Adaptive Position Updating Particle Swarm Optimization for UAV Path Planning

Junhao Wei, Yanzhao Gu, K. L. Eddie Law, Ngai Cheong
Faculty of Applied Sciences
Macao Polytechnic University
Macao, China
{P2312195, P2311998, eddielaw, ncheong}@mpu.edu.mo

***Abstract*—In the paper, we propose a path planning method for unmanned aerial vehicles (UAVs) based on an enhanced Particle Swarm Optimization (PSO) algorithm. To address the issue of classical PSO algorithms easily converging to local optima, we introduce a dynamic inertia weight adjustment strategy. Additionally, we incorporate Tent mapping initialization, Levy flight, and adaptive *t*-distribution to balance both global and local search capabilities. We validate the effectiveness of the improved algorithm through simulations on various benchmark functions and UAV path planning scenarios. Experimental results demonstrate significant enhancements in convergence speed, overall solution accuracy, and stability. Our improved PSO algorithm provides effective solutions for diverse UAV path planning problems in complex environments and lays the foundation for further research on dynamic environments and three-dimensional path planning in future.**

***Index Terms*—Levy flight, Adaptive *t*-distribution, Particle Swarm Optimization (PSO), path planning, unmanned aerial vehicle (UAV)**

I. Introduction

The Particle Swarm Optimization (PSO) [1] algorithm is inspired by observing message transmissions during the foraging behavior of bird flocks or fish. The theoretical basis of the PSO algorithm is to take a single particle as, for example, an individual bird in a flock of birds. In the algorithm, a particle is given a memory, and is able to find the optimal solution by interacting with other particles in the swarm. In fact, each particle in PSO represents a potential solution to the problem. Meanwhile, the position, velocity, and fitness value represent different characteristics of a particular particle. The fitness value is calculated by a fitness function and its value indicates the merit of the particle.

The particles move in the search space and each one updates the individual position by tracking an individual's best value, p_{best}, and the population's best value, g_{best}. The individual's best value p_{best} is the local best position of the fitness value calculated from the positions experienced by the individual. The population's best value g_{best} is the global best position of the fitness values discovered by all particles in the population.

PSO can be used to solve various optimization problems [2], such as function optimization, neural network training, etc. The path planning of an unmanned aerial vehicle (UAV) is exactly to find an optimal path that is planned after considering the surrounding environment factors, such as the altitude distance factors and obstacle constraints. PSO is thus one possible choice for planning the travel path of an UAV.

Although PSO offers numerous advantages, including its simple principles, ease of implementation, and fast convergence speed, it has notable disadvantages, for example, a lack of dynamic speed adjustment, susceptibility to getting trapped in local optima, which result in low convergence accuracy, and difficulty in achieving convergence. To address these problems, Shi et al. [3] introduced a dynamic inertia adjustment strategy. In the early stage of the algorithm, a larger inertia weight is used to promote the global search. In the later stage, it is gradually reduced to improve the accuracy of the local search. This dynamic inertia weight adjustment method can adaptively adjust the search behavior at different stages of the algorithm. This design effectively balances the relationship between the global and local exploitations, and then improves the overall performance of the algorithm.

In [4], Kennedy et al. introduced a local topology into the particle swarm algorithm. They restricted the neighbors of each particle to a fixed-size region, such that each particle only exchanges information with a portion of its neighbors. This method improved the convergence and stability of the PSO algorithm. Usually, local topology can be classified into various types, including ring, lattice, and star, etc., where each structure has its specific advantages and application scenarios. Local topology prevents premature global convergence by restricting the range of information propagation, effectively increase population diversity and enhance the robustness of the algorithm.

In [5], an idea of chaotic initialization was recommended. The chaos theory wss used to initialize the particle population and perturb the search process in order to improve the global search capability and convergence speed. Chaotic initialization used the traversal and randomness of chaotic sequences to generate initial particle positions, avoiding the problem of lack of population diversity that may result from traditional random initialization. Meanwhile, the introduction of chaotic perturbation in the search process could help particles jump out of the local optimum, enhance the global search ability and improve the convergence efficiency. The introduction of chaos theory enabled the Particle Swarm Optimization algorithm to

The authors appreciate the financial support by Macao Polytechnic University (MPU Grant No.: RP/FCA-06/2022) and Macao Science and Technology Development Fund (FDCT Grant No.: 0044/2023/ITP2).

ISBN 978-3-903176-65-2 © 2024 IFIP

cope with complex optimization problems more effectively, especially in high and multi-peak function optimization.

In recent years, various approaches on improving the stability of PSO algorithms have been proposed and tested. Methods such as dynamic parameter tuning, diversity maintenance, hybrid strategy and adaptive mechanism have been successively applied to particle swarm optimization. Dynamic parameter adjustment includes the adaptive adjustment of parameters such as inertia weights, acceleration coefficients and velocity limits to better adapt to the different demands in the search process. The diversity maintenance strategy prevents premature convergence by introducing crossover and mutation operations and maintaining population diversity. Hybrid strategies, on the other hand, combine the advantages of other optimization algorithms such as genetic algorithms, simulated annealing and differential evolution to further enhance the performance of PSO. And adaptive mechanisms enable the algorithms to dynamically adjust their behaviour according to the feedback information during the search process, thus improving the optimization efficiency and problem solving ability.

