Swarm-based 4D path planning for drone operations in urban environments

Wu, Yu; Low, Kin Huat; Pang, Bizhao; Tan, Qingyu

2021

Wu, Y., Low, K. H., Pang, B. & Tan, Q. (2021). Swarm-based 4D path planning for drone operations in urban environments. IEEE Transactions On Vehicular Technology, 70(8), 7464-7479. https://dx.doi.org/10.1109/TVT.2021.3093318

https://hdl.handle.net/10356/152550

https://doi.org/10.1109/TVT.2021.3093318

Swarm-based 4D Path Planning for Drone Operations in Urban Environments

Yu Wu[1, 2], Kin Huat Low[3], Bizhao Pang[4], Qingyu Tan[2]*

***Abstract*—Drones have a wide range of applications in urban environments as they can both enhance people's daily activities and commercial activities through various operations and deployments. With the increasing number of drones, flight safety and efficiency become the main concern, and effective drone operations can make a difference. Accordingly, 4D path planning for drone operations is the focus of this paper, and the swarm-based method is proposed to solve this complicated optimization problem. Under the framework of 'AirMatrix', the problem is solved in two levels, i.e., 3D path planning for a single drone and conflict resolution among drones. In the multi-path planning level, multiple alternative flight paths for each drone are generated to increase the acceptance rate of a flight request. The constraints on a single flight path and two different flight paths are considered. The goal is to obtain several different short flight paths as alternatives. A clustering improved ant colony optimization (CIACO) algorithm is employed to solve the multi-path planning problem. The crowding mechanism is used in clustering, and some improvements are made to strengthen the global and local search ability in the early and later phases of iterations. In the task scheduling level, the conflicts between two drones are defined in two circumstances. One is for the time interval of passing the same path point, another one is for the right-angle collision between two drones. A three-layer fitness function is proposed to maximize the number of permitted flights according to the safety requirement, in which the airspace utilization and the operators' requests are both considered. A 'cross-off' strategy is developed to calculate the fitness value, and a 'distributed-centralized' strategy is applied considering the task priorities of drones. A genetic algorithm (GA)-based task scheduling algorithm is also developed according to the characteristic of the established model. Simulation results demonstrate that 4D flight path of each drone can be generated by the proposed swarmed-based algorithms, and safe and efficient drone operations in a specific airspace can be ensured.**

Index Terms* — *drones; urban environments; 4D path planning; swarm-based method; multi-path planning; task scheduling

I. Introduction

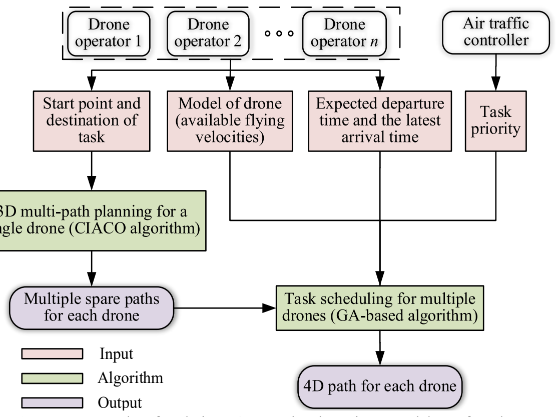
Unmanned aerial vehicles (UAVs) or drones have recently become more popular in civil and commercial applications as they can accomplish many tasks which are beyond the human or ground vehicles' ability, by taking the advantages of drone's height climbing, far reaching and speed capabilities. The applications of UAVs can also enhance the efficiency of executing tasks and reduce the pressure of over-crowded ground traffic [1]. It is more attractive if UAVs can be used in urban environments to facilitate people's lives, such as timely delivery, photography, or even air taxi in near future [2, 3]. Public agents and relevant industry will also benefit from UAVs in daily routines (surveillance, surveying and mapping, etc.) and emergencies (rescue, target tracking, etc.) [4, 5]. With multiple UAVs operating for their respective tasks over the same airspace in a city area, the airspace may be crowded, and the drones may collide with each other. The flight safety must be maintained throughout the drone operations to further improve the operation efficiency [6].

