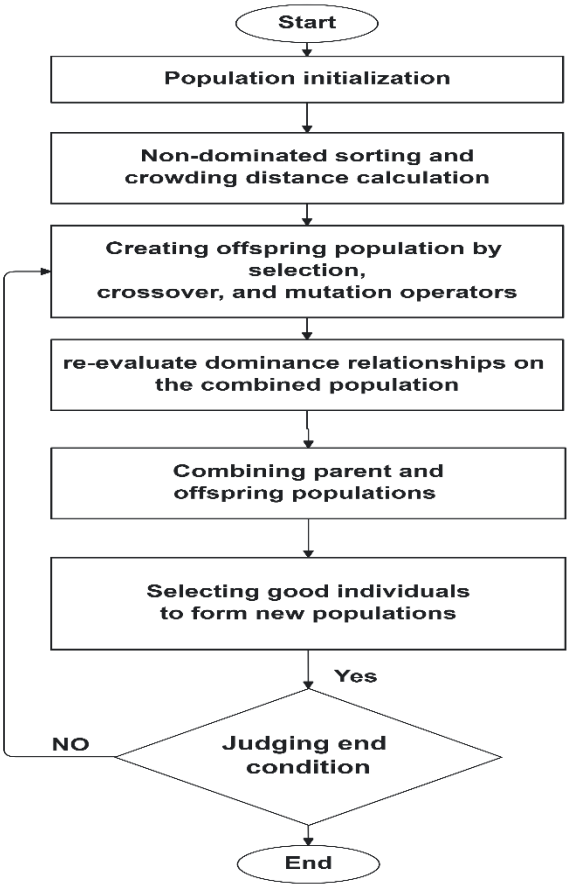


Fig. 2 Flow chart of the NSGA-II algorithm.

4. Algorithm Comparison Verification

To evaluate and validate the performance of the proposed multi-objective optimization algorithm on the ZDT1 benchmark, three standard performance indicators were employed: Hypervolume (HV), Spacing Metric (SP), and Generational Distance (GD). HV measures the proportion of objective space dominated by the obtained solutions, providing an overall assessment of convergence and diversity. SP evaluates the uniformity of distribution along the Pareto front, indicating how evenly the solutions are spread. GD quantifies the average distance between the approximate and true Pareto fronts, reflecting the accuracy of convergence. Together, these three metrics give a comprehensive assessment of both convergence quality and diversity of the obtained ZDT1 solutions.

4.1. Hypervolume (HV)

To evaluate both convergence and diversity of the obtained Pareto front, the Hypervolume (HV) metric is used HV measures the dominated area between the set of nondominated solutions and a reference point. A larger HV value indicates that the solutions cover a wider and better portion of the objective space. The HV calculation formula is expressed as:

$$HV = \sum_{i=1}^{N} \left(r_1 - f_1^{(i)}\right)\left(r_2 - f_2^{(i)}\right) \qquad (15)$$

where:

- N represents the number of nondominated solutions.
- $f_1^{(i)}, f_2^{(i)}$ denote the abjective values of the i-th Pareto solution sorted in ascending order of the first objective.
- r_1, r_2 are the coordinates of the reference point that dominates all solutions.
- A larger HV value reflects better convergence and wider coverage of the Pareto front.

4.2. Spacing Metric (SP)

To assess the uniformity of the Pareto front, the Spacing (SP) metric is employed. A smaller SP value indicates a more evenly distributed solution set, which corresponds to better diversity in multi-objective optimization. The SP calculation formula is defined as:

$$SP = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (d_i - d)^2} \qquad (16)$$

where:

- N represents the number of solutions in the Pareto front; therefore, there exist $N-1$ consecutive distances.
- d_i denotes the nearest-neighbor Euclidean distance of the i-th solution to any other solution.
- $\bar{d}$ is the arithmetic mean of all distances d_i.

– A lower SP value indicates a more uniform distribution across the Pareto front.

4.3. Generational Distance (GD)

To quantify the convergence accuracy of the obtained solutions, the Generational Distance (GD) metric is applied. GD measures how close the algorithm's approximated solutions are to the true Pareto front. A smaller GD value corresponds to better convergence performance. The GD formula is given by:

$$GD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\min_{j}\|P^{(i)} - PF^{(j)}\|^2\right)} \qquad (17)$$

where:

– N is the number of solutions obtained by the algorithm.
– $P^{(i)}$ denotes the i-th solution in the approximate Pareto set.
– $PF^{(j)}$ denotes the j-th point on the true Pareto front.
– $\min_{j} P^{(i)} - PF^{(j)}\|$ is the Euclidean distance between solution i and its nearest true Pareto point.
– A smaller GD value indicates better convergence ward the true Pareto front.

The comparative performance evaluation of MOPSO and NSGA-II was conducted under **identical experimental settings**, using the **ZDT1 benchmark problem**, the **same number of decision variables**, and identical **lower and upper bounds** across both algorithms. Each algorithm was executed with a **population size of 100** and **100 iterations (generations)** to ensure a fair and balanced comparison.

To obtain statistically meaningful results**, ten independent runs were performed for each optimizer,** and the final performance metrics were computed as averaged values of GD, SP, and HV. The ten-run averages clearly demonstrate the superior performance of MOPSO: it achieves a markedly lower convergence error (**GD = 0.001386**) than NSGA-II (**GD = 0.472701**), reflecting significantly closer proximity to the true Pareto front. Diversity is comparable between the two methods, with MOPSO obtaining **SP = 0.009028** and NSGA-II **SP = 0.008694**, indicating stable uniformity across solutions. However, the most notable difference appears in Pareto front coverage, where MOPSO achieves a substantially higher hypervolume (**HV = 0.865367**) compared to NSGA-II (**HV = 0.605660**), confirming stronger dominance and broader exploration of the objective space. To further illustrate the dynamic behavior of the algorithms, **one representative run** was selected from the ten runs, and the progression of GD, HV, and SP was plotted over the 100 generations for both approaches. In this selected run, Pareto front obtained are shown in Fig. 3

MOPSO consistently demonstrates superior performance with **GD = 0.003473**, **SP = 0.009298**, and **HV = 0.864456**, whereas NSGA-II obtains **GD = 0.218494**, **SP = 0.005920**,