The traditional particle swarm algorithm is easy to fall into the local optimal problem. Inspired by the grey wolf optimizer, a speed classification system is introduced into the traditional particle swarm optimization and the C-GWPSO hybrid algorithm is proposed [6]. Zhang et al. proposed a strategy for escaping local convergence based on a variable neighborhood search algorithm and established a criterion for the initiation probability of local search, achieving a balance between escaping local convergence and computational overhead. These methods improve the particle swarm optimization algorithm from different aspects, and improve the convergence speed and accuracy of the algorithm in different degrees.

In this paper, we propose an adaptive position updating particle swarm algorithm. It uses Tent mapping in population initialization to increase the randomness of the initialized population and increase the exploration ability of the algorithm. During individual particle updates, it uses a mixture of Levy flight strategy [7] and adaptive *t*-distribution strategy to help the particles to jump out of the local optima. Meanwhile, it adopts a dynamic inertia weight adjustment strategy to balance the its global and local searching ability and thus improve its optimization capability.

II. The Original PSO

In PSO, the position of a particle i in an N-dimensional space is represented by $X_i = (x_1, x_2, \ldots, x_N)$, and its velocity is $V_i = (v_1, v_2, \ldots, v_N)$. The fitness value of a particle is determined by an objective function. The particle knows its best position p_{best} and its present position. This can be seen as its own experience of flight. Additionally, each particle is aware of the best position discovered by any other particles in the entire swarm so far, which is called g_{best}. The g_{best} should be the best value among all p_{best} values. This can be considered as the experience of the peers. The particle then decides its next movement based on its own experience and the best experience among its peers.

PSO initializes a random swarm of particles and finds the optimal solution with each iteration. In each iteration, particles update their positions and velocities by tracking the p_{best} and g_{best}:

$$V_{i+1} = \omega \cdot V_i + c_1 \cdot r_1 \cdot (p_{best} - X_i) + c_2 \cdot r_2 \cdot (g_{best} - X_i) \quad (1)$$

$$X_{i+1} = X_i + V_i \quad (2)$$

where ω is the inertia weight; c_1, and c_2 are acceleration factors (usually set to 2); r_1, and r_2 are random numbers set between 0 and 1.

III. Adaptive Position Update PSO

A. Tent Mapping

Traditional PSO uses random initialization to generate an initial population. The pseudo-randomness can result in uneven particle distribution in the search space, and affect the global search capability. However, in the field of optimization, chaotic mapping can be used as a substitute for pseudo-random number generators to generate chaotic numbers between 0 and 1. Experiments have shown that using chaotic sequences for population initialization, selection, crossover, and mutation can influence the entire process of the algorithm, often achieving better results than pseudo-random numbers.

This paper introduces Tent mapping as a new initialization method. The Tent mapping is a chaotic mapping. It generates an initial population uniformly distributed in the search space, enhancing the algorithm's global search ability. Its expression is:

$$x(n+1) = \begin{cases} \frac{x(n)}{\alpha}, & \text{for } 0 \le x(n) \le \alpha \\ \frac{1-x(n)}{1-\alpha}, & \text{for } \alpha < x(n) \le 1 \end{cases} \quad (3)$$

where α is the chaos parameter which is set to 0.5.

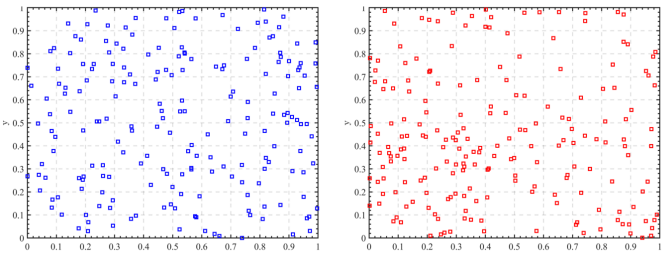

Fig. 1. Particles initialized by Random Number vs Tent Mapping

As shown in Fig. 1, the particles initialized by random number generator may exhibit dense and sparse regions, resulting in a relatively less random distribution. On the other hand, the particles initialized by Tent mapping tend to be more evenly distributed throughout the search space, as shown in Fig. 1.

B. Position Update Strategies

Levy Flight Strategy:
Levy flight is a random walk strategy with a long-tailed distribution, allowing particles to explore the solution space through long jumps. This property introduces randomness to a certain extent. And thus enhances exploration and global search capabilities, helps particles escape local optima. In each iteration, some particles execute Levy flight to increase global search ability. Fig. 2 is a simulation of Levy flight.

In the improved algorithm, Levy flight generates a random perturbation vector RL

$$RL = (RL_1, Rl_2, \ldots, RL_n) \tag{4}$$

where n is the problem dimension.