According to the regulation of Federal Aviation Administration (FAA) in the United State (CFR PART 107), small and light UAVs can fly below the height of 400*ft* (about 122*m*) from the ground [7]. Furthermore, it is recommended by Amazon that the space between the heights of 400*ft* and 500*ft* (about between 122*m* and 152*m*) is set as the isolation area to separate the UAVs and the manned aircraft [8]. Referring to the above information, the drones are regulated to fly below 120*m* in this study, so the drone must avoid the buildings by flying around them rather than flying above them. Besides, the drones can perform various tasks by flying between different pairs of start point and destination respectively, such as delivery, surveillance and target tracking.

Under such a circumstance, the path planning problem of drones is studied in this paper, in which the conflicts among multiple drones are considered. To be specific, the 3D flight path of drone is generated first without considering the time information, then in task scheduling, the time information of path is added, and the conflicts among drones are resolved by adjusting their departure times and flight velocities, i.e., a 4D path planning problem for multiple drones. The idea is inspired by the operation in civil aviation, in which the flight scheduling problem is decomposed into two parts to reduce the difficulty of solving the problem [9]. Different from the existing literatures that treating the problem from the user's perspective, manager's perspective is adopted to generate the 4D flight paths for drones from the standpoint of unmanned aerial system (UAS) traffic management (UTM).

According to the above descriptions, the framework of solving this 4D path planning problem is shown in Fig. 1.

[1] College of Aerospace Engineering, Chongqing University, Chongqing 400044, China
[2] Air Traffic Management Research Institute (ATMRI), Nanyang Technological University (NTU), Singapore 637460
[3] School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798; Principal Investigator of ATMRI's UAS Program [4] School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798


Drone operator 1
Drone operator 2
Drone operator n
Air traffic controller
Start point and destination of task
Model of drone (available flying velocities)
Expected departure time and the latest arrival time
Task priority
3D multi-path planning for a single drone (CIACO algorithm)
Multiple spare paths for each drone
Task scheduling for multiple drones (GA-based algorithm)
4D path for each drone
Input
Algorithm
Output
Fig. 1. Framework of solving 4D path planning problem for drone operations

In Fig. 1, there are two levels addressing the 4D path planning problem, and the essential information is provided by the drone operators and the air traffic controller. In the multi-path planning level, multiple paths for a drone are generated by the clustering improved ant colony optimization (CIACO) algorithm to increase the acceptance rate of a flight request, and the goal is to generate multiple different paths for each drone. In the task scheduling level, the results of multi-path planning for drones, the scheduled time information of flights and the task priorities are regarded as the inputs, and a flight scheme containing the 4D path information of drones is generated by the genetic algorithm (GA)-based method.

The main contributions of this work can be concluded as follows:

1. The model for this 4D path planning problem is developed. The urban environments and the flight rules of drone are formulated. Under the framework, the problem is divided into two subproblems. First, the 3D multi-path planning model for a single drone is established. Especially, the distance between two paths is defined to distinguish them in the multi-path planning problem. For multiple drones flying, the conflict model between two drones is described, and a three-level fitness function is designed to reflect both the airspace utilization and the operators' requests.

2. A multi-path planning algorithm based on the CIACO algorithm is proposed to generate several alternative flight paths for each drone, which can increase the accepted rate of flight request. In the CIACO algorithm, the crowding mechanism is introduced in the clustering process. Compared to the existing clustering algorithm, the number of sub-swarms are determined by the quality of individuals rather than giving a specified number. The basic ACO algorithm is also improved to strengthen the global search ability in the early stage of iteration and enhance the local search ability later. As far as the authors know, there is few publications concentrating on the ACO algorithm with multi-population. The proposed CIACO algorithm cannot only enrich the ACO theory, but also can be applied to other situations when multiple different solutions are required.

