drones

Article

Research on Particle Swarm Optimization-Based UAV Path Planning Technology in Urban Airspace

Qing Cheng [1,*,†], Zhengyuan Zhang [1,†], Yunfei Du [2] and Yandong Li [1]

[1] College of Air Traffic Management, Civil Aviation Flight University of China, Guanghan 618300, China; zyzhang123@cafuc.edu.cn (Z.Z.)
[2] School of Electronic Information Engineering, Beihang University, Beijing 100191, China
* Correspondence: deglechq@163.com
† These authors contributed equally to this work.

Abstract: Urban airspace, characterized by densely packed high-rise buildings, presents complex and dynamically changing environmental conditions. It brings potential risks to UAV flights, such as the risk of collision and accidental entry into no-fly zones. Currently, mainstream path planning algorithms, including the PSO algorithm, have issues such as a tendency to converge to local optimal solutions and poor stability. In this study, an improved particle swarm optimization algorithm (LGPSO) is proposed to address these problems. This algorithm redefines path planning as an optimization problem, constructing a cost function that incorporates safety requirements and operational constraints for UAVs. Stochastic inertia weights are added to balance the global and local search capabilities. In addition, asymmetric learning factors are introduced to direct the particles more precisely towards the optimal position. An enhanced Lévy flight strategy is used to improve the exploration ability, and a greedy algorithm evaluation strategy is designed to evaluate the path more quickly. The configuration space is efficiently searched using the corresponding particle positions and UAV parameters. The experiments, which involved mapping complex urban environments with 3D modeling tools, were carried out by simulations in MATLAB R2023b to assess their algorithmic performance. The results show that the LGPSO algorithm improves by 23% over the classical PSO algorithm and 18% over the GAPSO algorithm in the optimal path distance under guaranteed security. The LGPSO algorithm shows significant improvements in stability and route planning, providing an effective solution for UAV path planning in complex environments.

Keywords: urban environment; drones; obstacle avoidance; improved PSO algorithm; path planning

Citation: Cheng, Q.; Zhang, Z.; Du, Y.; Li, Y. Research on Particle Swarm Optimization-Based UAV Path Planning Technology in Urban Airspace. *Drones* **2024**, *8*, 701. https://doi.org/10.3390/drones8120701

Academic Editor: Chao Huang

Received: 21 October 2024
Revised: 18 November 2024
Accepted: 20 November 2024
Published: 22 November 2024

1. Introduction

Amid profound shifts in the global economy, traditional economic models are undergoing significant transformation and upgrades, and new technologies and industries are flourishing. Science and technology are moving forwards at an unprecedented speed, which is having extremely far-reaching impacts in various fields. Among them, a new generation of representative technologies, such as the Internet of Things (IoT) and artificial intelligence (AI), is becoming increasingly mature and widely used [1], which has led to a major breakthrough in the concept of smart logistics proposed by IBM in 2009 [2]. The rapid development of drone-related technologies is intricately linked with the improvement in smart logistics. The rapid development of drone-related technologies is linked to the improvement in smart logistics. In logistics, drones are an emerging distribution method which is changing industry patterns with its unique advantages [3].

At present, the advantages of drone logistics and distribution are mainly reflected in several aspects. Firstly, it is very efficient to use drones as a means of transportation. Compared with traditional ground logistics transport, drones are not restricted by road traffic conditions and can directly reach the destination [4] in less time. Drones rapidly navigate complex terrains to deliver emergency supplies and transport medicines and medical devices to remote medical institutions in a timely manner [5].

Secondly, using UAVs is extremely flexible. The route of a UAV is not affected by terrain and roads, and UAVs can easily traverse mountains, rivers, forests, and other complex terrains. In cities, UAVs can avoid congested traffic sections and reach their destination directly from the air [6,7]. Thirdly, UAVs are a cost-effective choice. The use of UAVs reduces the need to employ a large number of workers, as only a small number of technicians are needed to monitor and maintain the system. In addition, this reduces the maintenance costs of transport tools: compared with the huge logistics vehicles, the UAV structure is simple [8] and their maintenance costs are low. For small e-commerce enterprises, the use of drones for logistics and distribution will greatly reduce the operating costs and improve market competitiveness, enabling them to stand out against fierce market competition [9,10]. For example, in July 2024, Door Dash, a food delivery app company, partnered with Wing, a drone delivery service company, to launch a drone delivery service in Melbourne. The delivery drone is made of foam, weighs 5 kg, and can take off from its starting point (Eastland Shopping Centre), carry about 1 kg of cargo, and fly up to 7 km at 110 km/hour. Deliveries are usually completed within 30 min. In order to facilitate the safe and efficient receipt of takeaways, delivery companies require a small open space within 2 meters of the customer's address that is free of obstacles such as trees [11,12].

In UAV logistics and distribution, the role of path planning is crucial. Accurate and reasonable path planning can ensure the safe flight of drones in complex urban environments. After all, cities are densely populated with high-rise buildings, and good path planning can enable drones to effectively avoid collisions with obstacles such as buildings, and thus greatly reduce the probability of accidents. At the same time, efficient path planning can also significantly improve delivery efficiency. Through an in-depth analysis of the urban environment and the comprehensive consideration of distance, time and other factors, the optimal distribution path is planned so that the UAV can deliver the goods to its destination in the most effective way, thus shortening the distribution time [4,13]. In addition, reasonable path planning is also conducive to reducing operating costs, and choosing more direct flight routes can reduce energy consumption and lower the operating costs of UAV logistics in many ways.

However, research into UAV logistics and distribution faces many challenges. Firstly, the safety issue in the complex urban environment is extremely critical. Due to high buildings and dense populations [14] in the city, ensuring the safety of UAVs during flight and avoiding collisions with buildings undoubtedly constitute difficult problems that need to be addressed. Secondly, the accuracy and efficiency of path planning is extremely demanding. In complex urban environments, it is very important to plan the optimal delivery path to improve efficiency while the safety of the delivery process should be ensured. This brings many challenges to the design of the path planning algorithm [15].

In summary, this paper investigates an algorithm for UAV logistics and distribution path planning that is applicable to complex urban environments to command UAVs to achieve efficient transport and distribution on the basis of ensuring safety, which is both the core task and the key to the logistics of UAV path planning.

Existing studies of UAV path planning can be briefly classified into four categories: graph search method, potential field method, spatial sampling method, and meta-heuristic algorithms. The graph search method is often investigated mainly based on Dijkstra's algorithm and the A* algorithm [16]. Dijkstra's algorithm is a classical single-source shortest path algorithm. It starts from the start node and iteratively calculates the shortest distance to all other nodes. In each iteration, the node that is unvisited and closest to the start node is selected for expansion and the distances to its neighboring nodes are updated. This algorithm ensures that the shortest path is found, but the computational complexity is high. An improved Dijkstra's algorithm is proposed in [17] for the path planning problem of automated guided vehicles (AGVs) in smart warehousing. The A * algorithm is a heuristic graph search algorithm. It introduces a heuristic function based on Dijkstra's algorithm to estimate the distance from the current node to the target node. By combining the actual cost and the estimated cost, the A* algorithm is able to search in a more targeted way and improve the search efficiency. However, it will be limited by the capacity of the grid, which will increase exponentially due to the increase in dimensionality [18]. The artificial potential field method is used to guide the motion of objects by constructing virtual gravitational and repulsive fields. The target point generates a gravitational force to attract the object to come closer, and the obstacle generates a repulsive force for the object to avoid collision. The object moves under the action of the combined force. The method tends to fall into local minima, resulting in the inability to find the global optimal path [19,20]. In [21], the artificial potential field method is improved to make it more suitable for adaptive path planning for UAVs. The spatial sampling method mainly refers to the rapidly expanding random tree (RRT) algorithm, which explores paths by generating a tree structure through random sampling in space. From a starting point, it continuously expands to randomly sampled points until the target region is reached or other termination conditions are satisfied. It is suitable for high dimensional spaces and complex environments, but the paths may not be smooth or sufficiently optimized [22,23]. The RRT* algorithm is an improved version of the RRT algorithm. It rewires and optimizes the tree in the process of expanding it, making the generated paths closer to the optimal solution while improving the convergence speed [24]. Inspiration from nature often works well when dealing with path planning problems in the presence of complex dynamic environmental information. Meta-heuristic path planning algorithms based on nature-inspired methods have been increasingly used in the field of path planning due to their remarkable effectiveness in coping with the dynamic constraints of unmanned aerial vehicles (UAVs) and their excellent ability to search for the globally optimal solution in complex scenarios [25]. Various nature-inspired UAV path planning algorithms have been successfully developed, such as simulated annealing algorithm (SA) [26], ant colony algorithm (ACO) [27], genetic algorithm (GA) [28], differential evolutionary algorithm (DE) [29], and particle swarm algorithm (PSO) and its variants, which have been widely used in a variety of fields. Particle swarm algorithm (PSO) is a population intelligence-based optimization algorithm that solves various optimization problems by simulating the way a flock of birds or a school of fish behaves. The particle swarm algorithm as a population intelligence algorithm consists of three elements: simple individuals or information processing units; communication between individuals; and emergent behavior in the population, i.e., the complex behavior of the population is the result of patterns formed by individuals interacting over time, which cannot be inferred or predicted from the simple behavior of individuals. These elements enable each particle in the population to explore solutions based on its own experience as well as the experience of the population, rather than applying traditional evolutionary operators such as mutation and crossover. Therefore, the PSO algorithm can find the optimal solution faster and more consistently than other metaheuristic algorithms [30]. It is well known that PSO algorithms are less sensitive to changes in initial conditions and objective functions, and they are able to quickly adapt to a variety of environmental architectures by virtue of a small number of parameters including velocity weighting coefficients and learning factors. Considering these advantages, PSO and modified PSO are widely used in robotics, UAV navigation path planning, and other fields. Example include the particle swarm optimization-modified frequency bat hybrid algorithm (PSO-MFB) [31], which fuses the bat algorithm with PSO for robot path planning; AFRPSO [32], which is based on the artificial potential field method and the PSO algorithm for solving the robot path planning problem [32]; the improved particle swarm algorithm (IPSO) [33], which solves the optimal path problem of automated guided vehicles (AGVs); and the improved compression factor path subswarm algorithm (ICPPSA) [34], for solving the autonomous underwater vehicles (AUVs) path planning problem. These variants of PSOs are designed based on the same population structure. However, they provide their own insights into the encoding and decoding of particles as well as the search space, and the solutions obtained within the same environment and constraints yield different solutions. Hence, if these algorithms are applied to the field of UAV path planning, additional constraints need to be considered to improve the algorithm's navigational capability for UAV path planning aspects.