The step length s follows a Levy distribution, with the probability density function:

$$f(s) = \frac{c}{s^{(1+\beta)}} \tag{5}$$

where c is a normalization constant; β is the Levy index with the value of 1.5, controlling the distribution characteristics of the step length.

The implementation of Levy distribution is complex. In practical applications, it is often simulated using the Mantegna algorithm.

The formula for the step size s is as follows:

$$s = \frac{u}{|v|^{\frac{1}{\beta}}} \tag{6}$$

where $u \sim N(0, \sigma_u^2)$ and $v \sim N(0, \sigma_v^2)$ and

$$\sigma_u = \frac{\tau(1+\beta) \cdot \sin(\frac{\pi\beta}{2})}{\tau\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}}^{\frac{1}{\beta}} \tag{7}$$

$$\sigma_v = 1 \tag{8}$$

Then, each component of the random perturbation vector RL can be expressed as:

$$RL_i = s_i \tag{9}$$

This means RL is a vector composed of step length s_i generated from the Levy distribution. Each component of this vector is an independent Levy-distributed random number generated by the method described above.

The random perturbation vector RL updates the particle position, maintaining the velocity to increase exploration. When the particle's solution shows no significant change over iterations, the Levy flight strategy helps it escape local optima.

$$X_{i+1} = X_i + V_i + (g_{best} - X_i) \cdot RL \tag{10}$$

Adaptive *t*-distribution:
The *t*-distribution, also known as Student's *t*-distribution, is characterized by its degrees of freedom parameter, a. The probability density function of the *t*-distribution with a degrees of freedom is given by:

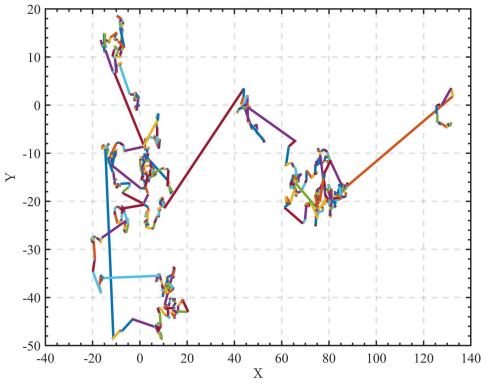

Fig. 2. Simulation of Levy flight

$$f(a) = \frac{\Gamma\left(\frac{a+1}{2}\right)}{\sqrt{a\pi} \cdot \Gamma\left(\frac{a}{2}\right)} \cdot \left(1 + \frac{b^2}{a}\right)^{-\frac{a+1}{2}} \tag{11}$$

where Γ denotes the Gamma function, a is the degrees of freedom, b is a random variable between negative infinity and positive infinity. The degrees of freedom parameter a affects the shape of the distribution curve. In this paper, a, which is set as 1 increase to e^4, is shown in Eqn.(13). As a approaches infinity, the curve converges to a Gaussian distribution, while as a approaches 1, the curve resembles a Cauchy distribution. The density function of the *t*-distribution as a varies is illustrated in the figure. To further enhance the optimization performance of PSO and prevent the algorithm from easily converging to local optima in the later stages of iteration, an adaptive t-distribution mutation strategy is employed for updating particle positions. The position update is expressed as:

$$X_{i+1} = g_{best} + t(a) \cdot g_{best} \tag{12}$$

$$a = \frac{1}{\exp\left(-4 \cdot \left(\frac{t}{T}\right)^2\right)} \tag{13}$$

Adaptive Position Updating Strategy:
Levy flight exhibits strong global search capabilities. When the current fitness is relatively high, incorporating Levy flight can help maintain diversity in the search process. Its substantial jump capability facilitates the exploration of previously undiscovered regions within the search space, allowing for the continued search of potentially better solutions and preventing premature convergence to local optima. The adaptive *t*-distribution can dynamically adjust the search range, enabling the algorithm to focus more intensely on local search when the fitness is good. This distribution is capable of adapting to different regions of the problem space during local search, thereby enhancing optimization efficiency. Combining Levy flight with the adaptive *t*-distribution allows the algorithm to dynamically adjust its search strategy based on the quality of

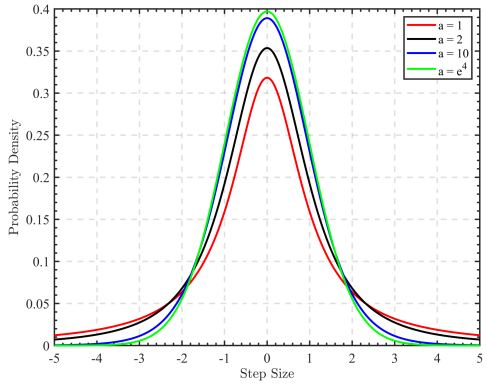

Fig. 3. Adaptive *t*-distribution

the current solution. Using Levy flight when fitness is high aids in exploring new regions, while employing the adaptive *t*-distribution when fitness is lower introduces a degree of perturbation, helping the algorithm escape local optima and achieving a better balance. Compared to the original PSO algorithm, this improvement offers enhanced capabilities and flexibility in both global exploration and local optimization. The integration of Levy flight and adaptive *t*-distribution contributes to better performance in complex optimization problems, accelerates convergence, and increases the robustness and adaptability of the algorithm.