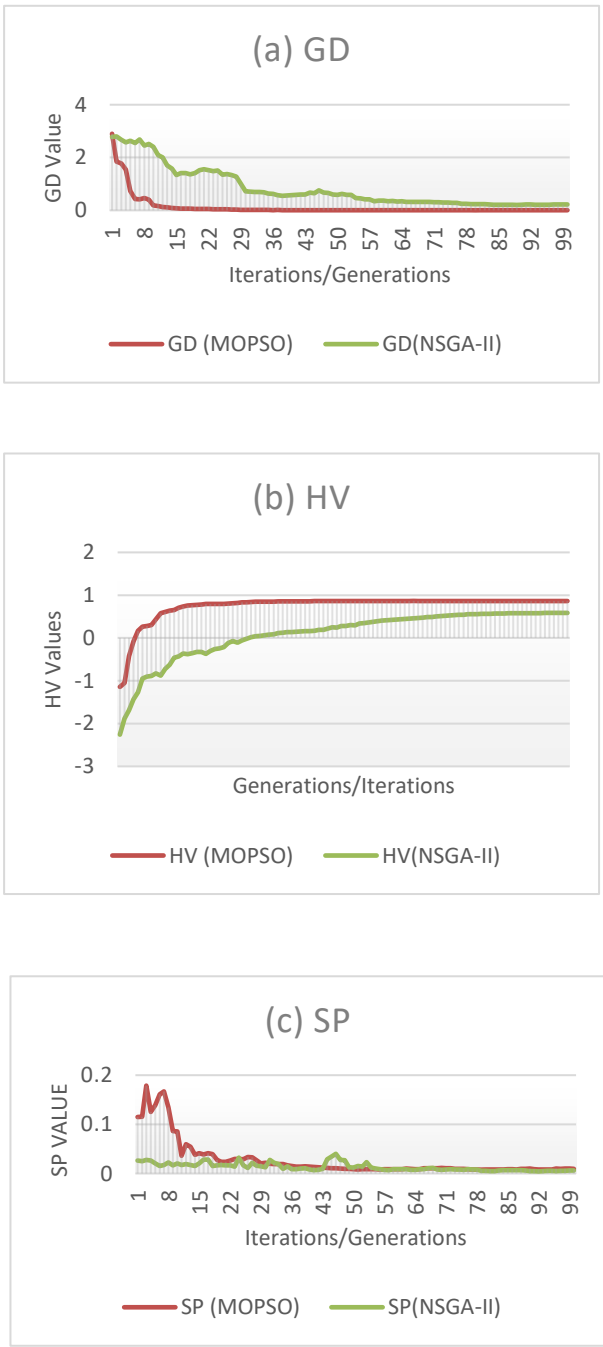


Fig. 3 SP , GD and HV Evaluation indicators

and **HV = 0.586174**. SP , GD and HV Evaluation indicators in Fig. 4

These results reaffirm that MOPSO converges more rapidly and achieves broader Pareto front coverage under identical experimental conditions, while NSGA-II shows slower convergence and more limited hypervolume expansion despite maintaining acceptable diversity and the values of GD, SP and HV improve if we increased number of Generations.

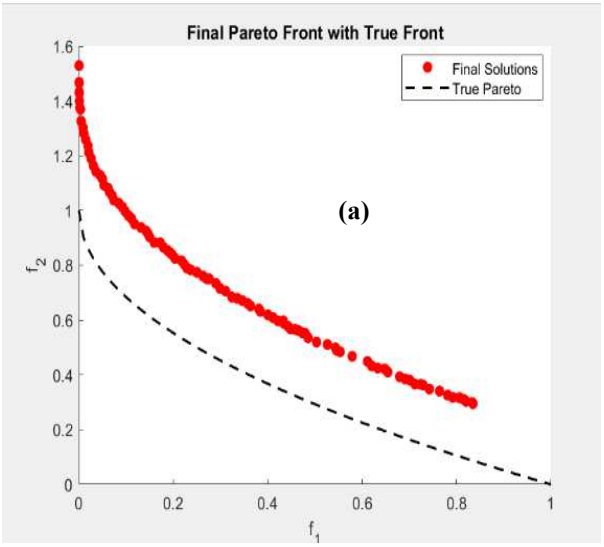


Fig. 4 a) MOPSO and b) NSGA-II Tests Pareto fronts

5. Simulation Verification

5.1. AR4 Manipulator Modeling

In this study, the 6-DOF AR4 manipulator developed by Annin was utilized. The link coordinate system is illustrated in Fig. 5, and the corresponding link parameters are presented in Table 1. The kinematic constraints for each joint of the manipulator are shown in Table 2, while Table 3 provides the sequence of positions passed by the joints.

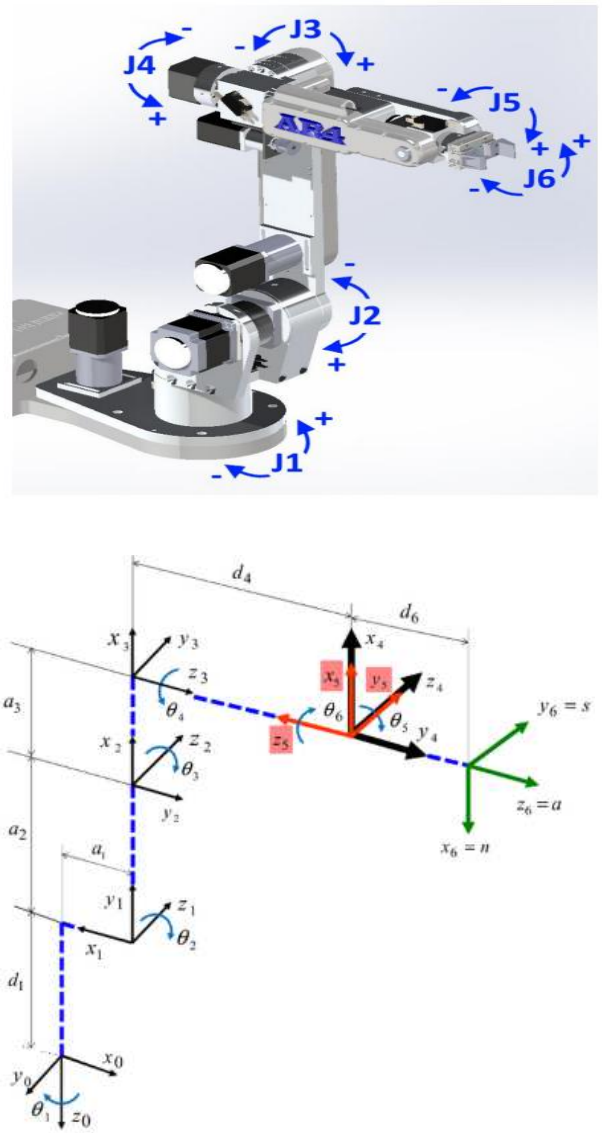


Fig. 5 The link coordinate system of AR4 6-DOF robot

Table 1 link parameters

Link i	α_{i-1} (°)	a_{i-1} (mm)	θ_i (°)	d_i (mm)	Range (°)
1	-90	64.2	θ_1	169.77	(–170, 170)
2	0	305	θ_2 - 90	0	(–42, 90)
3	90	0	θ_3 + 180	0	(–89, 52)
4	-90	0	θ_4	222.63	(–165, 165)
5	90	0	θ_5	0	(–105, 105)
6	0	0	θ_6	36.25	(–155, 155)

Table 2 kinematic constrains.

Link i	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
v_{max} (°/s)	148	148	148	178	178	178
a_{max} (°/s²)	74	74	74	89	89	89
j_{max} (°/s³)	100	100	100	120	120	120

Table 3 path defining.

