Multi UAV 3D Path Planning

Omar Magdy, Abdallah Mohamed, Waleed Atef

***Abstract*—This paper examines the performance of four optimization algorithms for multi-UAV path planning. The algorithms are evaluated based on their ability to minimize path length, avoid collisions, and enhance operational efficiency. Among the tested approaches, the Teaching-Learning-Based Optimization (TLBO) algorithm outperforms the others in both computational efficiency and path quality. The study provides insights into the strengths and limitations of each optimization technique and offers recommendations for selecting the most suitable algorithm for specific applications.**

***Index Terms*—Path Planing, 3D, UAVs.**

I. Introduction

This paper examines the performance of four optimization algorithms for real-time multi-UAV path planning. The algorithms are evaluated for their ability to minimize path length, avoid collisions, and enhance operational efficiency in dynamic environments. The study provides insights into the strengths and weaknesses of multiple optimization technique and offers recommendations for selecting the most suitable algorithm based on the specific task requirements. In 2019, Ropero et al. conducted a study to optimize the travel distance between UAVs and UGVs for planetary exploration. The research focused on multiple robot systems, including UAVs and AUVs. The Energy Constrained UAV and Charging Station UGV routing problem (ECU-CSURP) was a key constraint, leading to the development of the Optimal Operation Search Algorithm (TERRA), which helps in placing charging stops, hitting target points, reducing UGV travel distance, and computing the Total Service Point (TSP) for both UGVs and UAVs. [1] In 2020, researchers began to explore environmental constraints, such as avoiding capture by adversaries in complex confrontation environments. They solved the model using GWO, which inspired multi-cooperative systems for path planning. Improvements on GWO revealed better costs overall in mean, variance, and even during worst and best runs. [2] In 2021, Shahid et al. published a paper focusing on path planning in unmanned vehicles, focusing on aerial vehicles rather than hybrid models. They discussed various types of heuristics, such as GA, PCO, ACO, and GWO, and concluded that time, cost, and energy are the most efficient factors to be managed. [3]. Ruiz & Shirabayahi's 2023 paper discussed the Internet of Drones (IoD) and the need for optimization and betterment of route planning. They analyzed various heuristical and meta-heuristical algorithms, including A*, Greedy, Single-based, and population-based. They also discussed the advantages of various heuristical methods, such as VD, RRT, PRM, APF, and Djikstra. [4]. In 2022, Wan et al. conducted an experiment that improved the representation of the world in 3D, giving it more realism. They proposed an accurate UAV 3-D path planning approach using an enhanced multi-objective swarm intelligence algorithm (APPMS). The method involved multi-objective modeling of total flight distance and terrain threat with constraints, individual encoding, population initialization, and ranking for initial paths, and evolutionary operations, population updating, and selection for searched 3-D paths. [5]. The authors urged simulation closer to reality, including flight conditions, flight velocity, and more complex 3-D terrain data, to promote rapid data collection and decision-making for emergency response scenarios in the future.

[1]Mechatronics Department - Faculty of Engineering and Materials Science - German University in Cairo, Egypt
omar.abdelwaly, waleed.abdelrahman, abdallah.falah

II. Problem Formulation

A. Problem Statement

This project aims to optimize UAV path planning in a 3D environment by minimizing total flight distance while ensuring safe distances from obstacles. The optimization will employ meta-heuristic techniques, building upon the findings of Wan et al. [5].

B. Decision Variables

Let:

- (x_i, y_i, z_i) represent the coordinates of each waypoint i in the 3D environment.

C. Objective Functions

The optimization involves minimizing the following objective functions:

- **Objective Function 1 (Minimize Total Flight Distance):**

$$f_1 = \sum_{i=1}^{n+1} d_{i,i-1}$$

where $d_{i,i-1}$ is the distance between consecutive waypoints x_{i-1} and x_i.

- **Objective Function 2 (Minimize Danger Level):**

$$f_2 = \sum_{i=1}^{n} \sum_{j=1}^{n_g} \left(\frac{d_{\text{safe}}}{i,j} \right)^2$$

where d_{safe} is the safe distance from the UAV to the obstacles represented by the index j, and n_g is the total number of obstacles.