C. Dynamic Inertia Weight Adjustment

The inertia weight ω reflects a particle's ability to maintain its previous velocity. Shi.Y first introduced inertia weight into PSO, pointing out that a larger inertia weight promotes global search, while a smaller one favors local search. To better balance global and local search capabilities, Shi.Y proposed a linear decreasing inertia weight (LDIW):

$$\omega(k) = \omega_{start} \cdot (\omega_{start} - \omega_{end}) \cdot \frac{(T_{max} - k)}{T_{max}} \tag{14}$$

where ω_{start} is the starting inertia weight; ω_{end} is the final inertia weight; k is the current iteration number; and T_{max} is the maximum iteration number [3].

In general, the algorithm performs best when ω_{start}=0.9, ω_{end}=0.4. This decreasing inertia weight adjustment strategy makes the algorithm have a larger inertia weight for the global search at the beginning of the iteration. As the inertia weight decreases at the end of the iteration, the smaller inertia weight makes the algorithm able to have a stronger local search ability. This strategy improves the defects of the algorithm to a certain extent. In addition, there are fitness-distance balance adjustment strategies, nonlinear inertia weight adjustment strategies and so on.

For the nonlinear inertia weight adjustment strategy, it is pointed out that the use of a convex function decreasing strategy may lead to large oscillations in the subsequent iterations [8]. But the use of a concave function decreasing strategy can effectively avoid local optima. Therefore, we adopt a concave function based inertia weight decreasing update strategy:

$$\omega(k) = \omega_{start} - (\omega_{start} - \omega_{end}) \cdot \left(\frac{k}{T_{max}}\right)^2 \tag{15}$$

where ω_{start} is the starting inertia weight; ω_{end} is the final inertia weight; k is the current iteration number; T_{max} is the maximum iteration number.

The curve of ω changing with the iterations is shown as Fig. 4. Assuming ω_{start}=0.9, ω_{end}=0.4 and T_{max}=500.

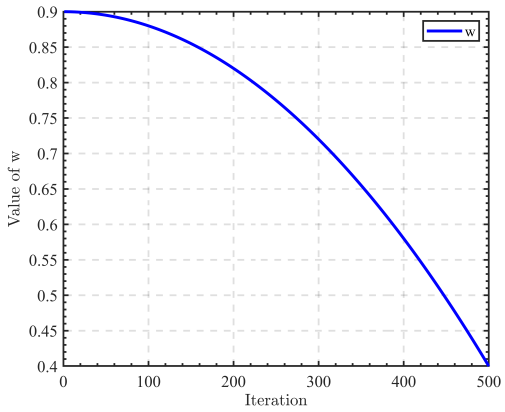

Fig. 4. The curve of ω changing with the iterations

D. Steps of adaptive position update PSO

The steps of the adaptive position updating particle swarm algorithm are as follows:

1) According to Eqn. (3), initialize particle positions and velocities using Tent mapping, calculate the fitness value of each particle. Then record the global best solution p^d_{best} and individual best solution $p^d_{best,i}$;
2) Determine whether the algorithm converges or not, if it converges then output p^d_{best}, if it does not converge then go to the next step;
3) Update the inertia weight according to Eqn. (15);
4) Update the particle velocity according to Eqn. (1);
5) Calculate the probability of adopting the position updating strategy and perform strategy selection. Execute Step (a) if Levy flight strategy is selected. Execute Step (b) if self-adaptive t-distribution strategy is selected;
    a) According to Eqn. (10), use the Levy flight strategy to update the particle position;
    b) According to Eqn. (12), use adaptive *t*-distribution strategy to update the particle position;
6) Calculate the cost value of the current particle, compare it with the individual best cost value and the population best cost value, record $pbest^d$ and $pbest^d_i$. Go to Step 2.
7) Output $pbest^d$, end.

IV. Simulation verification and analysis

In the simulation experiment, the original PSO algorithm is denoted as PSO, the PSO algorithm with Tent mapping and dynamic parameter tuning factor is denoted as GPSO, and Adaptive Position Updating PSO algorithm improved in this paper is denoted as IPSO.

A. Standard Functions

In order to verify the performance of the adaptive position updating particle swarm algorithm, we adopt ten classical test functions for optimization testing, which are listed in Table 1.

TABLE I
Classical Benchmark Functions

Function	Function Name	Dimension	Mode	Best Value
F1	Sphere	30	Uni-modal	0
F2	Schwefel's Problem 2.22	30	Uni-modal	0
F3	Schwefel's Problem 1.2	30	Uni-modal	0
F4	Schwefel's Problem 2.21	30	Uni-modal	0
F5	Rosenbrock	30	Uni-modal	0
F6	Quartic Function	30	Uni-modal	0
F7	Rastrigin	30	Multi-modal	0
F8	Griewank	30	Multi-modal	0
F9	Shekel's Function 2	4	Multi-modal	-10.4029

B. Parameter setting

In the simulation experiments, the learning factors of PSO, GPSO and IPSO are set as c_1=2, c_2=2. For PSO, the inertia weights ω=0.8. For GPSO and IPSO, the starting inertia weights ω_{start}=0.9, and the final inertia weights ω_{end}=0.4. For the three algorithms, the size of populations is N=30 and the maximum number of iterations is T=500.