Path points	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
1	0°	0°	0°	0°	0°	0°
2	30°	−10°	15°	30°	20°	25°
3	60°	−25°	35°	60°	40°	50°
4	90°	−35°	20°	90°	70°	75°
5	120°	−20°	0°	120°	90°	100°
6	150°	0°	-20°	150°	60°	125°
7	160°	30°	-40°	150°	30°	140°
8	140°	60°	-60°	130°	0°	120°

5.2. MOPSO AR4 Simulation

5.2.1. Pareto front

The MOPSO algorithm was configured with a swarm size of 100 particles (n_{pop} =100) over 100 iterations (MaxIt = 100), using a repository of 100 solutions (n_{rep} =100), inertia weight w = 0.5 with damping w_{damp} =0.99, cognitive and social coefficients c_1 =$=$ 1.5 and c_2 =2, and grid-based leader selection parameters n_{Grid} =100, α = 0.1, β = 2, γ = 2, μ = 0.5 .A time variable lower limit $t_{\min} = 2.5$ s and an upper limit of $t_{\max} = 5$ s were set.

Under these settings, the algorithm produced a smooth and well-distributed Pareto front illustrating trade-offs among total motion time, acceleration energy, and jerk energy. To illustrate the dynamic behavior of the algorithms, **one representative run** was selected from the ten runs Two representative solutions were selected for detailed comparison. The Pareto optimal front obtained from the simulation is shown in Fig. 6.

The maximum-time trajectory (86) required 33.5307 s and produced much lower acceleration (409.0709) and jerk (174.3367). with obtained optimal time sequence:
$\Delta t = [\,5,\ 3.5306,\ 5,\ 5,\ 5,\ 5,\ 5\,]$.

The minimum-time trajectory (72) completed the motion in 17.5037 s, but with much higher acceleration (3324.9413) and jerk (5929.4317) with the following optimal time sequence:
$\Delta t = [2.5000, 2.5000\ , 2.5000, 2.500, 2.500, 2.500, 2.5037]$ The time sequence of MOPSO algorithm was applied to simulate the robotic arm model, resulting in the motion trajectory passing through the path points as depicted in Fig. 7.

These extremes clearly highlight the trade-off in multi-objective optimization: Speed-critical tasks may prefer the fast (72) trajectory despite its high dynamic effort. Energy-efficient or precision-sensitive applications, such as industrial assembly, cooperative manipulation, or medical robotics, may benefit more from the smoother and slower (86) trajectory.

5.2.2. Joint space profiles

Analysis of all six joints showed consistent patterns across the fastest and slowest trajectories. The time-optimal solution (72) exhibited higher velocities, accelerations, and jerk, reflecting aggressive, dynamic motion. whereas the energy-efficient solution (86) produced smooth, gradual joint profiles with minimal dynamic peaks. Fig.8 illustrates the velocity, acceleration, and jerk curves of each joint.

All trajectories satisfied waypoint constraints, boundary conditions (zero initial and final velocity and acceleration), joint limits, and kinematic constraints, demonstrating that the MOPSO-optimized motion is feasible and robust. The end-effector accurately followed the 3D paths through all eight designated waypoints. The clear differences between the two extremes emphasize the algorithm's ability to generate solutions tailored to different operational priorities.