- **Objective Function 3 (Minimize Penalties):**
Position Penalty: The position penalty $\text{Penalty}_{\text{pos},j}$ is applied if the current position $p_{\text{current},j}$ is farther from the goal than the next position $p_{\text{next},j}$. Otherwise, it is zero.

$$\text{Penalty}_{\text{pos},j} = \begin{cases} \text{PF} \cdot (p_{\text{current},j} - p_{\text{next},j}) \cdot j, & \text{if } p_{\text{current},j} \geq p_{\text{next},j} \\ 0, & \text{otherwise.} \end{cases}$$

Distance Penalty: The distance penalty $\text{Penalty}_{\text{dist},j}$ is applied if the distance between the current position and the next position exceeds the maximum allowed distance MaxDist. Otherwise, it is zero.

$$\text{Penalty}_{\text{dist},j} = \begin{cases} \text{PF} \cdot (\text{dist}_j - \text{MD}) \cdot j, & \text{if } \text{dist}_j > \text{MD}, \\ 0, & \text{otherwise.} \end{cases}$$

Total Penalty: The total penalty is the sum of the *Position Penalty* and *Distance Penalty* over all track points j.

$$f_{\text{penalty}} = \sum_{j=1} \text{Penalty}_{\text{pos},j} + \text{Penalty}_{\text{dist},j}$$

Description of Variables:

- f_{penalty}: Total penalty for the drone's path.
- PF: A scalar value that scales the penalty contribution.
- $p_{\text{current},j}$: Current position of the drone at index j.
- $p_{\text{next},j}$: Next position of the drone at index $j+1$.
- dist_j: Euclidean distance between $p_{\text{current},j}$ and $p_{\text{next},j}$.
- MD: Maximum allowed distance between consecutive points.
- j: Index of the current track point, used to weight penalties progressively.
- j: index that determines the number of Drones.

D. Constraints

The following constraints must be satisfied to ensure safe and efficient UAV operation:

- **Constraint 1 (Maneuverability Constraint):**

$$\alpha = \arccos\left(\frac{L_{i-1,i}^2 + L_{i,i+1}^2 - L_{i-1,i+1}^2}{2 \cdot L_{i-1,i} \cdot L_{i,i+1}}\right) \leq a_{\max}$$

  This constraint ensures that the angle between the paths to consecutive waypoints does not exceed the maximum allowable angle $a_{\max}$.

- **Constraint 2 (Flight Angle Constraint):**

$$\beta = \arctan\left(\frac{H}{L}\right) \leq \beta_{\max}$$

  where H is the altitude difference between waypoints and L is the horizontal distance. This constraint limits the maximum angle of ascent or descent of the UAV.

- **Constraint 3:** The track points chosen for the drone are not similar to each other and are not similar to any obstacles.

$$\forall i \in \{1, 2, \ldots, n_t\}, \forall j \in \{1, 2, \ldots, n_{\text{obst}}\} :$$

$$(x_i, y_i, z_i) \neq (x_j, y_j, z_j) \quad \text{for all} \quad j \quad \text{(obstacle points)},$$

$$(x_i, y_i, z_i) \neq (x_k, y_k, z_k) \quad \text{for all} \quad k \neq i$$

(other track points),

where n_t is the number of track points, and n_{obst} is the number of obstacles.

- **Constraint 4 (Line Validity Constraint):** The straight-line path between each consecutive track point must be free from obstacles and other drones. This is checked using the 3D Bresenham algorithm for line generation.

$$\forall i \in \{1, 2, \ldots, n_t - 1\} \quad \text{and for each point} \quad \mathbf{p}$$

$$\text{on the line between} \quad (x_i, y_i, z_i) \quad \text{and}$$

$$(x_{i+1}, y_{i+1}, z_{i+1}) :$$

$$\mathbf{p} \notin \text{obstlist} \quad \text{and} \quad \mathbf{p} \notin \text{Droneinfo}$$

This ensures that no obstacles or drones are present along the direct path between consecutive track points.

E. Output

The output of the optimization algorithm is a 1D array of the sequence of track points for all drones combined. The output array is defined as:

$$\mathbf{P} = [(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_{n_t}, y_{n_t}, z_{n_t})]$$

where $\mathbf{P}$ contains the ordered set of all track points assigned to each drone, ensuring that the constraints are satisfied and the objective functions are minimized.