C. Comparison of experimental results and analyses

Each algorithm runs 30 times to optimize each of the four standard functions and the results of the runs are shown in Tables 2, 3, 4 and 5. The experimental results indicate that IPSO outperforms all other algorithms in the majority of test functions. Compared to the original PSO algorithm, IPSO demonstrates a significant and consistent improvement in overall performance. Both the figures and tables clearly show that IPSO achieves the best results in terms of mean and standard deviation across all algorithms, thereby demonstrating IPSO's superior adaptability and robustness when solving various types of problems. In functions F1-F9, IPSO exhibits rapid convergence speed and high optimization accuracy, is able to consistently escape local optima. Specifically, IPSO quickly identified the optimal solution in functions F1-F4 and F7-F8. For functions F5, F6, and F9, IPSO found superior solutions more efficiently than PSO and GPSO. However, in 30 runs of function F9, the optimal solution obtained by IPSO was slightly worse than that of GPSO, indicating that its precision in this particular case was marginally lower than that of GPSO. This finding is consistent with the well-known No Free Lunch Theorem, which asserts that no single algorithm can be universally optimal. Although IPSO performs impressively, its performance in certain specific scenarios may be surpassed by other algorithms.

Wilcoxon rank-sum test, a non-parametric method used to compare whether there is a significant difference between two independent samples, revealed a clear distinction between IPSO and both PSO and GPSO across all benchmark functions. This result confirms the effectiveness of the proposed improvements in the IPSO algorithm.

Friedman test is employed to identify statistically significant differences among algorithms. By comparing the performance of multiple algorithms across different datasets or test environments, the Friedman test effectively eliminates sample bias, providing a fairer comparison. Ranking the average Friedman values of the algorithms, IPSO ranked first with an average Friedman value of 1.0852, followed by GPSO at 1.9370, and PSO in third place. In summary, IPSO exhibits excellent performance on classical benchmark functions and shows a statistically significant difference when compared to the selected algorithms.

TABLE II
Performance Comparison of PSO, GPSO, and IPSO on Benchmark Functions

Function	Index	PSO	GPSO	IPSO
F1	ave	141.2970	2.2740	0.0000
	std	16.0267	0.9475	0.0000
F2	ave	170.8407	9.4671	0.0000
	std	189.7659	6.2203	0.0000
F3	ave	625.0044	365.8894	0.0000
	std	173.3739	916.3132	0.0000
F4	ave	4.8114	1.9603	0.0000
	std	0.4083	0.2491	0.0000
F5	ave	179041.9854	1027.4278	28.9692
	std	37647.5967	598.7805	0.061753
F6	ave	98.3636	6.3095	7.07E-05
	std	19.7443	6.965	7.10E-05
F7	ave	354.8062	209.3615	0.0000
	std	20.3392	30.9146	0.0000
F8	ave	1.0340	0.1139	0.0000
	std	0.0107	0.0541	0.0000
F9	ave	-4.8945	-9.3352	-10.3639
	std	1.6096	2.4978	0.0282

TABLE III
The Wilcoxon's p-value of each algorithm

Algorithm	PSO	GPSO
F1	1.2118E-12	1.2118E-12
F2	1.2118E-12	1.2118E-12
F3	1.2118E-12	1.2118E-12
F4	1.2118E-12	1.2118E-12
F5	3.0199E-11	3.0199E-11
F6	3.0199E-11	3.0199E-11
F7	1.2118E-12	1.2118E-12
F8	1.2118E-12	1.2118E-12
F9	3.0199E-11	3.9880E-04
(+/˜/-)	9/0/0	9/0/0

V. UAV Path Planning Simulation

A. PSO Algorithm in Path Planning

Optimization algorithms can be used for path planning because the path planning problem can be transformed into an optimization problem. In this optimization problem, the