The objective of the research in this paper is to optimize the constraints of UAV operation in urban airspace by adjusting the objective function to improve the efficiency and safety of UAV operation in urban airspace. Due to the fast convergence speed and high environmental adaptability of the PSO algorithm, this paper introduces the PSO algorithm and improves it, so that the improved PSO algorithm, while guaranteeing the advantages of the previous algorithm, cannot only take into account the characteristics of the UAV itself, but also generate high-quality solutions when it is applied to UAV path planning. In order to evaluate the feasibility and superiority of the algorithm, a part of the complex urban environment is constructed using 3D modeling, and the performance of the algorithm is evaluated in comparison with other algorithms by increasing the level of complexity step by step, whilst the real feasibility of the algorithm is also improved due to the fact that the scenarios are taken from a real city.

The main work of this paper is summarized as follows: (1) The objective function is optimized for the urban environment and the characteristics of the UAV itself, which takes into account high-quality criteria and constraints related to the inclusion of paths, threats, climb and turn angles, and flight. (2) An improved PSO algorithm is proposed, which, by considering various parameters and constraints, enables the algorithm to find a high-quality global optimal solution in space while ensuring the convergence speed. (3) The performance of the original PSO, GAPSO, etc. in UAV path planning, with the improved PSO algorithm developed in this paper, is tested in a simulation scenario based on real city mapping.

The rest of this paper is structured as follows: Section 2 focuses on the constraints considered in formulating the path planning cost function and the formulation results; Section 3 describes how the improved PSO algorithm solves the path planning problem and its implementation steps; Section 4 performs the comparison and evaluation of the algorithms in the simulation scenario with real scenario mapping; and Section 5 concludes the study.

2. Formulation of the Problem

The purpose of this study is to design a logistics UAV path planning algorithm suitable for complex urban environments, the critical cost function in the UAV path planning algorithm is optimized and additional constraints suitable for logistics UAVs are added. The optimization of the cost function involved in this study contains three aspects, namely path optimization, collision risk optimization, and attitude optimization.

2.1. Path Optimisation

In order to ensure the efficient operation of UAVs, the planned paths need to be based on different application scenarios and follow specific guidelines to achieve optimal results. The UAV path planning problem refers to generating a cost-minimizing and collision-free path between the origin and destination points [4]. Figure 1 represents an example of a drone overcoming a threat object. And, for the logistics and distribution field, the shortest time used is the optimal choice under the guarantee of safety, so in order to achieve the optimal path, this paper adopts the shortest route design, because the UAV needs to be controlled through the ground control station, and thus the flight path P_{ij} is expressed using a list of path points across which the UAV needs to fly. Here, i represents the serial number of the path points, and i is used to distinguish the different path points; j represents the dimensional serial number of the coordinates, and the coordinates of each path point are expressed using $l_{ij} = (x_{ij}, y_{ij}, z_{ij})$; and n represents the total number of path points in the UAV flight path. And, the Euclidean distance between two path points is denoted using $\|\overrightarrow{l_{ij} l_{i,j+1}}\|$. The heaviest cost, namely the path cost function, can be expressed as:

$$C_1(P_i) = \sum_{j=1}^{n-1} \left\| \overrightarrow{l_{ij} l_{i,j+1}} \right\| \tag{1}$$

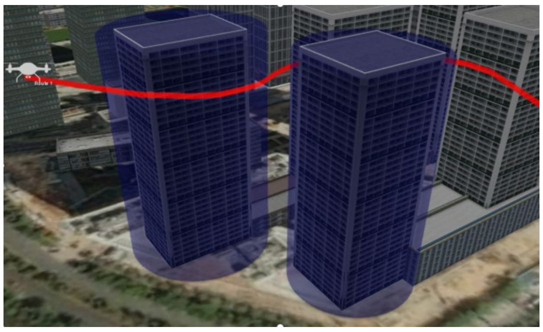


Figure 1. Example of a drone overcoming a threat.

2.2. Crash Risk Optimisation

When the UAV flies autonomously, the safe operation of the UAV must be ensured. Considering the complexity of threat modeling and the difficulty of obtaining real data, this study abstracts the threat environment, and the threat area is modeled as a cylinder, because the cylinder has the characteristics of being simple and intuitive, making it easy to compute and giving it strong applicability. The purpose of setting the threat region is to ensure the safety of UAVs in urban environments. As shown in Figure 2, U is the set of all existent threats, the center of the circle of the threat projection is G_U, the radius of the threat object is R_U, the perpendicular distance between the two neighboring path nodes, and the center of the circle is Q_U; taking into account the range of the UAV itself, let the diameter of the UAV be D, K denotes the danger area of the obstacle, the value varies with the specific signal situation of the UAV, and the distance of Q_U within the area of K is inversely proportional to the threat cost, the second heaviest cost, and the threat cost function can be expressed as:

$$C_2(P_i) = \sum_{j=1}^{n-1} \sum_{U=1}^{U} T_U\left(\overrightarrow{l_{ij} l_{i,j+1}}\right) \tag{2}$$

$$T_U\left(\overrightarrow{l_{ij} l_{i,j+1}}\right) = \begin{cases} \infty, & if\ D + R_U \geq Q_U \\ K + D + R_U - Q_U, & if\ \mathrm{K} + D + R_U \geq \mathrm{Q}_U > D + R_U \\ 0, & if\ \mathrm{K} + D + R_U < Q_U \end{cases} \tag{3}$$

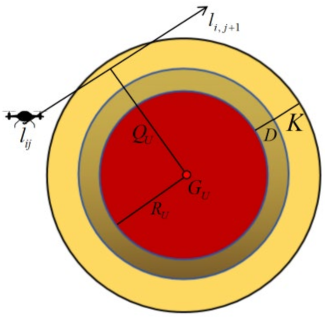


Figure 2. Threat cost.

2.3. Stance Optimization

The concept of the 'angle of climb' is crucial in the field of aviation, as it refers to the angle that an aircraft, drone, or other vehicle makes between its flight path and the horizontal plane during ascent. The 'smoothing cost' is used to evaluate the cost of smoothing the trajectory of a vehicle during flight. A large climb angle may mean that the vehicle can rapidly rise to the target altitude in a short period of time, but this often requires a larger power output, which may lead to a large amount of energy consumption, and may also increase the smoothing cost by increasing the mechanical wear and handling difficulty due to the drastic change in flight attitude. The climb angle ϕ in this study is shown in Figure 3, and the climb angle consists of paths $\overrightarrow{l_{i,j+1}l_{i,j+2}}$ and $\overrightarrow{l_{i,j+1}l''_{i,j+2}}$, which are represented by the climb paths with their projections on the XOY plane:

$$\phi = \arccos\left(\frac{\left\|\overrightarrow{l'_{i,j+1}l'_{i,j+2}}\right\|}{\left\|\overrightarrow{l_{i,j+1}l_{i,j+2}}\right\|}\right) \tag{4}$$

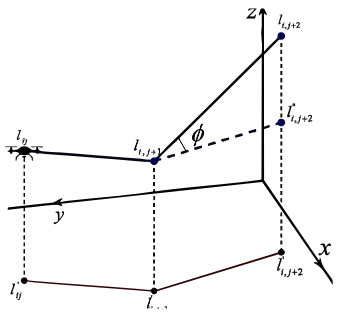


Figure 3. Climbing angle calculation.

The angle of turn is the same as the angle of climb, both of which directly affect the smoothing cost of UAV flight when varied, and in this study, the angle of turn, θ, is shown in Figure 4, and is represented by the projection of the UAV's flight trajectory in space onto the XOY surface:

$$\theta = \arcsin\frac{\left\|\overrightarrow{l'_{i,j}l'_{i,j+1}} \times \overrightarrow{l'_{i,j+1}l'_{i,j+2}}\right\|}{\left\|\overrightarrow{l'_{i,j}l'_{i,j+1}}\right\| \cdot \left\|\overrightarrow{l'_{i,j+1}l'_{i,j+2}}\right\|} \tag{5}$$

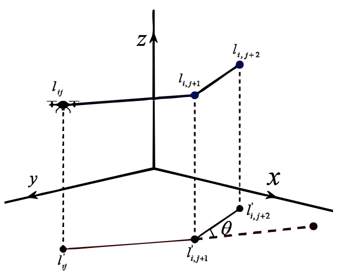


Figure 4. Turning angle calculation.

As a result, the smoothing cost relating the climb and turn angles is represented by the following equation, where λ is a penalty factor for the climb and turn angles.

$$C_3(P_i) = \lambda_1 \sum_{j=1}^{n-1} \left|\phi_{ij} - \phi_{i,j-1}\right| + \lambda_2 \sum_{j=1}^{n-2} \theta_{ij} \tag{6}$$

The total cost function is shown by the following, where Ω_m is the weight coefficient of each cost function, P_i is the decision variable containing the coordinates of the path points, and C_m is the set of the individual path cost functions, which can be fully determined and inputted into the UAV path planning by inputting and modifying the parameters of each cost.

$$C^{whole}(P_i) = \sum_{m=1}^{3} \Omega_m C_m(P_i) \tag{7}$$

3. LGPSO Algorithm

3.1. Random Inertia Weight

SHI and EBERHART proposed the introduction of the concept of inertia weights in the standard PSO algorithm at the 1998 IEEE International Conference on Evolutionary Computation. The design of its initial value affects the convergence speed and convergence quality of the particle swarm algorithm. This largely determines the amount of influence of historical factors on the current state [35].