F. Assumptions

In this study, several assumptions and considerations have been made to facilitate the path planning problem for drones. Firstly, we assume that the position and altitude of the drone can be adjusted directly by the algorithm, while the velocity and acceleration are controlled by the drone's onboard controller, meaning the algorithm does not control these dynamics. The entire map is assumed to be known and quantifiable, with the locations of obstacles represented as fixed, static points, all measured in integers and meters. Furthermore, we assume that the drone is equipped with perfect sensors that provide accurate, noise-free information about its position and the environment, which enables precise path planning. While obstacles are considered static for most of the problem, dynamic obstacles are also taken into account, which may move or change over time, requiring the path planning algorithm to adapt in real-time to avoid collisions. Finally, boundary conditions are in place to ensure the drone remains within a predefined operational area, with the drone prohibited from exceeding the map's limits. These assumptions aim to simplify the problem and reduce the complexity of environmental variables, although in real-world applications, factors like sensor noise and dynamic obstacles may require additional consideration.

III. Methodology

A. Simulated Annealing Algorithm (SA)

Simulated Annealing (SA) is a probabilistic optimization algorithm that explores large search spaces by accepting new solutions based on their energy (cost) and the current temperature. The algorithm starts with an initial temperature and uses geometric cooling to gradually decrease the temperature with a specified cooling rate, enabling a transition from exploration to exploitation. The process iterates until a stopping criterion, such as a minimum temperature or maximum iterations, is reached.

New solutions are generated by perturbing the current solution:

$$x_{\text{new}} = x_{\text{old}} + r_x, \quad y_{\text{new}} = y_{\text{old}} + r_y, \quad z_{\text{new}} = z_{\text{old}} + r_z,$$

where r_x, r_y, r_z are random values to introduce stochasticity in the search.

The algorithm probabilistically accepts new solutions to escape local optima, balancing exploration and exploitation effectively.

B. Genetic Algorithm (GA)

Genetic Algorithm (GA) is a natural selection-based optimization technique suitable for solving combinatorial problems and exploring large search spaces. The algorithm starts by initializing the population and selecting elite solutions based on fitness. Elites are preferred for crossover and mutation to generate offspring. Lower fitness values (representing better solutions) are favored in this minimization problem.

Crossover is performed using the whole arithmetic combination method, where offspring are generated as follows:

$$x_1 = \alpha \cdot x_{\text{parent1}} + (1-\alpha) \cdot x_{\text{parent2}},$$
$$y_1 = \alpha \cdot y_{\text{parent1}} + (1-\alpha) \cdot y_{\text{parent2}},$$
$$z_1 = \alpha \cdot z_{\text{parent1}} + (1-\alpha) \cdot z_{\text{parent2}},$$

where α is a weight factor between 0 and 1.

Mutation is introduced by adding random noise to the current solution, ensuring diversity:

$$x_{\text{new}} = x_{\text{old}} + \text{noise}_x, \quad y_{\text{new}} = y_{\text{old}} + \text{noise}_y, \quad z_{\text{new}} = z_{\text{old}} + \text{noise}_z,$$

where $\text{noise}_x, \text{noise}_y, \text{noise}_z$ are small random values.

The process of selection, crossover, and mutation is repeated iteratively until the stopping criterion, such as a fixed number of generations or a convergence threshold, is met.

C. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is inspired by the social behavior of bird flocks. The algorithm begins by initializing the population size, number of particles, iterations, and other parameters, including inertia weights (w), cognitive (c_1), and social (c_2) coefficients. Each particle has a position and velocity in the search space.

The position and velocity updates for particle i in the x, y, and z dimensions are governed by:

$$p_d^i = p_d^i + v_d^i$$
$$v_d^i = w_d \cdot v_d^i + c_{d1} \cdot r_1 \cdot (p_{\text{best}_d}^i - p_d^i) + c_{d2} \cdot r_2 \cdot (g_d - p_d^i)$$

where $d \in \{x, y, z\}$, p_d^i and v_d^i are the position and velocity components, $p_{\text{best}_d}^i$ is the personal best, g_d is the global best, and r_1, r_2 are random values between 0 and 1.

Using a star topology, all particles communicate to share the global best solution. Velocity updates are constrained by a maximum velocity limit. The algorithm iterates until a stopping criterion, such as convergence or a maximum number of iterations, is met.