3. A GA-based task scheduling algorithm is developed to determine the 4D paths of drones. Compared to the current studies, the task priorities of drones are considered. With the idea of 'distributed-centralized' scheduling, the drones with different priorities are scheduled in a distributed mode, and the drones with the same priority are scheduled in a centralized mode. In this way, the explicit meaning of task priority is indicated, and the constraints are decomposed into many sub-problems. A 'cross-off' strategy is introduced to calculate the three-layer fitness value. With the developed task scheduling algorithm, the difficulty of solving the large-scale optimization problem is reduced, which can be popularized to similar complicated scheduling problems.

II. Related Works

UAV path planning problems have received great attentions, and various UAVs, such as quadrotor [10], solar-powered UAV [11] and tiltrotor [12] are introduced to execute different tasks in urban environments. According to the complexity of path planning, two categories can be got, i.e., path planning for a single UAV and coordinated path planning for multiple UAVs. On the other hand, the path planning can be conducted in the continuous and discrete space based on the requirement of the specific problem [13]. The above issues are highly relevant to the work and the latest progress will be discussed below.

In the path planning problem for a single UAV, it is sometimes required that multiple paths are generated simultaneously as alternatives. The multi-path planning problem is mainly solved by the swarm and evolutionary algorithms as those algorithms can be run parallelly under the idea of subpopulations. There have been many literatures focusing on this issue, and the subpopulations are usually initialized by calculating the distance between different individuals. Two individuals which have the minimum distance are thought to be in the same subpopulation, and the above process is repeated until the number of subpopulations has reached the specified value [14]. There are two ways to treat the subpopulations, i.e., the roles of subpopulations are the same or different. In Ref. [15], K-means clustering method is used to divide the whole population into K subpopulations. In each subpopulation, the solutions with bad quality will be abandoned. Then the idea of niche is introduced to make the search conducted in their own subpopulation, thus generating K paths at one time. Different from K-means clustering method, the whole population is divided into several subpopulations in Ref. [16]. As there are different forms of position update formulas in differential algorithm (DE), such as DE/rand1, DE/rand/2, DE/best/1 and DE/current-to rand/1, each subpopulation can select a kind of formula to update the solutions, which can make each subpopulation evolve from different directions.

As for the discrete path planning for a single UAV, A^* algorithm, GA and ACO algorithm are usually applied as their original forms can be used in discrete optimization problem. In Ref. [17], a risk A^* algorithm, i.e., an ad-hoc variant of the A^* algorithm, is developed to conduct the offline path planning based on the information related to static risk factors. In GA, some modifications on the operators are made to improve the algorithm performance. For example, some individuals are selected and are analyzed to judge the searching value of different regions in Ref. [18], and the regions of evolution operator are reasonably restricted. To make the crossover operator more efficient, two nodes from different chromosomes are chosen so that they are closer to each other than one of them is to its adjacent node [19]. As for the ACO algorithm, the improvements are mainly from two aspects. One is to change the way of determining the next path point, and a guidance factor is added to the original formula in Ref. [20] to make the ants go toward the end point or certain direction. The other one is to modify the formula of updating the pheromone. To further strengthen the influence of good solution, only the pheromone corresponding to the best solution in the current iteration is added to the current pheromone matrix in Ref. [21].

The present studies on the path planning of multiple UAVs focus on the cases in the continuous space, and the distributed approaches are paid much attention. A distributed velocity-aware algorithm and collision avoidance algorithm is proposed to serve motion planning of multiple UAVs in Ref. [22]. The velocity-aware algorithm generates paths with acceleration vectors that converge to the predefined destinations, and the collision avoidance algorithm will be triggered to protect UAVs from collisions when the conflicts are predicted. The cooperative path planning problem of multiple UAVs with multi-objective functions is addressed in Ref. [23]. The multi-objective model with preemptive priorities is solved using fuzzy satisfactory optimization to balance the requirement of multi-objective optimization and preemptive priorities. Receding horizon control [24] and distributed model predictive control are often combined to deal with multi-UAV path planning problem, in which each UAV can determine its own action considering the constraints only related to itself [25]. Under the framework of model predictive control, various nature inspired optimization algorithms, such as particle swarm optimization (PSO) algorithm [26], adaptive grasshopper optimization algorithm (AGOA) [27] and improved grey wolf optimizer (IGWO) [28], can be applied in generating the optimal trajectories.