In order to achieve a balance between the global search capability and local search capability in PSO algorithm, the reasonable adjustment of inertia weights is a proven method. In many improvement schemes of the PSO algorithm, the inertia weights usually adopt a linearly decreasing strategy [36]. Although this setting helps explore the search space extensively, it suffers from the problems of large algorithmic overhead as well as inefficient search.

In the later stages of the algorithm, the inertia weights are decreasing to accelerate convergence, but they tend to fall into local optima and cannot be optimized. The algorithm can jump out of the local optimum faster by setting it as a random number obeying a certain distribution and adjusting it to the characteristics of random variables, which is good for maintaining the diversity of the population and improving the performance of the global search. Due to randomness, the weight values at the beginning and the end of the particle movement may be variable in size. When the particle is close to the optimal particle, the random inertia weights have a small probability to accelerate the convergence, and if they are large and the result of the adaptation function is poor, they will be discarded and regenerated; when the particle is far from the optimal particle, the random inertia weights have a chance to take a large value in order to accelerate the convergence.

If the random inertia weight is a small value and the adaptation function value is poor, that small inertia weight is eliminated and the algorithm regenerates the weight value. With the linear decreasing strategy, it is difficult for the algorithm to converge to the optimal solution without finding a suitable inertia weight value at an early stage. While the random distribution generates weight values, the algorithm can obtain more ideal values in the late stage to avoid the stagnation of the value of the adaptation function, and the random inertia weights are shown as follows, where μ_{min} is the minimum value of the random inertia weights; μ_{max} is the maximum value of the random inertia weights; $rand()$ is a [0,1] uniformly distributed random number; and, in the third term, $randn()$ is a random number of a normal distribution; σ (standard deviation) is used to measure the degree of deviation between the random inertia weights, ω, and their mathematical expectation. The purpose of this term is to control the weighting error in the values taken, so that the weights ω are favorable to evolve in the direction of the desired weights.

$$\omega = \mu_{\min} + (\mu_{\max} - \mu_{\min}) \times rand() + \sigma \times randn() \tag{8}$$

3.2. Asymmetric Learning Factor

In the classical PSO algorithm, there exists a significant problem in that the lack of particle diversity in the later stages of the optimization search tends to prematurely converge to local extremes [37]. To solve this problem, the learning factor can be adjusted to prompt the particles to carry out a wide range of searches in the initial period of the search, with the expectation of obtaining high-quality particles with better diversity and avoiding the interference of local extremes as much as possible. Learning factors C_1 and C_2 determines the influence of the experience information of individual particles and other particles on the trajectory of the search, and fully reflects the information exchange between particles. If a large value of C_1 is set, the particles will search too much in the local region; on the contrary, a large value of C_2 will lead the particles to converge to the local optimum prematurely. Therefore, in the early stage of the algorithm, a larger value of C_1 and a smaller value of C_2 will allow the particles to disperse as much as possible throughout the search space, i.e., to emphasize the 'individual independent consciousness' while being less influenced by other particles in the population, i.e., the 'social consciousness part'. In other words, this emphasizes 'individual independent consciousness' and is less influenced by other particles in the population, i.e., the 'social consciousness part', so as to increase the diversity of particles in the population. As the number of iterations increases, C_1 decreases linearly while C_2 increases linearly, which strengthens the ability of the particles to converge to the global optimal point. The asymmetric learning factor is represented as follows, where C_1^s and C_1^e are the initial and termination values of the individual learning factor, and similarly, the other two are the initial and termination values of the social learning factor, where d represents the current number of iterations of the algorithm and M represents the maximum number of iterations.

$$C_1 = C_1^s + (C_1^e - C_1^s) \times \frac{d}{M} \tag{9}$$

$$C_2 = C_2^s + (C_2^e - C_2^s) \times \frac{d}{M} \tag{10}$$

3.3. Spherical Coordinate System

In this paper, we adopt a spherical coordinate system instead of the traditional three-dimensional Cartesian coordinates. In spherical coordinates, the position is represented by radial distance r, climbing angle ϕ, and turning angle θ. For a 3D Cartesian coordinate (x, y, z), the equivalent spherical coordinate is (r, ϕ, θ). This encoding approach has notable advantages in path planning, as it enables a more intuitive representation of the trajectory direction from one waypoint to the next.

Importantly, these spherical coordinates are relatively defined to each waypoint, rather than a single-fixed reference point. Each coordinate set specifies the direction change needed to reach a waypoint and subsequently re-orients towards the next. This characteristic is a major reason why spherical coordinates are more suitable for this application, as they allow precise control over turning and climbing angles. Such control helps energy consumption, especially when large directional changes are involved.

The transformation from spherical coordinates to Cartesian coordinates is given by the following equations:

$$x_{ij} = x_{i,j-1} + r_{ij} \sin\phi_{ij} \cos\theta_{ij} \tag{11}$$

$$y_{ij} = y_{i,j-1} + r_{ij} \sin\phi_{ij} \sin\theta_{ij} \tag{12}$$

$$z_{ij} = z_{i,j-1} + r_{ij} \cos\phi_{ij} \tag{13}$$

where $x_{i,j-1}, y_{i,j-1}, z_{i,j-1}$ denote the Cartesian coordinates of the previous waypoint, and (r, ϕ, θ) represent the radial distance, climbing angle, and turning angle for the current waypoint, respectively. For scenarios in which the UAV maintains a constant speed, the radial distance r can be fixed, allowing the optimization to focus solely on adjusting the climbing angle ϕ and turning angle θ. This simplification effectively reduces the solution space, making it easier to generate feasible paths.

3.4. Improved Levy Flight Strategy Combined with Greedy Search

Levy flight, as a distinctive search strategy, has been shown to offer substantial benefits when integrated into particle swarm optimization (PSO) algorithms. It operates by alternately executing small and large step intervals, significantly enhancing the exploration ability and range of particle movements [38]. In traditional PSO, particle motion is often constrained, leading to a tendency to converge prematurely on local optima. By incorporating Levy flight, these limitations are overcome. The large step size enables particles to traverse larger areas quickly, expanding the search space, while the smaller step size allows for fine-tuning within already explored regions, thus improving both search efficiency and comprehensiveness. This unique dual-step characteristic is particularly beneficial in complex optimization problems with numerous local optima and dispersed solution distributions.

Levy flight also plays a key role in maintaining particle diversity by introducing irregular, random jumps. This mechanism fosters a more diversified exploration of the search space, which is essential for avoiding premature convergence to suboptimal solutions and for maintaining the potential to locate the global optimum. Furthermore, the extended search range enabled by Levy flight contributes to better convergence accuracy and speed, increasing the likelihood of discovering high-quality solutions more quickly and precisely.

Mathematically, Levy flight is typically represented as follows:

$$L = \frac{\alpha u}{|v|^{\frac{1}{\beta}}} \tag{14}$$

where β denotes parameters controlling the Levy flight characteristics, α denotes the fixed-step parameters, u and v denotes variables obeying a normal distribution. $u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_v^2)$, σ_u, σ_v can be expressed as follows:

$$\begin{cases} \sigma_u = \left(\dfrac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma\left[(1+\beta)/2\right]\beta 2^{((\beta-1)/2)}}\right)^{\frac{1}{\beta}} \\ \sigma_v = 1 \end{cases} \tag{15}$$

However, while Levy flight allows particles to escape the local optima, the fixed-step parameter α can limit search precision. Larger values of α promote global exploration but may reduce search accuracy, whereas smaller values increase accuracy but slow down the search process. To address this, an adaptive step factor is introduced, dynamically adjusting the step size and direction based on the current search state and historical information. The Gaussian kernel function is used to replace the fixed-step parameter, enabling dynamic step adjustment as the algorithm progresses. This adaptation ensures both global exploration in the early stages and local refinement near the optimal solution. For instance, if the algorithm becomes trapped in a local optimum, the Gaussian kernel can increase the step length, facilitating escape, while decreasing it near the global optimum to refine the search.

The Gaussian kernel function is defined as:

$$\varepsilon = e^{-\frac{(d-\frac{M}{2})^2}{2*\delta^2}} \tag{16}$$

Additionally, the improved Levy flight mechanism is represented as:

$$L = \frac{\varepsilon u}{|v|^{\frac{1}{\beta}}} \tag{17}$$

One notable drawback of the Levy flight mechanism is its inherent randomness, which does not guarantee that the particle's updated position will be an improvement over the previous one. Due to this uncertainty, the particle may end up in a worse position, potentially degrading the algorithm's overall performance. To mitigate this issue and avoid meaningless position updates, a greedy evaluation strategy is introduced to assess whether the updated particle positions should be accepted. This strategy involves a strict comparison of the fitness values of the updated and original positions, as defined by a specific objective function [39]. Specifically, the update is only accepted if the objective function value of the new position is better (i.e., smaller) than that of the original position, ensuring that the update results in a tangible improvement. This approach helps prevent unnecessary or detrimental position updates, leading to more efficient optimization by reducing redundant calculations and minimizing the risk of erroneous decisions. The implementation of the greedy evaluation strategy is as follows, where $\alpha_g^{new}(t)$ is the position of the particles after the greedy evaluation and $f(\alpha(t))$ represents the particle fitness function.

$$\alpha_g^{new}(t) = \begin{cases} \alpha_g^{new}, & \text{if } f(\alpha_g^{new}(t)) < f(\alpha_l^{old}(t)) \\ \alpha_l^{old}, & \text{if } f(\alpha_g^{new}(t)) > f(\alpha_l^{old}(t)) \end{cases} \tag{18}$$

In this paper, the improved particle swarm optimization algorithm (LGPSO) is applied to the optimal path planning problem for logistics UAVs. The flowchart and pseudocode for LGPSO are illustrated in Figure 5 and Algorithm 1. The operational procedure of the improved Lévy flight particle swarm optimization (LGPSO) is outlined as follows: Initially, the particle swarm is initialized, and in each iteration, the particles' positions and velocities are updated. Subsequently, the particle costs are evaluated, and the optimal solution is updated accordingly. After the particles undergo an update through the improved Lévy flight mechanism, a greedy search evaluation is conducted to determine whether the updated positions should be accepted. The optimal solution is then output once the maximum number of iterations is reached. The integration of Lévy flight and greedy search enhances the algorithm's search capabilities and convergence speed, thereby enabling more efficient pathfinding.