D. Teaching-Learning-Based Optimization (TLBO)

Teaching-Learning-Based Optimization (TLBO) is a population-based metaheuristic inspired by classroom dynamics, consisting of two phases: Teaching Phase and Learning Phase. The algorithm initializes the population size (students), iterations, and stopping criteria.

In the Learning Phase, each student's position $S_{i,j,k}$ is updated based on the fitness comparison with another student $S_{b,j,k}$, using the equations:

$$p_k = S_{i,j,k} \pm r \cdot (S_{i,j,k} - S_{b,j,k}),$$

where r is a random value, and the direction ($+$ or $-$) depends on the fitness comparison.

In the Teaching Phase, students update their positions by moving toward the best student (teacher) $T_{j,k}$, considering the mean position of the population $\text{Mean}_{j,k}$ and a randomized teaching factor TF:

$$p_k = S_{i,j,k} + r \cdot (T_{j,k} - TF \cdot \text{Mean}_{j,k}).$$

These updates iteratively refine the solutions until convergence or a stopping criterion is met.

E. Performance Indices and Algorithm Selection

Performance indices, including best cost, average cost, and standard deviation, are essential for comparing optimization algorithms. Best cost reflects the optimal solution found, average cost indicates overall performance, and standard deviation measures result consistency. These metrics guide the selection of the most suitable algorithm—such as SA, GA, PSO, or TLBO—based on the problem's characteristics and the algorithm's strengths in finding high-quality and reliable solutions.

IV. Results and Discussion

The algorithms in this study were configured with the following parameters for optimization:

General Parameters for All Algorithms:

- The algorithms performed for **100 iterations**.
- The **number of drones** was set to **2**.
- The grid size was defined as **10 x 10 x 10**.

- The **safe distance** between drones and obstacles was set to **2**.
- The **max horizontal angle** was limited to **75**°.
- The **max vertical angle** was restricted to **65**°.
- The **penalty factor** used for penalizing constraint violations was set to **10**.
- The **number of track points** for the drones was defined as **5**.
- The **maximum distance between track points** was set to **4**.

Simulated Annealing (SA) Parameters:

- The **final temperature** for cooling was set to **0.01**.
- The **cooling rate** was defined as **0.8**.
- The **initial temperature** was set to **800**.
- The number of **new solutions generated per iteration** was **2**.

Genetic Algorithm (GA) Parameters:

- The **population size** was set to **10**.
- The percentage of **elites** was set to **20%** (0.2).
- The percentage of **mutants** was set to **40%** (0.4).
- The percentage of **children** generated during crossover was set to **40%** (0.4).

Particle Swarm Optimization (PSO) Parameters:

- The weight for the **X dimension** was set to **0.85**.
- The weight for the **Y dimension** was set to **0.9**.
- The weight for the **Z dimension** was set to **0.4**.
- The **cognitive parameters** were set as follows:
  - For the X dimension: **2.2**.
  - For the Y dimension: **2.5**.
  - For the Z dimension: **1**.
- The **social parameters** were set as follows:
  - For the X dimension: **1.1**.
  - For the Y dimension: **1.5**.
  - For the Z dimension: **0.6**.
- The **number of particles** was set to **50**.

Teaching-Learning-Based Optimization (TLBO) Parameters:

- The **number of students** was set to **50**.

These parameters were selected after extensive trial and error. Through multiple experiments, these configurations have proven to yield the best performance so far in optimizing the 3D path planning problem.

A. SA Results

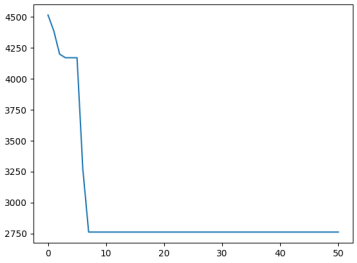

Fig. 1: Cost convergence curve SA

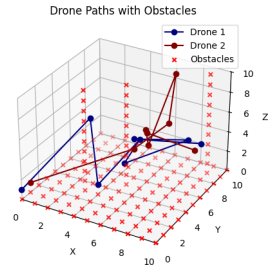

Fig. 2: Path plot SA

This figure illustrates the results obtained when at chosen parameters using SA.