There are fewer literatures concerning the coordinated path planning of UAVs in the discrete environment. In Ref. [29], four algorithms are proposed to calculate the discrete paths for UAVs, i.e., attraction approach, fuzzy logic approach, adaptive-network-based fuzzy inference system approach and PSO algorithm. All the four approaches are used to calculate the locations of the waypoints to be followed by the UAVs to minimize the distance and the risk. An adaptive path replanning method for UAVs is developed in the discrete space considering the uncertain and dynamic environments. Three strategies are designed to cope with the conflicts with the new no-fly zone, cooperative drones and non-cooperative drones [30].

To summarize the above-described works, the swarm-based methods which are inspired from the nature and society are widely applied due to their fewer requirements on the model and high computational efficiency, but appropriate algorithm needs to be selected carefully considering the characteristic of the specific problem. The multi-path planning problems are mainly solved in the continuous space in the existing literatures. As the path planning algorithms cannot be applied in the discrete space directly, the idea of subpopulation should combine with the discrete path planning algorithm to solve the multi-path planning problem in the discrete space. Furthermore, in many cases, the UAVs are regarded as homogeneous vehicles, and the task priority is not considered. The strategy of resolving the conflicts between UAVs in the discrete space also needs to be explored.

III. MODELING FOR THE 4D PATH PLANNING PROBLEM

In this section, the concept 'AirMatrix' [31] is introduced to describe the urban environments and the flight rules of drone. There are three assumptions in the 'AirMatrix' before establishing the 4D path planning model, and the details are presented below.

1. The drones considered in this study are UAVs with multiple rotors, which can climb and descend vertically.
2. The drone flies at a low velocity, and the velocity doesn't change much. Therefore, it can be approximately thought that the drone flies at a constant velocity.
3. The drone can be treated as a free point with three degrees of freedom as the attitude of drone is not concerned in the 4D path planning problem. Nevertheless, the safe distance is set to ensure that the drone must keep a certain distance away from the buildings and other drones.

A. *Environment modeling and the flight rules of drone*

The buildings described in 'AirMatrix' is defined as cubes with specific sizes, as shown in Fig. 2.

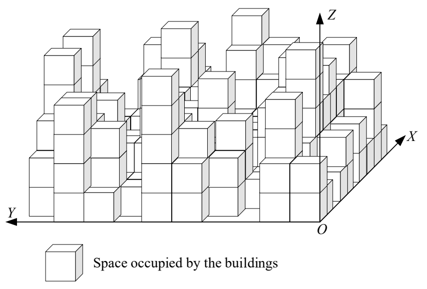


Fig. 2. Illustration of the buildings in 'AirMatrix' [32]

The buildings represented by cubes depicted in Fig. 2 are considered as the 'prohibited' areas for drone operations. For the path points defined in 'AirMatrix', the drones are allowed to fly along the straight line between two adjacent nodes, as illustrated in Fig. 3.