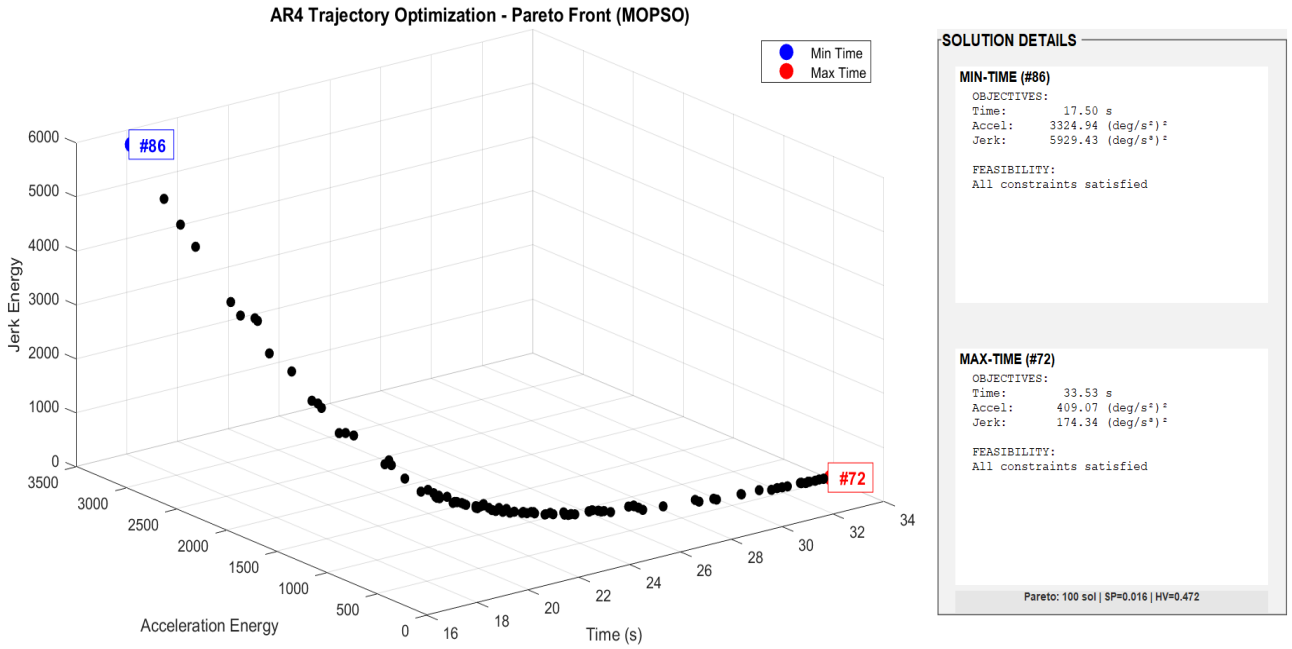


Fig. 6 Pareto Front of MOPSO

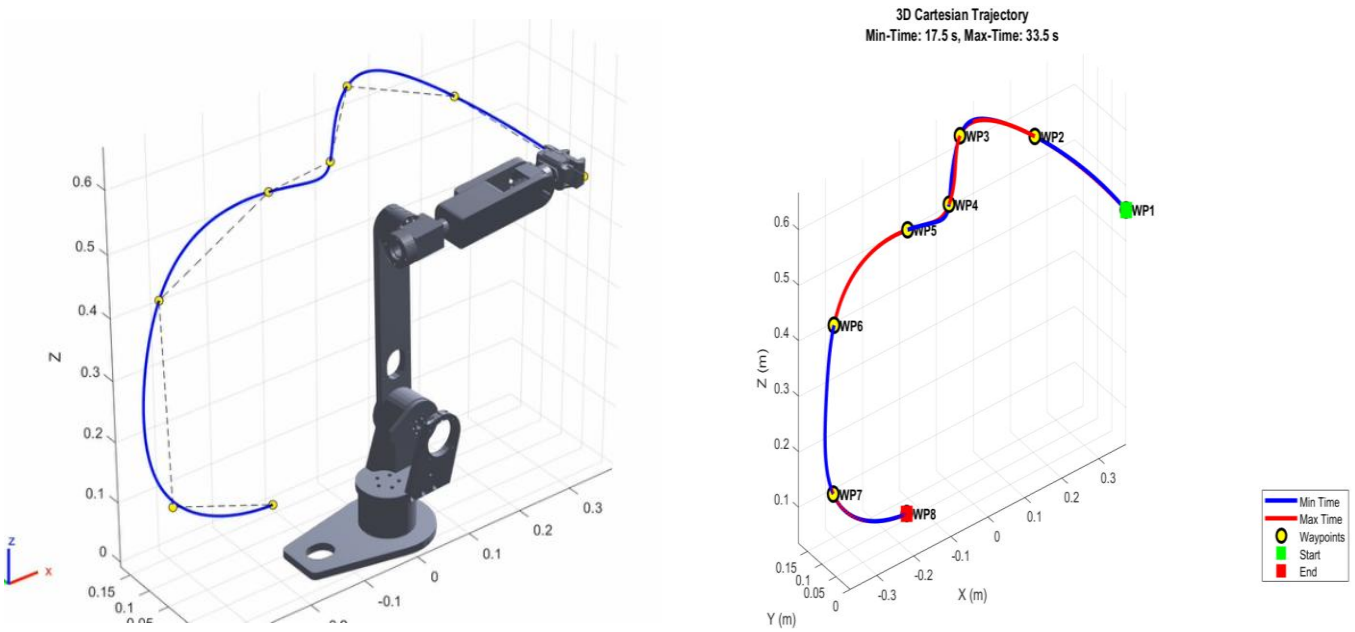


Fig. 7 MOPSO Motion trajectory passing through path points

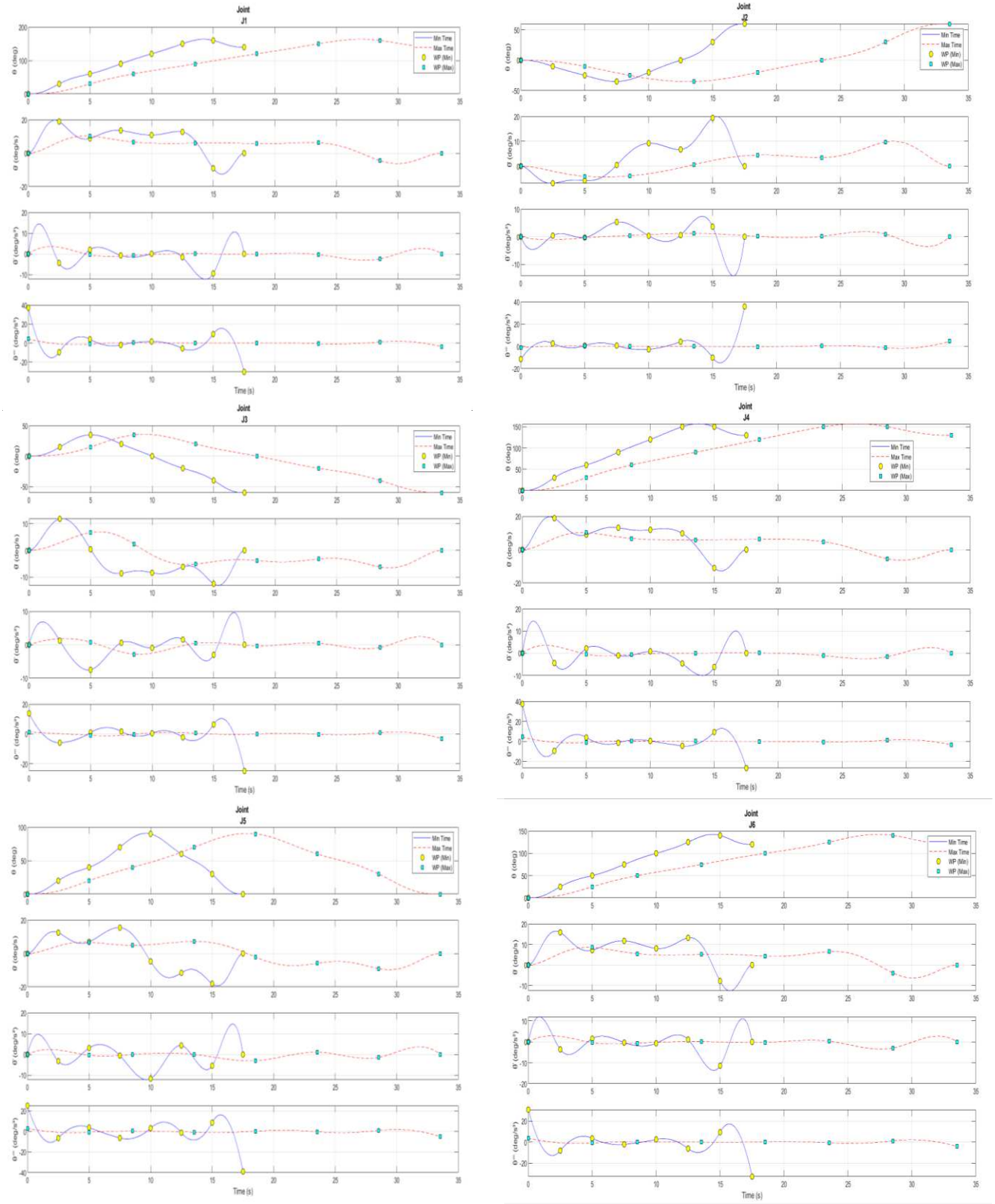


Fig. 8 MOPSO Joint trajectory-time sequence, Velocity, acceleration and jerk curves of each joint

5.2.3. Evaluation Metrices

Performance evaluation over ten independent runs yielded mean SP = 0.02119, confirming well-distributed and convergent Pareto fronts. The chosen representative run, achieving **SP = 0.01576,** reflects performance that is fully aligned with the overall statistical trend. The SP of the MOPSO algorithms was calculated, as shown in Fig.9.

These metrics collectively validate the stability, diversity maintenance, and overall effectiveness of the MOPSO configuration. The analysis of the two selected extreme solutions further shows how different optimization objectives—time versus smoothness—can be leveraged depending on the specific industrial, medical, or research application of the robot.

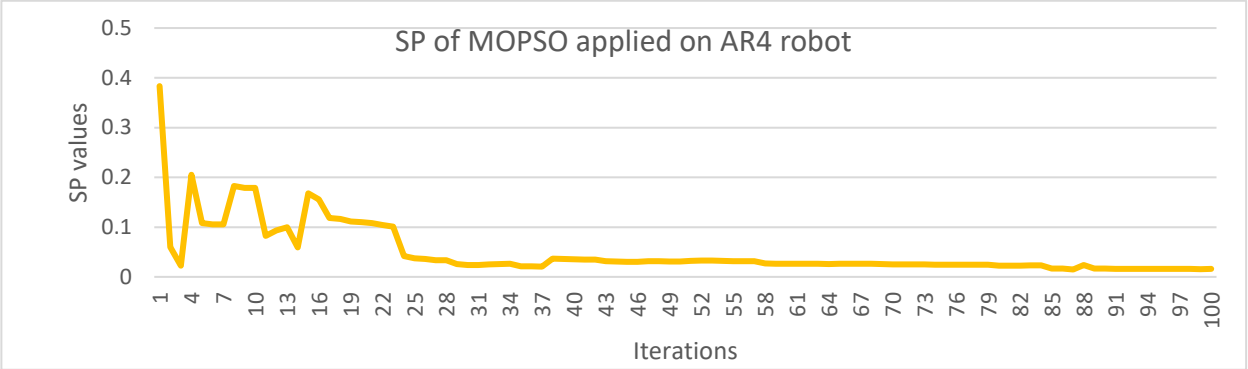


Fig. 9 MOPSO Evaluation indicators and Pareto fronts.