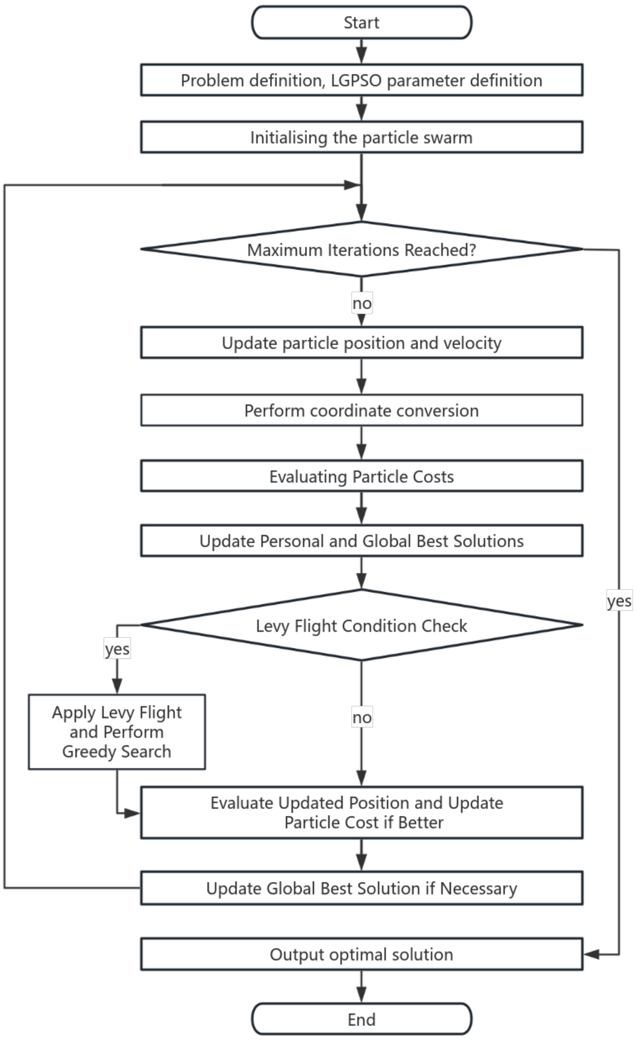

Start
Problem definition, LGPSO parameter definition
Initialising the particle swarm
Maximum Iterations Reached?
no
Update particle position and velocity
Perform coordinate conversion
Evaluating Particle Costs
Update Personal and Global Best Solutions
Levy Flight Condition Check
yes
Apply Levy Flight and Perform Greedy Search
no
Evaluate Updated Position and Update Particle Cost if Better
Update Global Best Solution if Necessary
yes
Output optimal solution
End

Figure 5. LGPSO flowchart.

Algorithm 1. The LGPSO algorithm's pseudocode

Input: Load the scene map and initial path planning data. Set swarm parameters: maximum iterations M, population size $npop$, velocity limits v_{min} and v_{max}, inertia weight ω, learning coefficients C_1 (individual) and C_2 (social), climb angle limit ϕ_{max}, turn angle limit θ_{max}, height limits h_{min} and h_{max}, and number of path nodes N.

```
1.  Particle swarm initialization: create an empty particle structure
2.  Initialize the global optimal solution
3.  Set the maximum number of iterations $M$
4.  While d ≤ $M$ do
5.      Update the optimal cost value $Best\ Cost$
6.      for each particle $i = 1\ to\ npop$
7.          Calculate random inertia weight $\omega$ and learning coefficients $C_1$ and $C_2$
8.          Update rate $v$ and location
9.          Map spherical coordinates of flight path to Cartesian coordinates
10.         Evaluate particle cost $Best\ Cost$
```

```
11.         if $Cost < PBest\ Cost$
12.             Update the personal best solution
13.         end if
14.         if $Cost < GBest\ Cost$
15.             Update the global optimal solution
16.         end if
17.         if Levy's flight conditions are satisfied (stagnation threshold reached)
18.             Conduct Levi's flight to update particles
19.             Perform a greedy search to evaluate updated particles
20.             if $greedy\ cost < New\ cost$
21.                 Update cost
22.             end if
23.             if $New\ cost < PBest\ Cost$
24.                 Update personal best
25.             end if
26.             if $PBest\ Cost < GBest\ Cost$
27.                 Update the global optimal solution
28.             end if
29.         end if
30.     end for
31. end while
32. Display iteration information and output plotting results
```

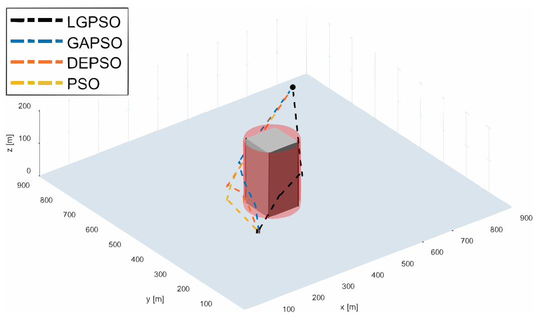
4. Evaluation of Algorithms for Simulation Scenarios

In order to comprehensively evaluate the performance of the LGPSO algorithm, this paper selects the scenario threat level, optimal fitness, optimal distance, and algorithm stability as the evaluation metrics. The scenario threat level is an indicator that can intuitively reflect the performance of the algorithm in dealing with complex scenarios, because there are various potential threats in different scenarios, which is an important consideration for the effectiveness of the algorithm; optimal fitness is a key indicator to measure the proximity of the solution found by the algorithm to the ideal optimal solution, and a high fitness implies that the algorithm obtains a better result; the optimal distance is also an important element of the evaluation, which reflects that the optimal distance is also an important evaluation element, which reflects the relationship between the path determined by the algorithm and the ideal shortest path, and helps judge the strengths and weaknesses of the algorithm in path planning; moreover, the stability of the algorithm ensures that the algorithm is able to consistently and stably output reliable results under a variety of different inputs and complex environments.

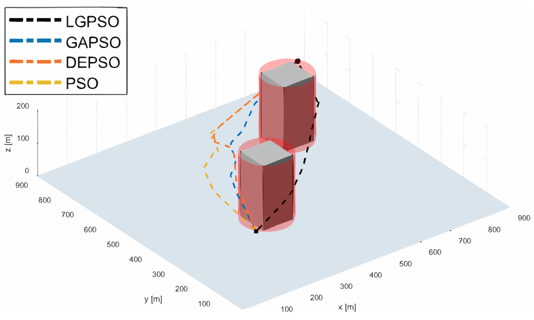
4.1. Hypothetical Threat Algorithm Evaluation

In order to comprehensively evaluate the performance of the LGPSO algorithm, this paper makes an in-depth comparison with the PSO and variant PSO algorithms widely used in related fields, including the genetic particle swarm algorithm (GAPSO) [40], the differential evolutionary particle swarm algorithm (DEPSO) [41], and the primitive particle swarm algorithm (PSO).

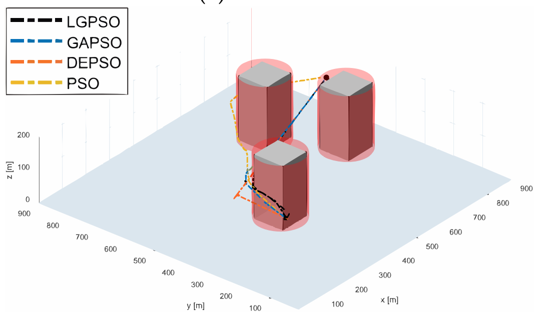
In the preliminary testing phase, four hypothetical threat scenarios with step-by-step escalation are employed to precisely control the variables by strictly regulating the cost function and the relevant parameters of each algorithm. Subsequently, a detailed comparison is made of the top-view and path adaptation generated by each algorithm, as illustrated in Figures 6 and 7. This comparison enables a clearer understanding of the performance differences across algorithms in specific scenarios, providing a solid foundation for further analysis and the evaluation of their effectiveness. It is evident from Table 1 that, in simple scenarios, all algorithms generate feasible paths that satisfy requirements in terms of path length, threat, turn angle, climb and dive angle, and altitude. Although all algorithms achieve normal convergence, the LGPSO algorithm achieves near-optimal solutions, in contrast to the PSO, GAPSO, and DEPSO algorithms which only converge to relatively good solutions. There is a certain difference in their respective fitness values, and this difference is not negligible. In fact, this difference fully reflects the differences and characteristics of different algorithms when dealing with the same problem.