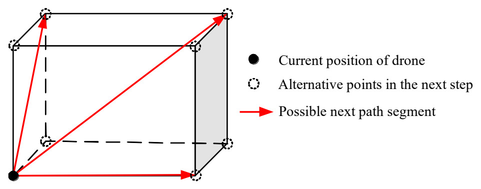

Current position of drone
Alternative points in the next step
Possible next path segment

Fig. 3. Possible routes for drone flying according to the cube of 'AirMatrix'

In Fig. 3, an edge, a face diagonal or a body diagonal of a cube can be chosen as the next path segment, and there are

26 optional path segments. In most cases, the number of optional path segments is less than 26. For example, the drone is forbidden to fly back, which will increase the length of flight path. This requirement can be achieved by setting an appropriate fitness function to guide the drone to head for the destination. Besides, the drone is sometimes located at the boundary of 'AirMatrix' or there are buildings nearby, and the number of optional path segments is also less than 26 in this case. Note that the edge length of the cube is set considering the maneuverability of drone and the safe distance between two drones (also the distance between the drone and the building is considered), thus making the drone fly safely and not change the flight height or turn frequently. In a real flight, the straight line between two adjacent nodes is regarded as the input of the flight controller, and there are some errors between the straight line and the real flight trajectory of drone due to the tracking error. Therefore, the path points are just provided as the reference trajectory for the inner flight controller, and the real flight trajectory of drone is influenced by the path planning and flight control together.

B. *Multi-path planning model for a single drone flying*

As defined above, there are a certain number of path points in each flight path, and they are the variables to be optimized. The constraints come from two aspects, i.e., a single flight path and multiple flight paths. For a single flight path, all its path points must be within the space occupied by 'AirMatrix' and keep certain distance with the ground to ensure the safety. Assume that the edge length of the cube is a, the above constraints can be denoted by

$$\begin{cases} x_{i+1} = x_{min} + ai, \quad i = 0,1,\dots,\frac{x_{max}-x_{min}}{a} \\ y_{j+1} = y_{min} + aj, \quad j = 0,1,\dots,\frac{y_{max}-y_{min}}{a} \\ z_{k+1} = z_{min} + ak, \quad k = 0,1,\dots,\frac{z_{max}-z_{min}}{a} \end{cases} \tag{1}$$

where x_{min}, x_{max}, y_{min}, y_{max}, z_{min} and z_{max} are the boundaries of 'AirMatrix', x_i , y_i and z_i are the alternative path points. To further narrow the search space and accelerate the search speed, the constraints in Eq. (1) can be converted to the following form:

$$\begin{cases} x_{i+1} = L_X + ai, \quad i = 0,1,\dots,\frac{U_X-L_X}{a} \\ y_{j+1} = L_Y + aj, \quad j = 0,1,\dots,\frac{U_Y-L_Y}{a} \\ z_{k+1} = z_{min} + ak, \quad k = 0,1,\dots,\frac{z_{max}-z_{min}}{a} \end{cases} \tag{2}$$

where $L_X = max\,\{x_{min}, (x_1 - la)\}$ and $U_X = min\,\{x_{max}, (x_{N_{max}} + la)\}$ denote the lower and upper boundary along axis *OX* respectively. The subscript '1' and N_{max} mean the start and end point of path along axis *OX*. The same definitions are made for L_Y and U_Y along axis *OY*. In Eq. (2), the space area in *OXY* plane is a rectangle whose length and width are both enlarged by 2*la* based on the original rectangle defined by the start and end point, and *l* is the amplification factor. The flight height is usually composed by several layers, so the search space in direction *OZ* is kept unchanged.

Besides, each path point must keep a certain distance away from the buildings, and the repeated path points are not allowed in a flight path to avoid the detour. The above two constraints can be written as

$$min\left\{\sqrt{\left(x_{n,} - \tilde{x}\right)^2 + \left(y_{n,} - \tilde{y}\right)^2 + \left(z_{n,} - \tilde{z}\right)^2}\right\} \geq a, (\tilde{x}, \tilde{y}, \tilde{z}) \in S_B \tag{3}$$

$$\sqrt{(x_{n1} - x_{n2})^2 + (y_{n1} - y_{n2})^2 + (z_{n1} - z_{n2})^2} \neq 0 \tag{4}$$