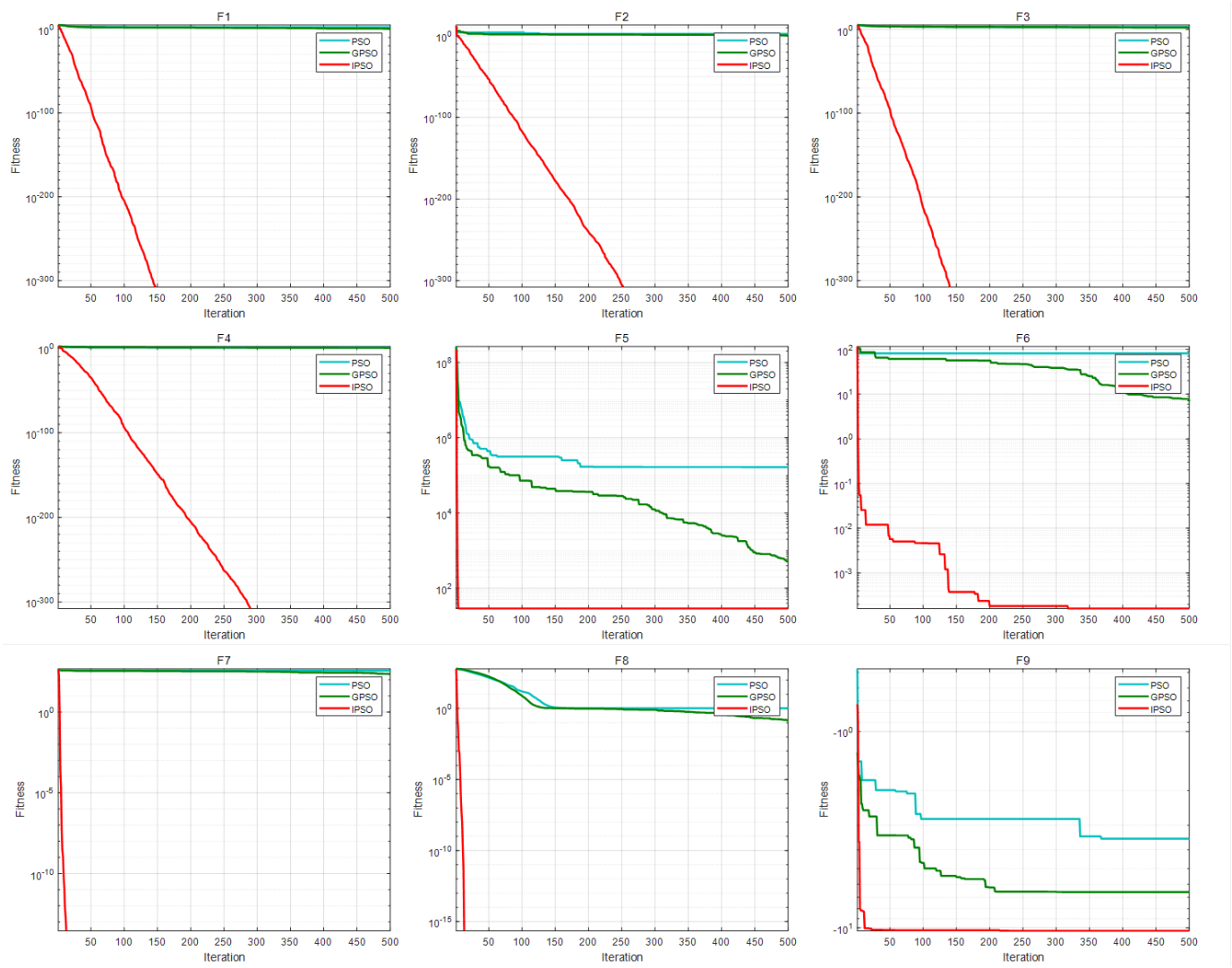

Fig. 5. Sphere function and its convergence curve.

TABLE IV
FRIEDMAN VALUE OF EACH ALGORITHM

Algorithm	PSO	GPSO	IPSO
F1	3.000	2.000	1.000
F2	3.000	2.000	1.000
F3	3.000	2.000	1.000
F4	3.000	2.000	1.000
F5	3.000	2.000	1.000
F6	3.000	2.000	1.000
F7	3.000	2.000	1.000
F8	3.000	2.000	1.000
F9	2.800	1.4333	1.7667
Average Value	2.9778	1.9370	1.0852
Rank	3	2	1

choice of path is represented as a cost function. The overall objective is to minimize this cost function. Particle Swarm Optimization algorithm is an optimization algorithm based on population intelligence and is particularly suitable for solving complex nonlinear optimization problems. Through iterative optimization, the particle swarm gradually converges and outputs the globally optimal path, i.e. the path with the lowest cost function value. As soon as the end of the delineation process, the outputted optimal path is the best flight route from the starting point to the end point taking into account the flight path and the distance from the obstacles.

B. Map creation

In UAV 3D path planning simulation, the creation of maps is crucial. In order to simulate the complex terrain in reality, we create a map of a mountainous area, as shown in Figure 8, where the peaks are modeled as obstacles. Within the range of the map, we defined the location, height and extent of the peaks to generate the terrain data. The peaks are distributed in different areas of the map, with heights ranging from 300 to 800 and the size of peaks varies. This setting is able to simulate

a real mountain environment and increase the difficulty of path planning. And thus ensure the diversity and complexity of the terrain to better test the performance of the algorithms.