5.3. NSGA-II AR4 Simulation

5.3.1. Pareto front

The NSGA-II algorithm was configured with a population of **100 individuals** and **100 generations**, using a crossover fraction of **0.7** and adaptive feasible mutation with a strength of **0.3** to maintain a balanced search between exploration and refinement. Constraint handling used a strict tolerance of $\mathbf{1\times10^{-12}}$, and early stopping was disabled to allow full evolutionary progression. A time variable lower limit $t_{\min} = 2.5$ s and an upper limit of $t_{\max} = 5$ s were set. Under this setup, NSGA-II produced a smooth and well-distributed Pareto front that clearly revealed the trade-offs among motion time, acceleration energy, and jerk energy for the AR4 trajectory optimization problem.

To illustrate the dynamic behavior obtained from NSGA-II, one representative run from the ten independent simulations was examined in detail. From its resulting Pareto front, two extreme trajectories were selected to clearly demonstrate the inherent trade-offs between execution time, acceleration energy, and jerk energy. The Pareto optimal front obtained from the simulation is shown in Fig. 10.

The **maximum-time trajectory (15)** required **33.5064 s**, producing **substantially lower dynamic effort**, with a total acceleration energy of **412.58** and a jerk energy of **182.53**. with obtained optimal time sequence:

$$\Delta t = [4.9828, 3.9779, 4.8572\ , 4.8731, 4.8987, 4.9287, 4.9875].$$

This solution corresponds to a **smooth, low-stress motion profile**, prioritizing minimal dynamic load at the expense of longer execution time. It reflects an energy-efficient trajectory with reduced vibration and improved stability. In contrast, the **minimum-time trajectory (78)** completed the motion in only **18.5567 s** but demanded **much higher acceleration (2266.63)** and **jerk (3075.47)**. This represents a **high-aggressiveness motion**, where the algorithm prioritizes rapid task completion while sacrificing smoothness and increasing mechanical stress. with the following optimal time sequence:

$$\Delta t = [2.9218, 2.5419, 2.5720, 2.5551, 2.5643, 2.5453, 2.8559]\ .$$

The time sequence of NSGA-II algorithm sequence was applied to simulate the robotic arm model, resulting in the motion trajectory passing through the path points as depicted in Fig. 11.

These two extreme solutions clearly highlight the fundamental trade-off in multi-objective trajectory optimization using NSGA-II. Applications that demand **speed and rapid response**—such as high-throughput industrial manipulation, emergency actions, or competitive pick-and-place—may favor the fast solution (#78) despite its

elevated dynamic cost. Conversely, tasks requiring **precision, safety, or energy conservation**, including medical robotics, human-robot collaboration, and long duration operation—can benefit more from the smooth and slower trajectory (#15), where reduced acceleration and jerk improve stability and prolong actuator lifespan.

5.3.2. Joint space profiles

Analysis of all six joints showed consistent patterns across the fastest and slowest trajectories. The time-optimal solution (78) exhibited higher velocities, accelerations, and jerk, reflecting aggressive, dynamic motion. whereas the energy-efficient solution (15) produced smooth, gradual joint profiles with minimal dynamic peaks. Fig.12 illustrates the velocity, acceleration, and jerk curves of each joint.

All trajectories satisfied waypoint constraints, boundary conditions (zero initial and final velocity and acceleration), joint limits, and kinematic constraints, demonstrating that the NSGA-II -optimized motion is feasible and robust. The end-effector accurately followed the 3D paths through all eight designated waypoints. The clear differences between the two extremes emphasize the algorithm's ability to generate solutions tailored to different operational priorities.