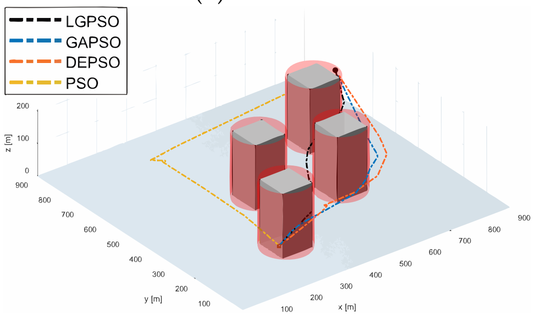
(**a**) Scenario 1



(**b**) Scenario 2

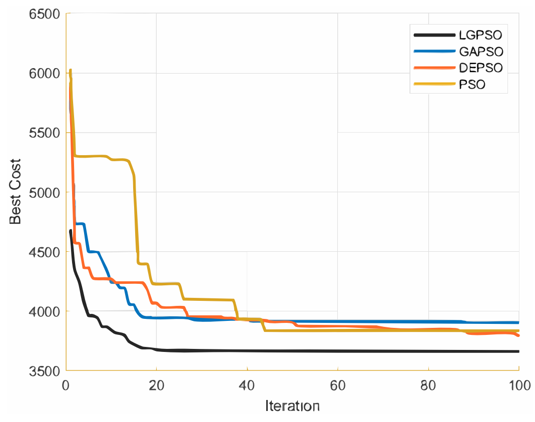


(**c**) Scenario 3

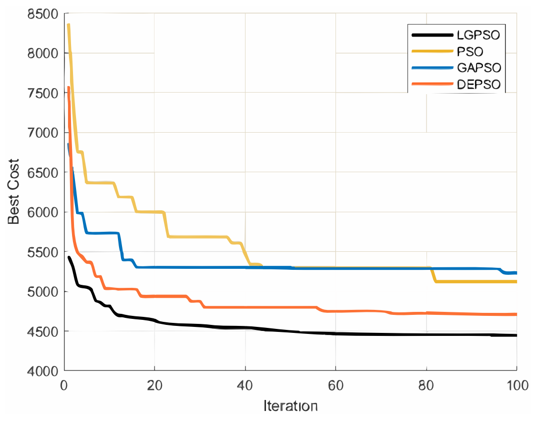


(**d**) Scenario 4

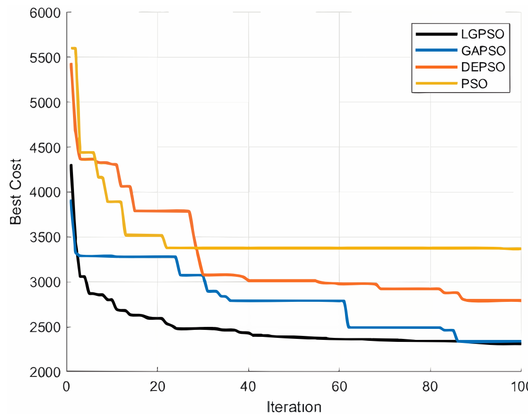
Figure 6. Top view of the path of the PSO and its variants in four virtual scenarios.



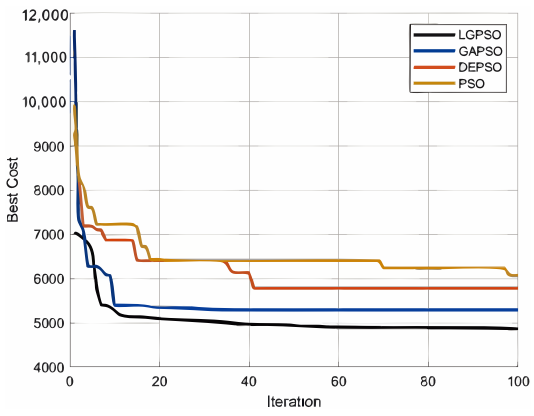
(**a**) Scenario 1



(**b**) Scenario 2



(**c**) Scenario 3



(**d**) Scenario 4

Figure 7. Optimal fitness values for iterations of PSO and its variants of the algorithm.

Table 1. Optimal fitness of each algorithm.

	LGPSO	**GAPSO**	**DEPSO**	**PSO**
Scenario	Best cost	Best cost	Best cost	Best cost
1	3659	3873	3659	3774
2	4476	5324	4736	5208
3	2310	2337	2792	3368
4	5040	5375	5751	6136

In addition, for the threat level is quantified, set the threat level as Q, and the number of threatening bodies as ϑ. τ is the scale factor to make the results more intuitive, k_1 and k_2 represent the threatening body influence coefficient and the path influence coefficient, respectively, and taking into account the security, $k_1 > k_2$ and $k_1 + k_2 = 1$, R_{total} is the average danger radius of all the threatening bodies, which is why this paper builds a simulation environment threatening body radius that does not have more than 100 individuals, wherein it is set that $R_{max} = 100$. Then, the threat level is expressed in Table 2:

$$\begin{cases} Q = \tau(k_1\vartheta \times \dfrac{R_\iota}{R_{\max}} + k_2 n) \\ R_{total} = \dfrac{\sum_{\iota=1}^{\vartheta} R_\iota}{\vartheta} \end{cases} \tag{19}$$

Table 2. Threat level for scenarios 1–4.

Scenario	**Threat Level**
1	10.23%
2	17.46%
3	23.19%
4	28.52%

4.2. Simulation Scenario and Parameter Setting

In order to further evaluate the performance of the algorithm, this paper uses the real scene for mapping, and the scene used for simulation is based on a small portion of the real city as the basis for 3D modeling operations, the effect presented by the simulated scene and the evaluation parameters of the algorithm are shown in Figure 8 and Table 3.

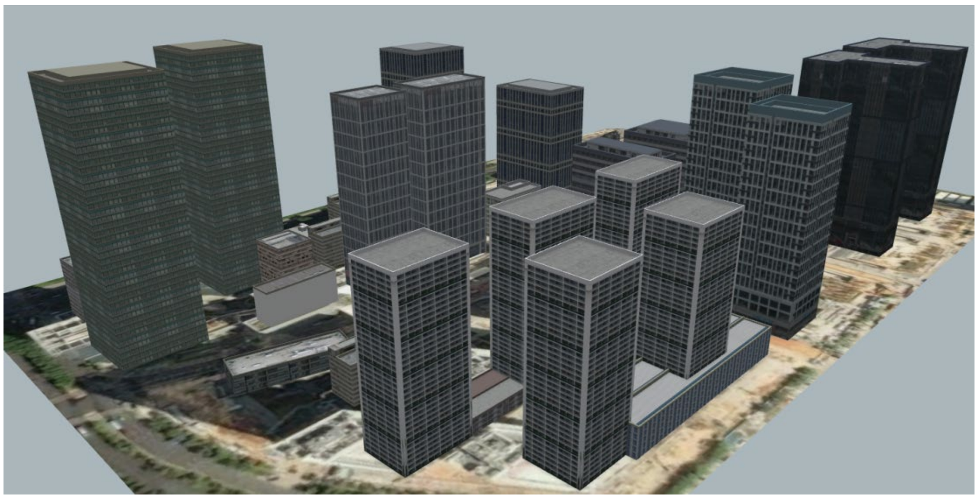


Figure 8. Three-dimensional modeling of real urban environments.

Table 3. System parameters used for simulation experiments.

Parameter	Parameter Symbol	Value
Particle swarm population size	*Npop*	500
Maximum number of iterations	M	200
Minimum flight altitude	h_{min}	30
Maximum flight altitude	h_{max}	200
Maximum turning angle	θ_{max}	60
Maximum climbing angle	φ_{max}	45
Number of waypoints	N	20/30
Speed control factor	α	0.5
Diameter of the drone	D	1

4.3. Comparison of Algorithms for Simulation Scenarios

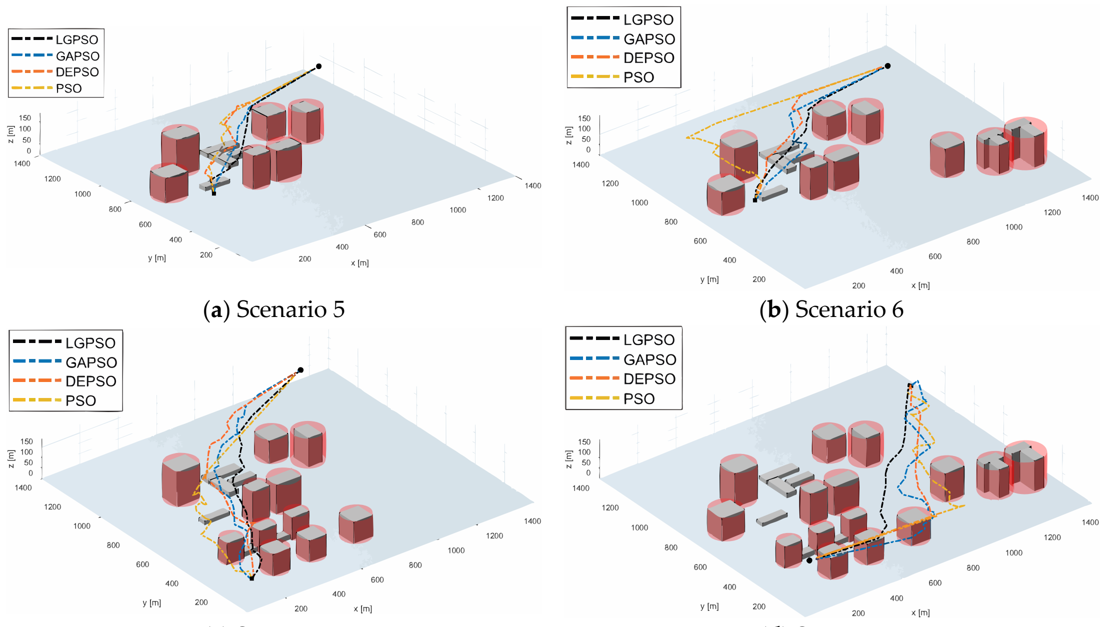
The simulation scenarios are comprehensively evaluated in a progressively open-ended manner, adopting a phased approach to algorithm testing. Table 4 represents the threats in the real-world scenario mapping environment. During this process, the number of waypoints in each stage increases incrementally. Additionally, a 3D roadmap generated by the LGPSO algorithm in the simulation scenario is provided, along with a 3D view and path adaptation, as shown in Figures 9 and 10.