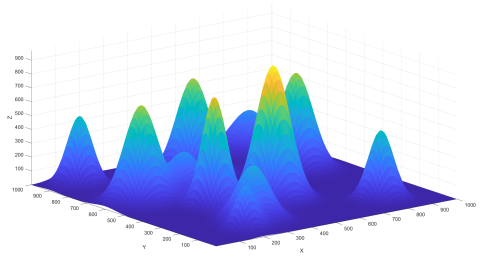

Fig. 6. Map model

C. Cost Function and Constraint Condition

1) Path Cost, P_c: The path cost, P_c, represents the distance between the start and the target point of the flight. In UAV path planning, P_c is a comprehensive metric used to evaluate the quality of a given flight path, reflecting the resources consumed by the UAV during its flight. P_ct encompasses not only the physical distance but also multiple factors such as time and energy. Generally, it can be assumed that Path Cost is proportional to both energy consumption and distance. As P_c increases, the distance traveled becomes longer, and energy consumption rises accordingly.If the total number of path points is n, the path cost P_c is defined as:

$$P_c = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{16}$$

where (x_i,y_i,z_i) are the 3D coordinates of the UAV's path point.

2) Terrain Cost, T_c: The terrain coxt, T_c, indicates the threat of collision with mountains during the flight of a UAV. The T_c is defined as:

$$T_c = \sum_{i=1}^{n} M_T \cdot \delta(z_i < Z_2(x_i, y_i)) \tag{17}$$

where M_T represents the threat coefficient of the terrain environment, is set as 1; $Z_2(x_i, y_i)$ is the terrain height at point(x_i,y_i);

If $z_i < Z_2(x_i, y_i)$, $\delta(z_i < Z_2(x_i, y_i))$ = 1. This indicates that the UAV collides with the mountain.

3) Boundary Cost, B_c: The B_c represents the boundary cost. It ensures that the UAV flies within the specified space during its mission. B_c is defined as:

$$\begin{aligned} B_c = \sum_{i=1}^{n} M_B \cdot [&\delta(x_i > x_{max}) + \delta(x_i < x_{min}) \\ &+ \delta(y_i > y_{max}) + \delta(y_i < y_{min}) \\ &+ \delta(z_i > z_{max}) + \delta(z_i < z_{min}) \,] \end{aligned} \tag{18}$$

where M_B represents the coefficient of the boundary constraint, is set as 1; x_{max},y_{max}, z_{max} are the upper bounds of the UAV's 3D flight coordinates; x_{min},y_{min}, z_{min} are the lower bounds of the UAV's 3D flight coordinates; and is the boundary constraint coefficient. Each dimension of the UAV path point (x_i, y_i, z_i) must be within the flight boundaries. Otherwise, a boundary cost will be counted.

4) Cost Function: UAV path planning needs to consider the combined total cost of the path cost, terrain cost, and boundary cost. Hence, our goal is to minimize an objective function, J, i.e., the total cost for UAV 3D path planning:

$$\min J = \min\{\text{Total Cost}\} = \min\{P_c + T_c + B_c\} \tag{19}$$

where P_c is the path cost; T_c is the terrain cost; B_c is the boundary cost.

In each iteration, the particle calculates the cost value of the current path based on the cost function and compares it with its historical best cost value and the global best cost value. If the costs value of the current path is better, the particle's historical best position and fitness value are updated, and the global best position and fitness value are updated when necessary. In this way, the particle swarm gradually approaches the optimal path and finally plans the most suitable path.

D. UAV Path Planning Simulation

In this paper, we set the range of the map as 1000 km × 1000 km × 1000 km, the starting point as (120, 20, 100) km, and the end point as (920, 720, 600) km. We also set the number of iterations as 150, the size of the population as 100. Meanwhile, we set M_T = 1 and M_B = 1 in the Cost Function. Each algorithm plans the path for 30 times respectively.

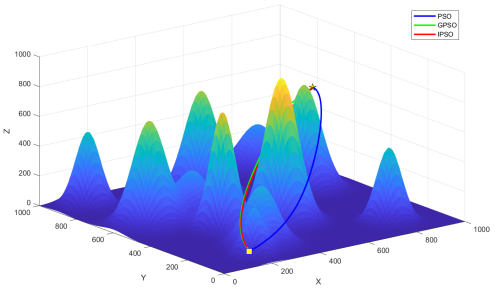

Fig. 7. Side view of the planned paths

As shown in Fig. 8 and 9, comparing the paths generated by PSO and GPSO, the path planned by IPSO is more reasonable with fewer inflection points and higher quality. High accurate path planning not only improves the safety of UAV flight, but also ensures the success of the mission. PSO and GPSO may be weak in this aspect, especially when facing complex terrain or dense obstacles.

As shown in Fig. 10, when the iteration reaches more than thirty times, both PSO and GPSO fall into the local optimal solution, and the generation value remains unchanged. However, IPSO can still judge the current position update