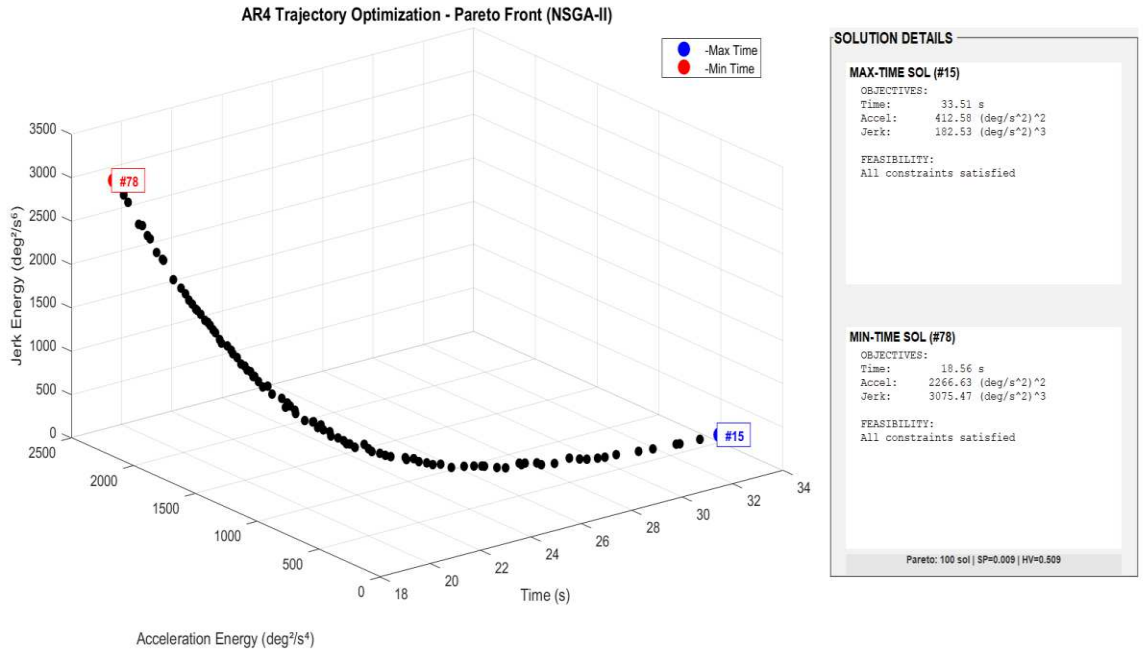


Fig. 10 The NSGA-II Pareto optimal fronts

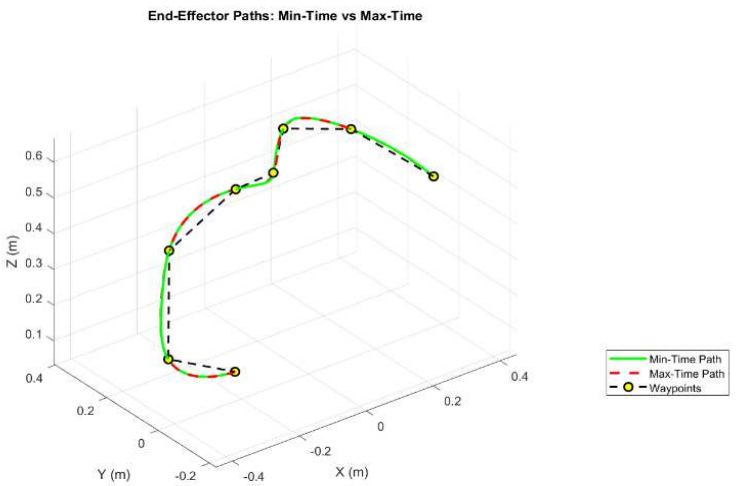


Fig. 11 NSGA-II Motion trajectory through path points

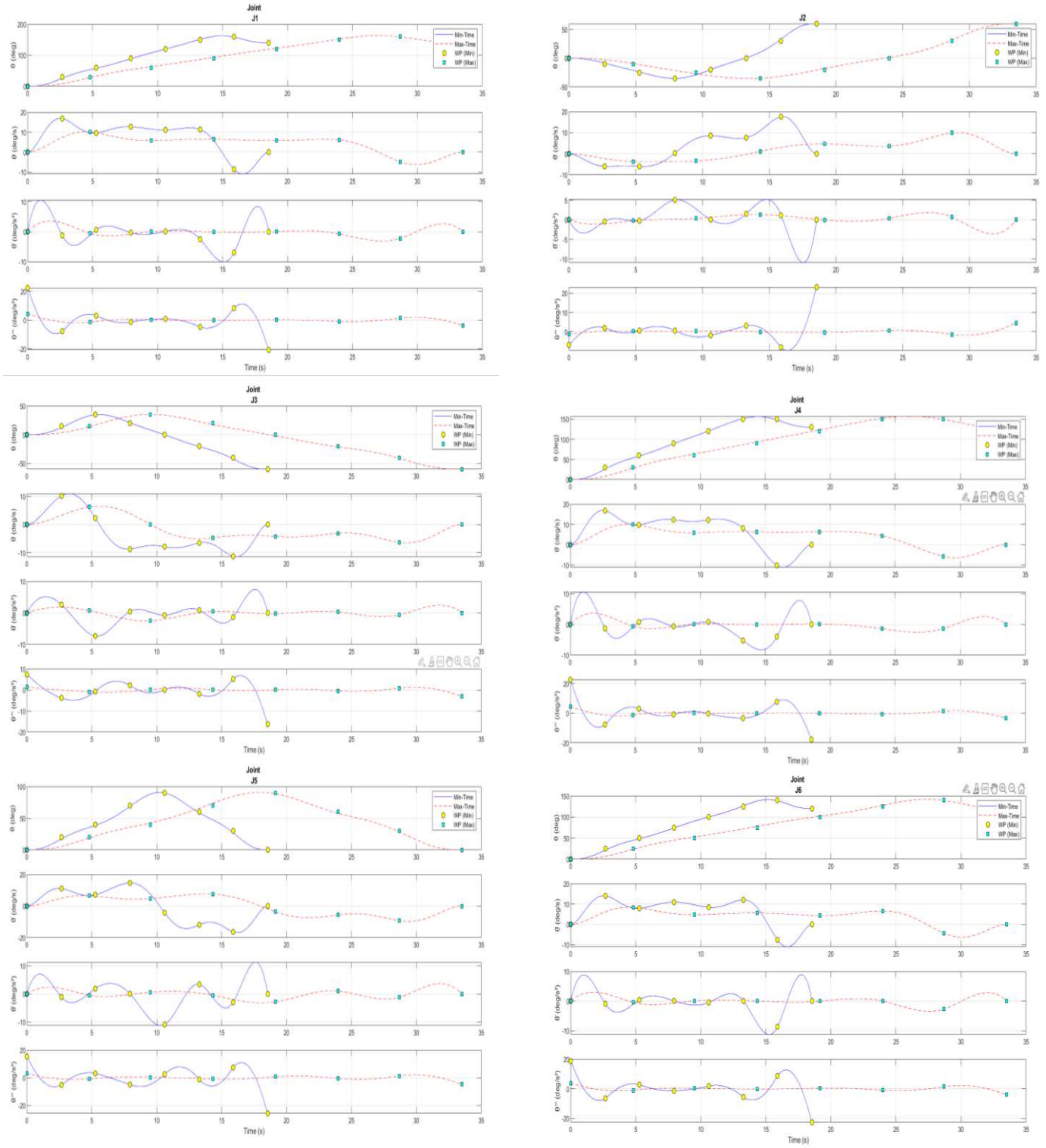


Fig. 12 NSGA-II Joint trajectory-time sequence, Velocity, acceleration and jerk curves of each joint

5.3.3. Evaluation Metrices

Performance evaluation over ten independent runs yielded mean SP = **0.01166 confirming** well-distributed and convergent Pareto fronts.

The chosen representative run, achieving **SP = 0.0138** reflects performance that is fully aligned with the overall statistical trend. The SP of the NSGA-II algorithms was calculated, as shown in Fig.13.

Same as MOPSO These metrics collectively validate the stability, diversity maintenance, and overall effectiveness of the NSGA-II configuration.

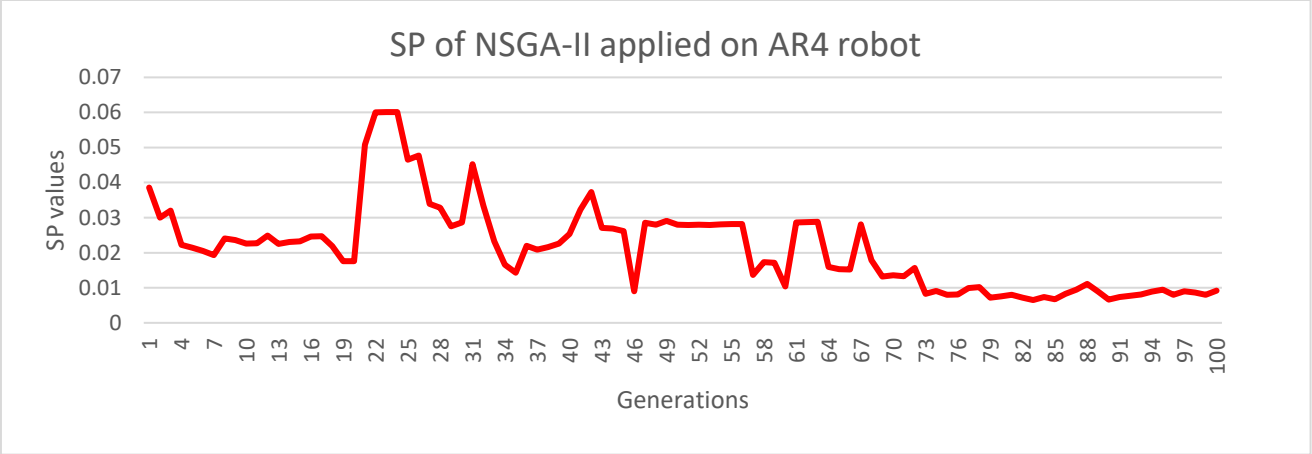


Fig. 13 NSGA-II Evaluation indicator SP.