Upon closer inspection, it is evident that LGPSO excels in path selection, outperforming both PSO and GAPSO. While aiming for the shortest distance, LGPSO effectively ensures the safety and feasibility of the paths in a practical manner, successfully avoiding the pitfalls of local optimality. Although GAPSO enhances the quality of feasible solutions through crossover and genetic operations, a noticeable performance gap remains between LGPSO and GAPSO. LGPSO stands out among various algorithms due to its unique advantages, offering a more effective approach to solving related problems. Its exceptional performance in path planning is not only reflected in its ability to rapidly find the shortest path but also in its strong focus on safety, feasibility, and its successful avoidance of local optimal traps. In contrast, although GAPSO makes strides in improving the quality of feasible solutions, it still requires further refinement to reach the overall performance level of LGPSO.

Figure 9 clearly shows the 3D views of the paths obtained by LGPSO in the four simulation scenarios. It can be clearly seen that the paths plotted by the LGPSO algorithm are extremely smooth while maintaining an appropriate flight altitude with respect to the buildings. Moreover, the climb and turn processes appear to be very gentle. Such a characteristic is of great significance, which can effectively avoid unnecessary power consumption when the UAV performs drastic climbs and turns. In this way, the LGPSO algorithm not only ensures the safety and stability of the UAV flight, but also improves the endurance of the UAV to a certain extent, which provides a strong guarantee for the efficient operation of the UAV in practical applications.

Table 4. Threat level for scenarios 5–8.

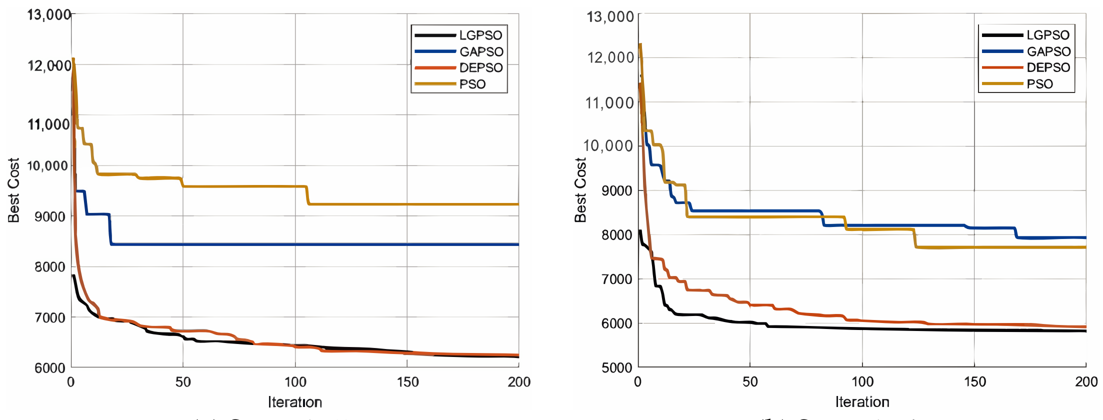
Scenario	Threat Level
5	45.13%
6	53.50%
7	73.57%
8	81.93%



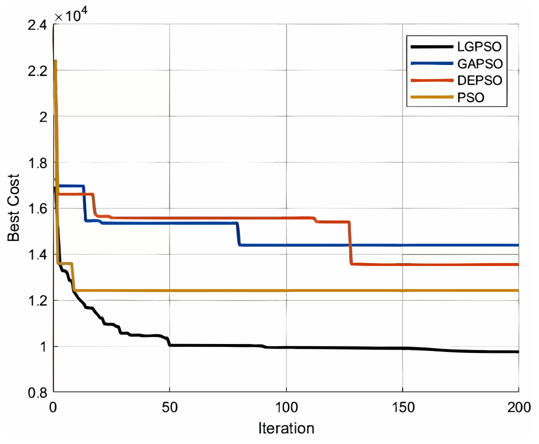
(**a**) Scenario 5 (**b**) Scenario 6

(**c**) Scenario 7 (**d**) Scenario 8

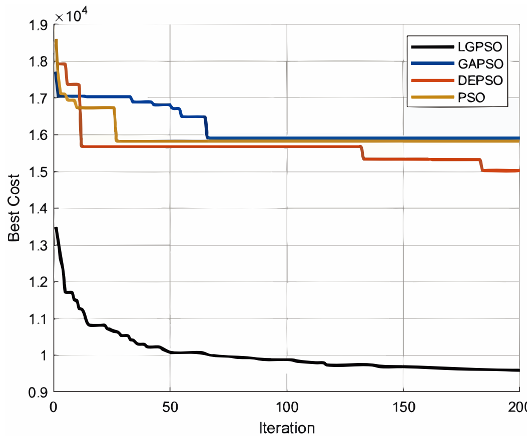
Figure 9. Top view of the path of the PSO and its variants in four step-up real city scenarios.



(**a**) Scenario 5 (**b**) Scenario 6



(**c**) Scenario 7



(**d**) Scenario 8

Figure 10. PSO and its variant algorithms' optimal fitness values in urban environments.

The stability of the LGPSO algorithm and the variant PSO algorithm against which it is compared can be explored in depth by conducting multiple trials of the LGPSO algorithm on complex paths in the final simulation scenario. The best fitness, standard deviation (SD), standard deviation fluctuation coefficient (SDFC), and relative standard deviation rate of change (RSDCR) data in Table 5 provide an important basis for evaluating the performance of the algorithms. The average fitness reflects the overall performance level of the algorithm over multiple trials, and a lower value means that it is easier to find a better solution in that simulation scenario. At the same time, observing the change in average fitness under a different number of trials can determine the stability of the algorithm, and smaller fluctuations indicate that the algorithm's performance is stable in repeated trials and is not significantly affected by random factors. The standard deviation measures the degree of dispersion of the algorithm in the results of multiple trials; smaller values indicate that the algorithm results are concentrated and have good stability; larger values indicate that the results are dispersed and may be more affected by random factors and less stable. In addition, the SDFC reflects the fluctuation of the standard deviation of fitness in different iteration cycles, where the closer the value is to zero, the more stable the standard deviation changes are in the iteration, and the smoother the change is in the dispersion of fitness data produced by the algorithm. SD_i indicates the standard deviation of the current fitness, and $\overline{SD}$ indicates the average standard deviation of the algorithm.

$$\begin{cases} SDFC = \sqrt{\dfrac{\dfrac{1}{M}\sum\limits_{i=1}^{M}\left(SD - \overline{SD}\right)^2}{\overline{SD}}} \\ \overline{SD} = \dfrac{1}{M}\sum\limits_{i=1}^{M} SD_i \end{cases} \tag{20}$$

RSDCR measures the change in the standard deviation of the fitness of the algorithm in different stages. The standard deviation of the first 50 iteration stages is selected as the initial stage. The last 50 are selected as the current stage. The average of the standard deviation of the two stages is calculated. The negative value indicates that the standard deviation in the fitness is decreasing from the initial stage to the current stage. The absolute value of the standard deviation reflects the degree of change, and the closer it is to zero, then the algorithm's stability is better in the two stages. $SD_{current}$ and $SD_{initial}$ represent the average standard deviation of the first 50 and the last 50 iteration stages, respectively, since the average standard deviation has 50 iteration stages.

$$RSDCR = \left| \frac{SD_{current} - SD_{initial}}{SD_{initial}} \right| \tag{21}$$

As shown in Table 5, the results demonstrate that the LGPSO algorithm exhibits a more stable performance in multiple path tests in the final simulation scenario. The standard deviation of LGPSO fluctuates within a narrow range, indicating consistent operation throughout the trials. In contrast, PSO is more prone to local optima, as evidenced by its lower standard deviation values, which reflect less variability but also lower robustness. The analysis of the standard deviation fluctuation coefficient (SDFC) and the relative standard deviation change rate (RSDCR) further supports the advantages of the LGPSO algorithm. Both SDFC and RSDCR values for LGPSO are moderate, approaching zero, indicating high stability across different iteration cycles and phases. These values emphasize that LGPSO maintains performance consistency without large fluctuations or random variations, unlike other PSO variants. Overall, the results demonstrate that LGPSO significantly improves upon traditional particle swarm optimization methods, offering a more reliable and stable approach to path planning. This combination of stability and improved pathfinding makes LGPSO a promising tool for future applications and optimizations.

Table 5. LGPSO algorithm's multiple-verified optimal fitness, SD, SDFC, and RSDCR.

	Times	SD	SDFC	RSDCR
LGPSO	1	599.93	0.1045	0.0511
	2	575.43	0.1963	0.2945
	3	635.69	0.1493	0.1156
Statistical mean		**603.68**	**0.1500**	**0.1537**
GAPSO	1	633.04	0.2306	0.2390
	2	641.01	0.3056	0.8524
	3	751.29	0.3571	1.4697
Statistical mean		**675.11**	**0.2978**	**0.8537**
DEPSO	1	937.33	0.3776	1.2589
	2	530.00	0.2868	0.3876
	3	888.39	0.1666	0.3636
Statistical mean		**785.24**	**0.2770**	**0.6700**
PSO	1	208.1752	0.5427	0.6244
	2	82.1869	0.5905	0.6580
	3	318.7434	0.2923	0.4693
Statistical mean		**203.04**	**0.4752**	**0.5839**

In flight path planning, the frequency and magnitude of directional changes critically affect path smoothness and efficiency. To move beyond subjective assessments such as "visually smooth", this study introduces a quantitative framework for scientifically evaluating LGPSO's performance. Specifically, the standard deviation and maximum values of turning angles and climb angles are calculated to objectively measure the degree of directional changes. The normalized smoothness score and efficiency score provide a robust basis for evaluating path planning performance, and a final score formula F_{final} integrates these metrics into a comprehensive assessment. As shown in Table 6. Smoothing efficiency and the parameters of normalization, the relevant parameters are presented, which play an important role in the calculation and evaluation of the path planning performance metrics.