where S_B is the space occupied by the buildings, and $(\tilde{x}, \tilde{y}, \tilde{z})$ is a point belonging to S_B. n_1 and n_2 are two different path points in the same flight path. Eq. (3) implies that the distance between the drone and a building must be no less than a. The maximum number of path points also should be set to avoid the drone deviating the destination too much. In other words, the drone is not allowed to make too many detours when flying to the destination. The constraints on the battery capacity, the flying time and the distance of drone also can be reflected in some degree. Besides, it will also take a lot of time for the path planning algorithm to determine a large number of path points. Assume that the maximum number of path points and the permitted number of path points are N_{max} and N_{permit} ($N_{max} \leq N_{permit}$), the N_{permit} is set as the sum of path points in each direction, as expressed by

$$N_{permit} = \frac{|x_{de} - x_{st}|}{a} + 1 + \frac{|y_{de} - y_{st}|}{a} + 1 + \frac{|z_{de} - z_{st}|}{a} + 1 \tag{5}$$

To ensure that the constraint of Eq. (5) can always be met, the drone is allowed to fly through N_{permit} path points at most in the path planning algorithm.

For each drone, multiple flight paths can be submitted to increase the probability of flight permission. The distance of two flight paths can be evaluated by

$$R_{PQ} = \sum_{s=1}^{N_{path}-1} d(P_s, Q_s) \tag{6}$$

where R_{PQ} denotes the distance between flight path P and Q, and the flight path is divided into N_{path} pieces. Note that P_s and Q_s are the *s*th equal diversion point of the two flight paths, and the definition of $d(P_s, Q_s)$ is shown in Fig. 4.

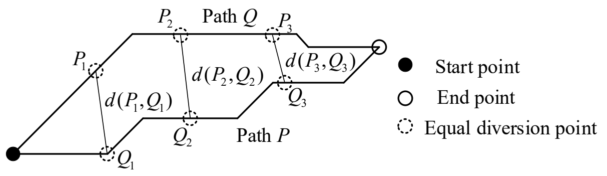


Fig. 4. Definition of $d(P_s, Q_s)$

If R_{PQ} is greater than a specified threshold D_T, the two flight paths are defined to be different, as given by

$$R_{PQ} > D_T \tag{7}$$

Next, a fitness function is designed to evaluate the quality of a flight path, which is given by

$$J_{path} = \left(\sum_{i=1}^{N_{max}-1} ||G_{i+1} - G_i||\right) + ||G_{N_{max}} - Tar|| \tag{8}$$

where G_i is the *i*th waypoint, and *Tar* is the position of destination. In Eq. (8), the first item is the total flight distance of drone, and the second item is the penalty value which denotes the distance between the last path point of drone and the destination. When the drone can reach the target, the penalty value is 0.

Assume that M_{max} different paths are required to be generated for each drone, the path points on each path are to be optimized, and the goal of path planning is to minimize the lengths of M_{max} different paths (denoted in Eq. (8)) subjected to the constraints in Eqs. (2), (3), (4), (5) and (7).

Note that the energy consumption or the maximum endurance is not considered directly in the path planning model because before the drone operator has submitted the flight request, a preliminary evaluation must be conducted. The task with specific start point and destination which is beyond the cruising ability of a specific drone will not be submitted, and only the drones with enough energy are further considered for path planning. The flight request contains the basic parameters of drone (size and performance) and the information of the task (task type, expected departure time, latest arrival time and so on).

C. *Task scheduling model for multiple drones*

After the flight request has been submitted to the air traffic controller, a task priority will be assigned to each drone according to the importance of task. For example, the emergency such as rescue and fire flighting will be assigned a high task priority, and the task priorities of some entertainment applications such as aerial photography and media broadcast are low. Besides, drones are regulated to fly at a constant speed during the flight, and the flight velocity can be selected from three values, i.e., V_H, V_M and V_L, which represent the high-speed mode, middle-speed mode and low-speed mode respectively. In real applications, as there are various types of drones, the values of V_H, V_M and V_L may be different for heterogeneous drones.