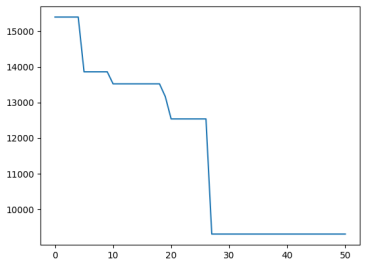

Fig. 3: Cost convergence curve SA with increased number of drones

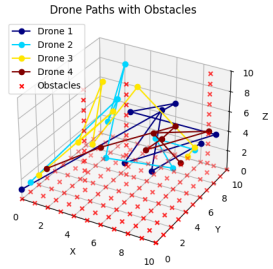

Fig. 4: Path plot SA with increased number of drones

The results in this figure demonstrate the impact of increasing the number of drones on both the cost convergence and path plots SA.

B. GA Results

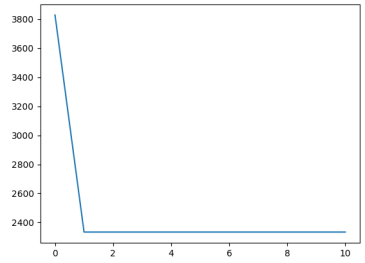

Fig. 5: Cost convergence curve for GA

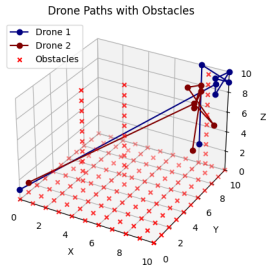

Fig. 6: Path plot for GA

This figure illustrates the results obtained when at chosen parameters using GA.

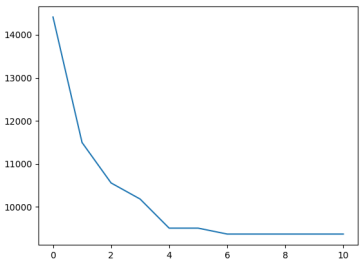

Fig. 7: Cost convergence curve for GA with increased number of drones

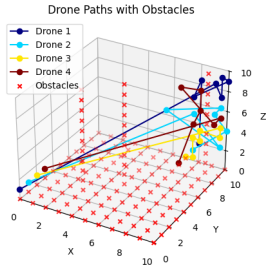

Fig. 8: Path plot for GA with increased number of drones

The results in this figure demonstrate the impact of increasing the number of drones on both the cost convergence and path plots when using GA.

C. PSO Results

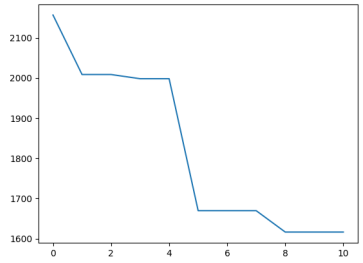

Fig. 9: Cost convergence curve for PSO

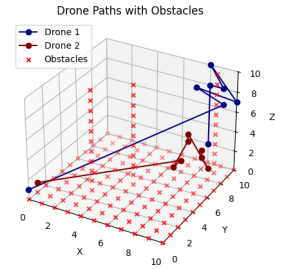

Fig. 10: Path plot for PSO

This figure illustrates the results obtained when at chosen parameters using PSO.

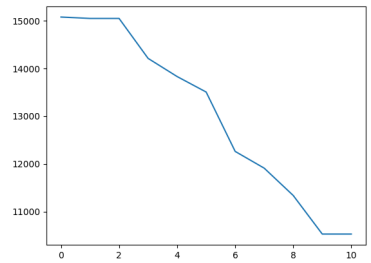

Fig. 11: Cost convergence curve for PSO with increased number of drones

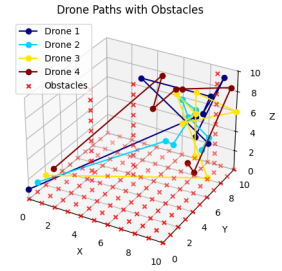

Fig. 12: Path plot for PSO with increased number of drones

The results in this figure demonstrate the impact of increasing the number of drones on both the cost convergence and path plots when using PSO.

D. TLBO Results

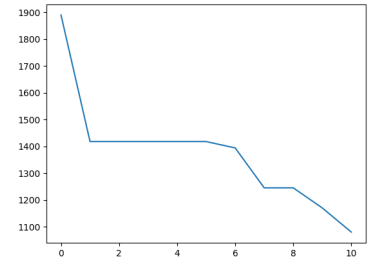

Fig. 13: Cost convergence curve for TLBO

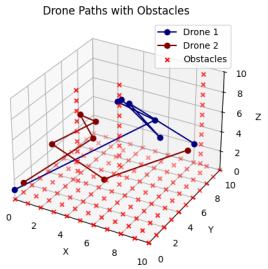

Fig. 14: Path plot for TLBO

This figure illustrates the results obtained when at chosen parameters using TLBO.