Table 7 presents a comparative analysis of LGPSO, GAPSO, DEPSO, and PSO over nine experimental runs. LGPSO achieves a superior average score of 0.9526, significantly outperforming GAPSO (0.8155), DEPSO (0.8212), and PSO (0.7727). This demonstrates

LGPSO's effectiveness in reducing directional changes while maintaining high efficiency, resulting in smoother turning and climbing angles compared to other algorithms. These findings, validated by smoothness and efficiency metrics, provide an objective evaluation of LGPSO's performance. To enhance the practical relevance of the results, high-threat scenarios were specifically selected for evaluation. In these environments, UAVs face more frequent flight attitude changes, presenting a challenging test for algorithms to balance smoothness and efficiency. LGPSO's superior performance under these conditions underscores its robustness and adaptability. By adopting this quantitative framework and focusing on high-threat scenarios, this study establishes a systematic and objective approach for evaluating flight path planning algorithms. LGPSO demonstrates strong capabilities in terms of optimizing path smoothness and efficiency while minimizing directional changes, offering both theoretical insights and practical guidance for complex flight environments. This framework also serves as a reference for future research aiming to improve the path planning performance.

Table 6. Smoothing efficiency and the parameters of normalization.

$S_{smoothness}$	Smoothness score
$E_{efficiency}$	Efficiency scores
F_{final}	Final score
ϕ_{std}	Standard deviation of the climb angle
θ_{std}	Standard deviation of the turning angle
$\kappa_1\ \kappa_2$	Normalized weighting factors
Normalized Value	Normalization
$o_{Threshold}$	Normalization anchor

$$Normalized\,Value = \frac{\log(1+ActualValue)}{\log(1+o_{Threshold})} \tag{22}$$

$$S_{smoothness} = 1 - \frac{Normal(\phi_{std}+\phi_{\max}+\theta_{std}+\theta_{\max})}{4} \tag{23}$$

$$E_{efficiency} = \frac{2}{Normal(\phi_{\max}+\theta_{\max})} \tag{24}$$

$$F_{\text{final}} = \sqrt{\kappa_1 \cdot S_{smoothness}} + \kappa_2 \cdot \ln(1+E_{efficiency}) \tag{25}$$

Table 7. Algorithm scoring comparison.

	LGPSO	**GAPSO**	**DEPSO**	**PSO**
Times	F_{final}	F_{final}	F_{final}	F_{final}
1	0.9468	0.7472	0.8175	0.8174
2	0.9823	0.8359	0.8869	0.8122
3	0.9547	0.8345	0.8332	0.7556
4	0.9753	0.8356	0.8339	0.7609
5	0.9250	0.8039	0.7536	0.7094
6	0.9449	0.8048	0.8193	0.7751
7	0.9506	0.8604	0.8219	0.7801
8	0.9300	0.8194	0.7875	0.7309

9	0.9637	0.7980	0.8373	0.8131
Average	0.9526	0.8155	0.8212	0.7727

In the field of average path planning, significant differences exist in the performance of various algorithms. LGPSO stands out as a benchmark, demonstrating superior path planning efficiency when compared to the other methods. As shown in Table 8, LGPSO achieves the most efficient path planning, with its average path length serving as the baseline for comparison. When compared to LGPSO, GAPSO exhibits a significantly longer average path length, at 118.12% of LGPSO's, indicating that GAPSO is less effective in optimizing path lengths. DEPSO falls in the middle, with its average path length at 114.87% of LGPSO's, still reflecting an inefficiency compared to LGPSO. PSO, on the other hand, shows the least favorable performance, with its average path length reaching 123.33% of LGPSO's, highlighting a considerable gap in the path planning efficiency.

In conclusion, LGPSO clearly outperforms other algorithms in terms of average path planning efficiency. Its ability to generate shorter and more efficient paths provides a valuable reference point for future developments in this area. While other algorithms like GAPSO, DEPSO, and PSO exhibit certain strengths, their performance in path planning optimization is notably weaker in comparison to LGPSO, suggesting the need for further research and refinement to enhance their efficiency.

Table 8. Comparison of optimal paths for each algorithm.

	LGPSO	**GAPSO**	**DEPSO**	**PSO**
Times	Path length	Path length	Path length	Path length
1	1768.1247	2153.3162	2038.9932	2199.0006
2	1770.6639	2075.3105	2052.4824	2226.5190
3	1787.4794	2125.3744	2005.1949	2093.3901
4	1808.1679	2034.0863	1966.2196	2214.3915
5	1767.3313	2147.9138	2087.4669	2185.3727
6	1807.2969	2159.0797	2104.7675	2230.4805
7	1760.2967	2131.7145	2078.6496	2182.9303
8	1809.2969	2047.7225	2104.0591	2189.8022
9	1770.6639	2083.5089	2001.0443	2272.0466
Average path	1783.26 (100%)	2106.45 (118.12%)	2048.76 (114.87%)	2199.33 (123.33%)

4.4. Feasibility of Running Path Planning Optimization Algorithms on Embedded Systems

Due to experimental limitations, direct performance testing on platforms like the NVIDIA Jetson is not feasible. However, by referencing existing research and system examples, it is evident that similar embedded hardware platforms have sufficient computational power to run complex optimization algorithms. For instance, in certain UAV path planning studies [42], the Hex Cube Black autopilot was used, which is based on the FMUv3 open hardware design of the Pixhawk project and runs the PX4 operating system (NuttX OS). This hardware system is highly flexible, supporting the integration of various sensors and communication devices, and its computational power is adequate to handle complex path planning and algorithmic tasks. Similar computational capabilities are found in platforms like NVIDIA Jetson, suggesting that our path planning optimization algorithm (LGPSO) could also run efficiently on such embedded devices. The Mission Planner software: ArduCopter V4.0.7 Quad mentioned in the study demonstrates the UAV system's ability to execute efficient flight mission tasks and path planning in multi-sensor environments. This software not only processes sensor data but also handles mission planning, flight mode settings, and real-time monitoring, further validating the finding that embedded systems can successfully support the execution of complex algorithms. Similarly, our path planning optimization algorithm requires data collection, processing, and control command generation, which shares many similarities with the flight mission planning process described in the study.

While direct experimental data are not available, by analyzing existing research and system examples, we can reasonably infer that optimization algorithms like particle swarm optimization (PSO) can be effectively executed on embedded devices, particularly on platforms with sufficient computational power and sound resource management strategies. This inference provides theoretical support for the application of our algorithm on embedded systems.

5. Conclusions

In this paper, the particle swarm algorithm LGPSO is proposed for the UAV path planning problem, focusing on safety and feasibility in the path generation process. LGPSO incorporates constraints related to optimality, safety, and feasibility through a well-designed cost function and is constructed based on the correspondence between the UAV's intrinsic motion components and the search space. In 3D real-city map test scenarios, LGPSO consistently demonstrates the ability to generate high-quality paths quickly and often finds the global optimal solution. Multiple trials of single paths in the final simulation scenario, evaluated using the standard deviation fluctuation coefficient (SDFC) and the relative standard deviation change rate (RSDCR), confirm the reliability and stability of the algorithm. In terms of path planning distance, with LGPSO's average path length set as a benchmark (100%), GAPSO, DEPSO, and PSO show relative increases of 18.12%, 14.87%, and 23.33%, respectively. These results indicate LGPSO's ability to generate shorter paths, improving flight efficiency and reducing energy consumption. Additionally, a 3D visualization of the paths generated by LGPSO reveals smooth trajectories with gentle climbs and turns, effectively reducing unnecessary power consumption and ensuring stable and safe flight.

This study introduces a quantitative framework for scientifically evaluating LGPSO's performance. Specifically, the standard deviation and maximum values of turning angles and climb angles are calculated to objectively measure the degree of directional changes. The normalized smoothness score and efficiency score provide a robust basis for evaluating the path planning performance, and a final score formula F_{final} integrates these metrics into a comprehensive assessment. Table 7 presents a comparative analysis of LGPSO, GAPSO, DEPSO, and PSO across nine experimental runs. LGPSO achieves an average score of 0.9526, significantly outperforming GAPSO (0.8155), DEPSO (0.8212), and PSO (0.7727). This demonstrates LGPSO's effectiveness in reducing directional changes while maintaining high efficiency, resulting in smoother turning and climbing angles compared to other algorithms. These findings are validated through quantitative metrics, providing a systematic evaluation of LGPSO's performance.

To improve the practical relevance of these results, high-threat scenarios were specifically selected for evaluation. In these environments, UAVs experience more frequent flight attitude changes, providing a more challenging test for the algorithms' ability to balance smoothness and efficiency. The superior performance of LGPSO in such conditions further highlights its adaptability and reliability in addressing complex scenarios.

In light of the findings and current limitations, future research will focus on two aspects. First, given the high complexity of real-world environments, this study only used a single simulation scenario opened step-by-step for testing. To enhance the universality of the results, future work will incorporate additional real urban environments and introduce more specific risk-interference factors to simulate the diverse challenges encountered in practical applications. Second, the current threat body setup has limitations, particularly when dealing with buildings of special shapes, often resulting in the excessive loss of usable space. Future research will focus on optimizing the threat body setup to better encapsulate buildings while preserving more space, thereby improving the algorithm's practicality and performance.

This study provides a systematic and objective approach for evaluating UAV path planning algorithms. LGPSO demonstrates its capability to optimize path smoothness and efficiency while addressing safety and feasibility constraints, offering valuable insights and practical guidance for UAV operations in complex environments.

Author Contributions: Conceptualization, Z.Z. and Q.C.; writing—original draft preparation, Z.Z.; visualization, Y.D.; writing—review and editing, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the China Transport Engineering First-Class Discipline Construction Fund (No. CZYL2024002).