To determine a flight scheme for drones, the departure delay, the flight velocity and the flight path number must be provided, and they are also the decision variables in this task scheduling problem. With the information, the 4D flight path of each drone can be determined with a known requested departure time. Assume that the number of drones is N_U, and a drone is free of conflict with the arrival time and the other drones if the following three constraints are satisfied:

$$T_{d-r}^{u} + t_{delay}^{u} + \frac{L^u}{V^u} \le T_{a-l}^{u} \tag{9}$$

$$|T^u(x_o, y_o, z_o) - T^v(x_o, y_o, z_o)| > \Delta t \tag{10}$$

$$\begin{cases} \boldsymbol{if}\ P_u(n_1) + P_u(n_1+1) = P_v(n_2) + P_v(n_2+1) \\ \qquad\qquad\qquad \boldsymbol{then} \\ T^u(n_1) > T^v(n_2+1)\ or\ T^v(n_2) > T^u(n_1+1) \end{cases} \tag{11}$$

where $u, v \in \{1, 2, \ldots, N_U\}$. In Eq. (9), T_{d-r}^{u} and t_{delay}^{u} are the requested departure time and the departure delay of drone *u*. L^u is the length of flight path, and V^u is the flight velocity. T_{a-l}^{u} denotes the latest acceptable arrival time. If the actual arrival time of drone *u* is later than T_{a-l}^{u}, the corresponding flight will be rejected. Eq. (10) means that if the same path point (x_o, y_o, z_o) is contained both in the flight paths of drones *u* and *v*, the time interval of passing the same point for drone *u* and *v* (denoted as $T^u(x_o, y_o, z_o)$ and $T^v(x_o, y_o, z_o)$, respectively) must be greater than the safety time interval Δt to avoid the conflict. However, Eq. (10) is not sufficient to ensure the no-conflict flights. Another form of conflict (called as the right-angle collision) between drones *u* and *v* is shown in Fig. 5.

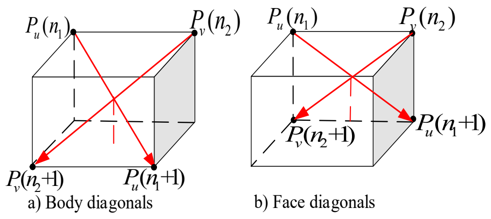


Fig. 5. Right-angle collision between two drones

In Fig. 5, n_1 and n_2 are the serial numbers of path point of drone *u* and drone *v* respectively, and $P_u(n_1)$ is the corresponding position. $T^u(n_1)$ is the moment that the drone *u* passes $P_u(n_1)$. Although drones *u* and *v* are both safe at the two positions, they will collide with each other during the travel between $P_u(n_1)$ and $P_u(n_1+1)$ (or $P_v(n_2)$ and $P_v(n_2+1)$). The mathematical form of checking this conflict is given in Eq. (11).

To sum up, the constraints from Eq. (9) to Eq. (11) describe the possible conflicts in task scheduling from two aspects. Eq. (9) presents the constraint for a single drone, and the conflicts between two drones are shown in Eqs. (10) and (11). Actually, the constraint that the distance between two drones cannot be too small at a specific moment is converted to an equivalent constraint that the time interval of passing the same point must be greater than a certain value, as expressed in Eqs. (10) and (11). The reason can be explained as follows. In the 'AirMatrix', it is not so convenient to calculate the position of drone at a specific moment as the drone is not located at the vertex of cube most of the time. With the above operation, the process of conflict detection becomes much concise.