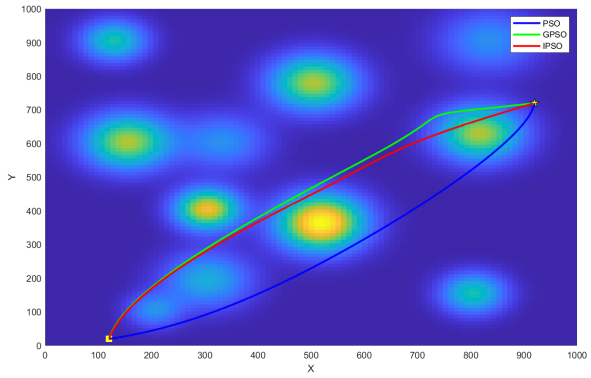

Fig. 8. Top view of the planned paths

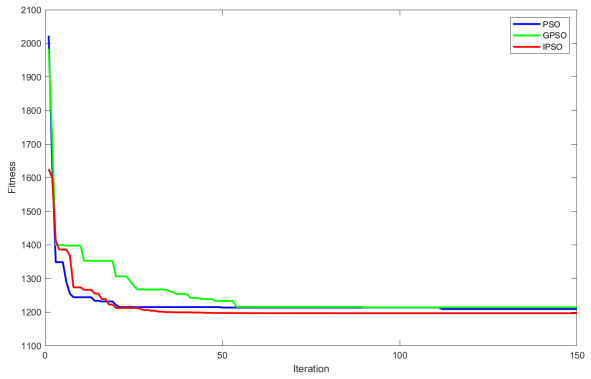

Fig. 9. Iteration curves of path cost function

strategy to be chosen based on the current fitness value. Thus it can be able to jump out of the local optimal, and keep approaching the global optimal. As shown in Table 6, the length of the path planned by IPSO is much lower than that of PSO and GPSO. IPSO is able to better find the optimal path from the start point to the end point, reducing the total distance of the UAV in flight, and thus reducing energy consumption and flight time. This advantage is important in practical applications, especially in scenarios where energy and time savings are required in UAV missions.

VI. Conclusion

In this paper, a UAV path planning method based on adaptive position updating PSO algorithm is proposed. By combining the Tent mapping Initialization, Levy flight and adaptive *t*-distribution strategies, together with the dynamic inertia weight adjustment, the search performance and global optimization ability of the particle swarm algorithm are significantly improved. Simulation experiments show that, upon comparing to performance of PSO and GPSO, the adaptive position updating PSO proposed in this paper(IPSO) demonstrates better performance in solving complex optimization problems and UAV path planning.

In the optimization of the standard test function, the adaptive position update particle swarm algorithm not only excels in convergence speed and search accuracy, but also significantly outperforms the traditional algorithm in global search capability and stability. Through the simulation in UAV 3D path planning, it is shown that IPSO is able to plan more reasonable and higher quality paths, which significantly reduces the total distance and energy consumption of the UAV flight, and improves the safety of the flight and the success rate of the mission.

Future research can further explore more hybrid strategies and dynamic adjustment mechanisms to further improve the adaptability and robustness of the algorithm. In addition, the method proposed in this paper can also be extended and applied to other complex optimization problems, such as robot navigation, intelligent traffic optimization and industrial process control. In conclusion, the application of IPSO in optimization problems and path planning is promising, with important research value and practical significance.

TABLE V
Simulation results of route planning

Algorithm	Average path length
PSO	1235.98
GPSO	1224.24
IPSO	1211.97

References

[1] J. Kennedy, R. Eberhart, "Particle Swarm Optimization," *Proc. Internat. Conf. Neural Networks (ICNN'95)*, Nov. 27 - Dec. 1, 1995, pp.1942-1948, Perth, Australia.

[2] J. Nan, "PSO-BP Neural Network Based Prediction Study for Nonlinear Problems," *Proc. 2023 IEEE Internat. Conf. Electrical, Automation and Computer Engineering (ICEACE)*, Changchun, China, 29-31 Dec., 2023, pp. 1507-1512.

[3] Y. Shi R. Eberhart, "A Modified Particle Swarm Optimizer," *Proc. 1998 IEEE Internat. Conf. Evolutionary Computation*, *IEEE World Congress Computational Intelligence*, Anchorage, AK, USA, 1998, pp. 69-73.

[4] J. Kennedy, R. Mendes, "Population Structure and Particle Swarm Performance," *Proc. 2002 Congress Evolutionary Computation (CEC'02)*, Vol. 2, pp. 1671-1676, Honolulu, HI, USA, 2002.

[5] Y. Gao, S. L. Xie, "Chaos Particle Swarm Optimization Algorithm," *J Computer Science*, Vol. 31, No. 8, 2004, pp. 13-15.

[6] F. Wang, Z. Wang, M. Lin, "Robot Path Planning based on Improved Particle Swarm Optimization," *2021 IEEE Internat. Conf. Big Data, Artificial Intelligence & Internet of Things Engineering (ICBAIE)*, Nanchang, Chain, 2021, pp. 887-891.

[7] D. Guo, Y. Zheng, J. Lu, S. Liang, T. Chen, J. Yang, C. Zhou, "Research on Enhanced Hybrid Whale Optimization Algorithm with Inverse Learning Strategy and Levy Flight Mechanism," *2023 4th Internat. Conf. Computer Engineering & Intelligent Control (ICCEIC)*, Guangzhou, China, 2023, pp. 283-289.

[8] G.-R. Cai, S.-L. Chen, S.-Z. Li, W.-Z. Guo, "Study on the Nonlinear Strategy of Inertia Weight in Particle Swarm Optimization," *2008 4th Internat. Conf. Natural Computation*, Vol. 7, pp. 683-687, Jinan, China.