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Osuwa, A.A.; Ekhoragbon, E.B.; Fat, L.T. Application of artificial intelligence in Internet of Things. In Proceedings of the 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN), Girne, Northern Cyprus, 16–17 September 2017; pp. 169–173. https://doi.org/10.1109/CICN.2017.8319379.
2. Ding, Y.; Jin, M.; Li, S.; Feng, D. Smart logistics based on the internet of things technology: An overview. *Int. J. Logist. Res. Appl.* **2021**, *24*, 323–345. https://doi.org/10.1080/13675567.2020.1757053.
3. Rejeb, A.; Rejeb, K.; Simske, S.J.; Treiblmaier, H. Drones for supply chain management and logistics: A review and research agenda. *Int. J. Logist. Res. Appl.* **2023**, *26*, 708–731. https://doi.org/10.1080/13675567.2021.1981273.
4. Gugan, G.; Haque, A. Path planning for autonomous drones: Challenges and future directions. *Drones* **2023**, *7*, 169. https://doi.org/10.3390/drones7030169.
5. Birtchnell, T. Drones in human geography. In *Handbook on Geographies of Technology*; Edward Elgar Publishing Cheltenham, UK: 2017; pp. 231–241. https://doi.org/10.4337/9781785361166.00024.
6. Jiménez-Jiménez, S.I.; Ojeda-Bustamante, W.; Marcial-Pablo, M.d.J.; Enciso, J. Digital terrain models generated with low-cost UAV photogrammetry: Methodology and accuracy. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 285. https://doi.org/10.3390/ijgi10050285.
7. Soto-Vergel, A.J.; Velez, J.C.; Amaya-Mier, R.; Pardo, M. Transforming ground disaster response: Recent technological advances, challenges, and future trends for rapid and accurate real-world applications of survivor detection. *Int. J. Disaster Risk Reduct.* **2023**, *98*, 104094. https://doi.org/10.1016/j.ijdrr.2023.104094.
8. Tadić, S.; Kovač, M.; Čokorilo, O. The application of drones in city logistics concepts. *Promet-Traffic&Transportation* **2021**, *33*, 451–462. https://doi.org/10.7307/ptt.v33i3.3721.
9. Li, Y.; Liu, M.; Jiang, D. Application of Unmanned Aerial Vehicles in Logistics: A Literature Review. *Sustainability* **2022**, *14*, 14473. https://doi.org/10.3390/su142114473.
10. Benarbia, T.; Kyamakya, K. A literature review of drone-based package delivery logistics systems and their implementation feasibility. *Sustainability* **2021**, *14*, 360. https://doi.org/10.3390/su14010360.
11. Iqab, M. Harnessing Drones for Faster, Cheaper, Greener Logistic Solutions in Challenging Environments. Master's Thesis, LAB University of Applied Sciences, Lahti, Finland, 2024. Available online: https://urn.fi/URN:NBN:fi:amk-202405039113 (accessed on 18 November 2024).
12. Agnihotri, A.; Bhattacharya, S. *The Drone and Robot on Your Street: The Future of Food Delivery*; SAGE Business Cases Originals; SAGE Publications: London, UK, 2024. https://doi.org/10.4135/9781071928875.
13. Raivi, A.M.; Huda, S.A.; Alam, M.M.; Moh, S. Drone routing for drone-based delivery systems: A review of trajectory planning, charging, and security. *Sensors* **2023**, *23*, 1463. https://doi.org/10.3390/s23031463.
14. Syd Ali, B. Traffic management for drones flying in the city. *Int. J. Crit. Infrastruct. Prot.* **2019**, *26*, 100310. https://doi.org/10.1016/j.ijcip.2019.100310.
15. Mohamed, N.; Al-Jaroodi, J.; Jawhar, I.; Idries, A.; Mohammed, F. Unmanned aerial vehicles applications in future smart cities. *Technol. Forecast. Soc. Change* **2020**, *153*, 119293. https://doi.org/10.1016/j.techfore.2018.05.004.
16. Fan, D.; Shi, P. Improvement of Dijkstra's algorithm and its application in route planning. In Proceedings of the 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, Yantai, China, 10–12 August 2010; pp. 1901–1904. https://doi.org/10.1109/FSKD.2010.5569452.
17. Sun, Y.; Fang, M.; Su, Y. AGV path planning based on improved Dijkstra algorithm. *J. Phys. Conf. Ser.* 2021, *1746*, 12052. https://doi.org/10.1088/1742-6596/1746/1/012052.
18. Candra, A.; Budiman, M.A.; Pohan, R.I. Application of A-Star Algorithm on Pathfinding Game. *J. Phys. Conf. Ser.* **2021**, *1898*, 12047. https://doi.org/10.1088/1742-6596/1898/1/012047.
19. Duhé, J.-F.; Victor, S.; Melchior, P. Contributions on artificial potential field method for effective obstacle avoidance. *Fract. Calc. Appl. Anal.* **2021**, *24*, 421–446. https://doi.org/10.1515/fca-2021-0019.

20. Yao, Q.; Zheng, Z.; Qi, L.; Yuan, H.; Guo, X.; Zhao, M.; Liu, Z.; Yang, T. Path Planning Method With Improved Artificial Potential Field—A Reinforcement Learning Perspective. *IEEE Access* **2020**, *8*, 135513–135523. https://doi.org/10.1109/ACCESS.2020.3011211.
21. Fan, X.; Guo, Y.; Liu, H.; Wei, B.; Lyu, W. Improved artificial potential field method applied for AUV path planning. *Math. Probl. Eng.* **2020**, *2020*, 6523158. https://doi.org/10.1155/2020/6523158.
22. Véras, L.G.D.; Medeiros, F.L.; Guimaráes, L.N. Systematic literature review of sampling process in rapidly-exploring random trees. *IEEE Access* **2019**, *7*, 50933–50953. https://doi.org/10.1109/ACCESS.2019.2908100.
23. Kingston, Z.; Moll, M.; Kavraki, L.E. Sampling-based methods for motion planning with constraints. *Annu. Rev. Control Robot. Auton. Syst.* **2018**, *1*, 159–185. https://doi.org/10.1146/annurev-control-060117-105226.
24. Noreen, I.; Khan, A.; Habib, Z. Optimal path planning using RRT* based approaches: A survey and future directions. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*. 97–107. https://doi.org/10.14569/IJACSA.2016.071114.
25. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. https://doi.org/10.1016/j.cie.2019.106040.
26. Pardalos, P.M.; Mavridou, T.D. Simulated annealing. In *Encyclopedia of Optimization*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 1–3.
27. Gutjahr, W.J. ACO algorithms with guaranteed convergence to the optimal solution. *Inf. Process. Lett.* **2002**, *82*, 145–153. https://doi.org/10.1016/S0020-0190(01)00258-7.
28. Mirjalili, S. Evolutionary algorithms and neural networks. *Stud. Comput. Intell.* **2019**, *780*, 43–53. https://doi.org/10.1007/978-3-319-93025-1.
29. Kok, K.Y.; Rajendran, P. Differential-evolution control parameter optimization for unmanned aerial vehicle path planning. *PLoS ONE* **2016**, *11*, e0150558. https://doi.org/10.1371/journal.pone.0150558.
30. Yarat, S.; Senan, S.; Orman, Z. A Comparative Study on PSO with Other Metaheuristic Methods. In *Applying Particle Swarm Optimization: New Solutions and Cases for Optimized Portfolios*; Mercangöz, B.A., Ed.; International Series in Operations Research & Management Science; Springer: Berlin/Heidelberg, Germany, 2021; Volume 306, pp. 49–72.
31. Ajeil, F.H.; Ibraheem, I.K.; Sahib, M.A.; Humaidi, A.J. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Appl. Soft Comput.* **2020**, *89*, 106076. https://doi.org/10.1016/j.asoc.2020.106076.
32. Zheng, L.; Yu, W.; Li, G.; Qin, G.; Luo, Y. Particle swarm algorithm path-planning method for mobile robots based on artificial potential fields. *Sensors* **2023**, *23*, 6082. https://doi.org/10.3390/s23136082.
33. Qiuyun, T.; Hongyan, S.; Hengwei, G.; Ping, W. Improved particle swarm optimization algorithm for AGV path planning. *IEEE Access* **2021**, *9*, 33522–33531. https://doi.org/10.1109/ACCESS.2021.3061288.
34. Li, X.; Yu, S. Three-dimensional path planning for AUVs in ocean currents environment based on an improved compression factor particle swarm optimization algorithm. *Ocean Eng.* **2023**, *280*, 114610. https://doi.org/10.1016/j.oceaneng.2023.114610.
35. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), Anchorage, AK, USA, 4–9 May 1998; pp. 69–73. https://doi.org/10.1109/ICEC.1998.699146.
36. Gad, A.G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. https://doi.org/10.1007/s11831-021-09694-4.
37. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. https://doi.org/10.1007/s00500-016-2474-6.
38. Li, J.; An, Q.; Lei, H.; Deng, Q.; Wang, G.-G. Survey of Lévy Flight-Based Metaheuristics for Optimization. *Mathematics* **2022**, *10*, 2785. https://doi.org/10.3390/math10152785.
39. Chen, H.-M. Greedy methods in plume detection, localization and tracking. In *Greedy Algorithms*; Intech: Vienna, Austria, 2008; pp. 305–324.
40. Deng, D.; Ma, Y.; Gong, J. Multi UAV Cooperative mission planning based on parallel GAPSO algorithm. *Electro Opt. Control.* **2016**, *23*, 18–22. https://doi.org/10.3969/j.issn.1671-637X.2016.11.005.
41. Parouha, R.P.; Verma, P. A systematic overview of developments in differential evolution and particle swarm optimization with their advanced suggestion. *Appl. Intell.* **2022**, *52*, 10448–10492. https://doi.org/10.1007/s10489-021-02803-7.
42. Mesquita, R.; Gaspar, P.D. A Novel Path Planning Optimization Algorithm Based on Particle Swarm Optimization for UAVs for Bird Monitoring and Repelling. *Processes* **2022**, *10*, 62. https://doi.org/10.3390/pr10010062.