5.4. Comparative Analysis

According to the simulation run selected and across the full optimization process, the two algorithms exhibit markedly different dynamic behaviors in terms of diversity, convergence, and overall stability.

Across 100 iterations, the SP results shown in Fig. 14 reveal clear behavioral differences between NSGA-II and MOPSO.

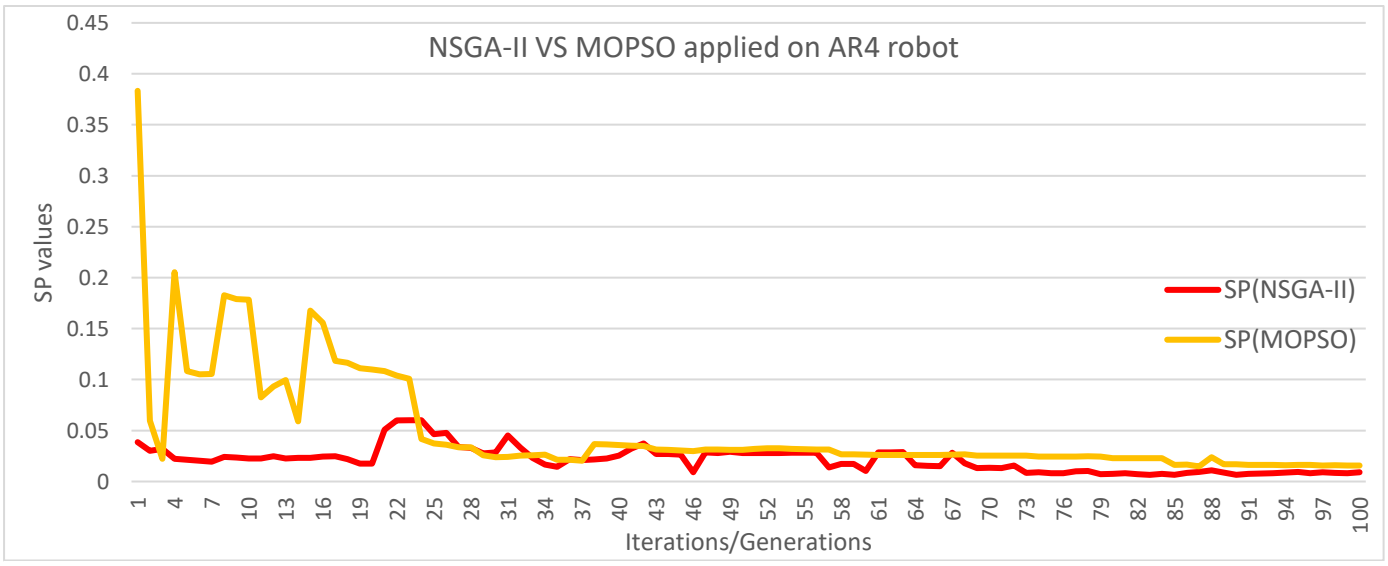


Fig. 14 Evaluating indicator (SP) comparison.

In the early stage (iterations 1–20), MOPSO exhibits significant variability in solution spacing during initial iterations, reflecting its strong global exploration capability at the expense of early diversity maintenance, with SP jumping from very small values (~0.02) to extremely large values (up to 0.38). This reflects its strong exploratory behavior and poor early uniformity. NSGA-II behaves more steadily, with SP remaining within a narrower range of 0.017–0.06, indicating better spacing control during early exploration.

During the mid-optimization phase (iterations 40–60), both algorithms begin to stabilize. NSGA-II maintains consistent SP values between 0.022 and 0.033, showing an increasingly structured distribution of solutions. MOPSO also improves, with SP mostly between 0.026 and 0.035. Although slightly more variable than NSGA-II, MOPSO achieves comparable performance in this phase, indicating a more balanced exploration–exploitation behavior.

In the final iterations (80–100), NSGA-II achieves its most stable and lowest SP values, typically between 0.006 and 0.012, reflecting a highly uniform final Pareto front. MOPSO also stabilizes, with SP generally between 0.015 and 0.024, but still exhibits more variability than NSGA-II due to its swarm-driven dynamics. Overall, NSGA-II maintains more consistent and uniform spacing across all phases, especially at convergence.

MOPSO improves over time and reaches acceptable uniformity, but its final spacing remains slightly less stable than NSGA-II. This confirms NSGA-II's advantage in producing a more evenly distributed Pareto set, while MOPSO preserves stronger exploration at the cost of uniformity.

6. Conclusions

This research establishes a comprehensive multi-objective trajectory optimization framework that systematically evaluates and compares the performance characteristics of MOPSO and NSGA-II for 6-DOF robotic manipulators, targeting minimum execution time, acceleration energy, and jerk energy. Quintic B-spline trajectories are used to ensure smooth, vibration-free joint motion.

Benchmarking on ZDT1 reveals clear behavioral differences. MOPSO shows extremely fast convergence with very low GD values (0.001386 average; 0.003473 in the selected run), far outperforming NSGA-II (0.472701 average; 0.218494 in the selected run). SP values on ZDT1 are similar (MOPSO: 0.009028; NSGA-II: 0.008694), though NSGA-II maintains more stable spacing through the iterations. MOPSO also achieves a much higher HV (0.865367 vs. 0.605660), indicating broader and quicker front coverage. These results confirm MOPSO's rapid convergence and expansive search behavior, while NSGA-II offers smoother and more consistent refinement.

When applied to the AR4 case study. MOPSO achieves SP = 0.02119, whereas NSGA-II reaches a lower SP = 0.01166, showing that NSGA-II produces a more uniform and well-distributed Pareto front for the robot problem. This aligns with its stable spacing behavior observed in ZDT1, while MOPSO's higher SP indicates greater variability in complex real-world search spaces

7. Future work

This study has presented a comparative analysis of MOPSO and NSGA-II for 6-DOF robotic trajectory optimization, demonstrating their complementary behaviors and proposing an innovative hybrid framework. The key contributions include:

- comprehensive performance benchmarking on both standard test functions and real-world manipulator applications
- quantitative verification of MOPSO's rapid convergence versus NSGA-II's distribution uniformity
- A discussion and illustration of a sequential MOPSO→NSGA-II hybrid strategy as a potential approach for future research

The results obtained from both the ZDT1 benchmark and the AR4 trajectory optimization clearly show that MOPSO and NSGA-II possess complementary strengths that can be effectively combined. MOPSO consistently demonstrates **rapid global convergence**, reaching the neighborhood of the true Pareto front in far fewer generations, as evidenced by its very low GD values. NSGA-II, on the other hand, provides **superior refinement and distribution stability**, achieving lower SP values and producing more uniformly spaced final Pareto fronts. **Based on these observations, a promising recommendation for future research** is the development of a series hybrid MOPSO–NSGA-II algorithm, where MOPSO executes the initial exploration phase and quickly locates high-quality solution regions, after which NSGA-II takes over to refine, redistribute, and regularize the Pareto front**.**

To support this concept, a **hybrid algorithm flowchart** has been prepared in Fig. 15, illustrating the two-phase structure of the proposed method: (1) a MOPSO-based fast-convergence stage, and (2) a NSGA-II refinement stage. This framework is expected to combine the speed of swarm intelligence with the strong elitism and diversity preservation of evolutionary operators, improving overall convergence quality, spacing uniformity, and solution

robustness in complex multi-objective robotic trajectory planning. Future implementations of this hybrid approach may achieve significantly better performance than using either algorithm alone, particularly in real-world applications such as the AR4 manipulator.