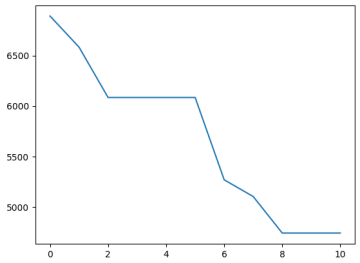

Fig. 15: Cost convergence curve for TLBO with increased drones

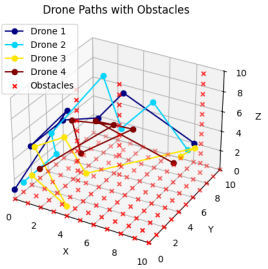

Fig. 16: Path plot for TLBO with increased drones

The results in this figure demonstrate the impact of increasing the number of drones on both the cost convergence and path plots when using TLBO.

E. Performance Indices:

Algorithm	Best Cost	Average	Standard Deviation
SA	1626.135601	2559.138445	504.7294579
GA	1750.64430599196	1969.743236	231.4685849
PSO	1699.114426	1853.937292	88.95554565
TLBO	589.1857517	772.7015465	128.3792142

TABLE I: Performance Comparison of Algorithms

Based on the performance comparison in Table I, the TLBO algorithm achieved the best cost with a value of 589.19, outperforming the other algorithms. The PSO algorithm achieved the best average value with 1853.94, indicating that it generally provides a good balance between exploration and exploitation. The GA algorithm displayed the lowest standard deviation of 231.47, which reflects its more consistent performance in finding optimal solutions across iterations.

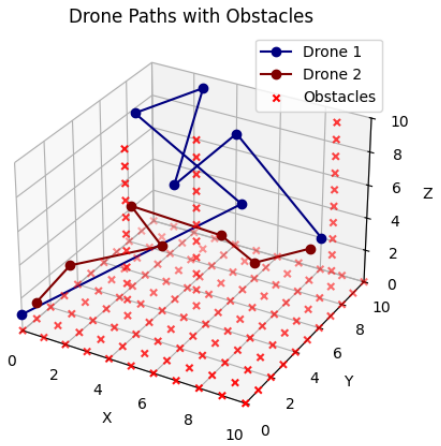

Fig. 17: Path of the Best Cost Algorithm (TLBO)

Additionally, a plot of the best cost solution illustrates the TLBO algorithm's effectiveness 17. It visualizes the drones' optimized trajectory, showcasing obstacle avoidance and contextualizing the numerical results.

V. Conclusion and Future Recommendations

In conclusion, while the TLBO algorithm achieved the best cost, each algorithm has its strengths in different areas, with PSO excelling in average performance and GA providing more consistent results.

However, we recommend considering machine learning algorithms, as they can be trained more effectively and have the ability to adapt to the problem at hand. Machine learning approaches, particularly reinforcement learning, could offer a more dynamic and scalable solution to complex path planning tasks, improving overall performance and robustness in changing environments.

References

[1] F. Ropero, P. Muñoz, and M. D. R-Moreno, "Terra: A path planning algorithm for cooperative ugv–uav exploration," *Engineering Applications of Artificial Intelligence*, vol. 78, pp. 260–272, 2019.

[2] C. Xu, M. Xu, and C. Yin, "Optimized multi-uav cooperative path planning under the complex confrontation environment," *Computer Communications*, vol. 162, pp. 196–203, 2020.

[3] N. Shahid, M. Abrar, U. Ajmal, R. Masroor, S. Amjad, and M. Jeelani, "Path planning in unmanned aerial vehicles: An optimistic overview," *International Journal of Communication Systems*, vol. 35, no. 6, p. e5090, 2022.

[4] "Toward uav path planning problem optimization considering the internet of drones."

[5] Y. Wan, Y. Zhong, A. Ma, and L. Zhang, "An accurate uav 3-d path planning method for disaster emergency response based on an improved multiobjective swarm intelligence algorithm," *IEEE Transactions on Cybernetics*, vol. 53, no. 4, pp. 2658–2671, 2022.