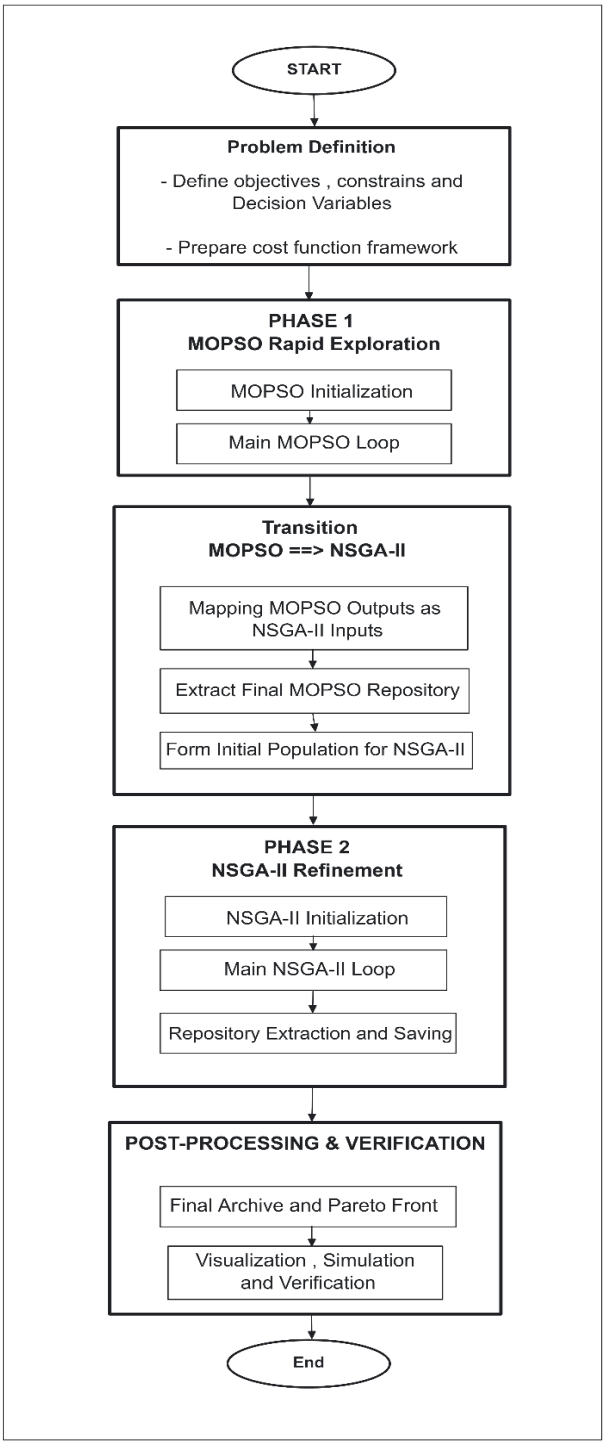


Fig. 15 MOPSO-NSGA-II Hybrid algorithm Flow chart

Funding this research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

7. References

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling Multiple Objectives with Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279.

Cui, L. L., Wang, H., & Chen, W. (2019). Trajectory planning of a spatial flexible manipulator for vibration suppression. *Robotics and Autonomous Systems*, *123*, 103316.

De Boor, C. (1978). *A Practical Guide to Splines*. Springer.

Deb, K. (2001). *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist Mult objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Francisco, R., Carlos, L., & Francisco, V. (2021). Multi-objective optimization of costs and energy efficiency associated with autonomous industrial processes for sustainable growth. *Technological Forecasting and Social*, *173*, 121115.

Guo, Y., & Lai, G. (2020). Review of joint space trajectory planning and optimization for industrial robots. *Journal of Mechanical Transmission*, *44*(2), 154–165.

Hou, J., Du, J., & Chen, Z. (2023). Time-optimal trajectory planning for the manipulator based on improved non-dominated sorting genetic algorithm II. *Applied Sciences*, *13*(11), 6757.

Hu, S., Wan, Y., Liang, X., Hou, J., & Zhang, S. (2024). Multi-objective trajectory optimization of 6-DOF manipulator based on improved NSGA-II algorithm. *Advances in Mechanical Design, Mechanisms and Machine Science*,

Fig 16 MOPSO-NSGA-II Hybrid algorithm Flow chart

2451–2466. https://doi.org/10.1007/978-981-97-0922-9_156

Huang, J. S., Hu, P. F., Wu, K. Y., & others. (2018). Optimal time-jerk trajectory planning for industrial robots. *Mechanism and Machine Theory*, *121*, 530–544.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks*, 1942–1948.

Lan, J. Y., Xie, Y. G., Liu, G. J., & others. (2020). A multi-objective trajectory planning method for collaborative robots. *Electronics*, *9*(5), 859.

Liu, Y. Y., & Zhang, X. Y. (2022). Trajectory optimization for manipulators based on external archives self-searching multi-objective particle swarm optimization. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *236*(2), 1188–1201.

Ramabalan, S., Saravanan, R., & Balamurugan, C. (2009). Multi-objective dynamic optimal trajectory planning of robot manipulators in the presence of obstacles. *International Journal of Advanced Manufacturing Technology*, *41*(5–6), 580–594.

Reyes-Sierra, M., & Coello, C. A. C. (2006). Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, *2*(3), 287–308.

Saravanan, R., Ramabalan, S., & Balamurugan, C. (2009). Evolutionary multi-criteria trajectory modeling of industrial robots in the presence of obstacles. *Engineering Applications of Artificial Intelligence*, *22*(2), 329–342.

Shi, X. L., Fang, H. G., & Guo, L. J. (2016). Multi-objective optimal trajectory planning of manipulators based on quintic NURBS. *2016 IEEE International Conference on Mechatronics and Automation (ICMA)*, 759–765.

Shi, X. L., Fang, H. G., Pi, G., & others. (2018). Time-energy-jerk dynamic optimal trajectory planning for manipulators based on quintic NURBS. *2018 3rd International Conference on Robotics and Automation Engineering (ICRAE)*, 44–49.

Wang, K. F., Zhang, Y. B., & Huang, J. W. (2023). Multi-objective trajectory planning of manipulator based on improved NSGA-II algorithm. *Chinese Journal of Construction Machinery*, *21*(3), 215–220.

Wang, Z. S., Li, Y. B., Shuai, K., & others. (2022). Multi-objective trajectory planning method based on the improved elitist non-dominated sorting genetic algorithm. *Chinese Journal of Mechanical Engineering*, *35*(1), 1–15.