D. *Evaluation index for the task scheduling model*

To maximize the airspace utilization, the number of permitted drone operations in certain airspace should be as many as possible under the precondition of safety. Therefore, the number of permitted flights is regarded as the fitness function to estimate the flight scheme, as expressed by

$$J_{sch}^{1} = max\ \{\textstyle\sum_{w=1}^{N_U} a_w\} \tag{12}$$

where J_{sch}^{1} denotes the maximum number of permitted flights. $A = [a_w]_{1\times N_U}$ is an array to record the information on flight permission of drones. $a_w = 1$ if drone *w* is permitted to fly, otherwise $a_w = 0$. However, as the number of flights is an integer, it is possible that there are equal number of permitted flights in two flight schemes. In this case, the total departure delay of all the flights is regarded as the second layer of fitness function because the operators always prefer to minimize the departure delay as given by

$$J_{sch}^{2} = \min\ \{\textstyle\sum_{w=1}^{N_U} a_w t_{delay}^{w}\} \tag{13}$$

where J_{sch}^{2} denotes the minimum total departure delay of all the flights. In a few cases, when the total departure delays of two flight schemes are the same, the preference on the flight path should be considered. For each drone, the operator always prefers to select a shorter flight path, and the flight path with shorter length will be assigned a smaller number. Therefore, the sum of operators' flight path number is set as the third layer of fitness function as given by

$$J_{sch}^{3} = \min \{\textstyle\sum_{w=1}^{N_U} a_w m_w\} \quad (14)$$

where J_{sch}^{3} denotes the minimum sum of operators' flight path number, and m_w is the flight path number of drone *w*.

To sum up, the decision variables of the task scheduling problem are the departure delay (t_{delay}^{u}), the flight velocity (V^u) and the flight path number of each drone, and the goal is to maximize or minimize the three-layer fitness function in Eqs. (12)-(14). Eqs. (9)-(11) are the constraints must be met in this task scheduling problem.

IV. Multi-path planning by clustering improved ant colony optimization algorithm

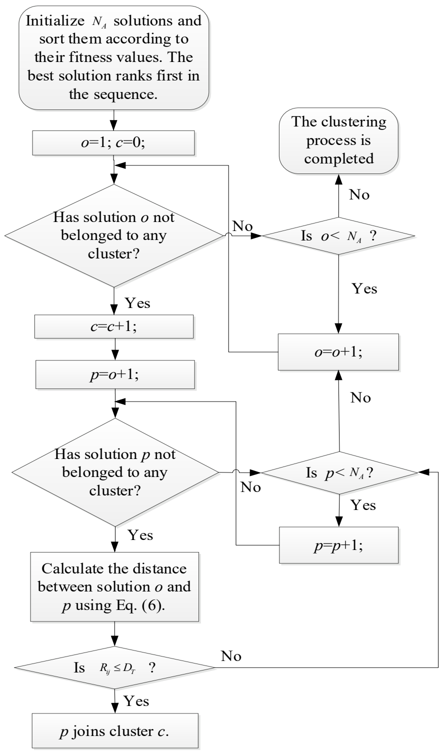
As there are multiple alternative flight paths for each drone, a multi-path planning algorithm is required. Some algorithms, such as pseudo-spectral method [33], tabu search algorithm [34] and simulated annealing algorithm [35], which begin with only one initial solution fail to satisfy the demand [36]. Besides, the final optimal solution is sensitive to the initial solution, which increase the difficulty of getting the optimal solution.

Although A^* algorithm can be applied to solve the discrete path planning problem and can generate the optimal path, only one solution can be obtained, which fails to satisfy the requirement of multi-path planning. Swarm-based optimization algorithms are suitable for solving the multi-path planning problem. A certain number of initial solutions are employed in those algorithms, and those solutions are updated according to the specific formulas in every iteration. After the maximum number of iterations is reached, the solutions whose number equals to that of the initial solutions are outputted. In this way, multiple optimal solutions are obtained, but the diversity of those optimal solutions cannot be guaranteed.

Among many swarm-based algorithms, the original form of GA and ACO algorithm both can be used to solve the discrete optimization problems. However, in GA, the on-off state of every possible position of drone in 'AirMatrix' must be recorded when coding, which makes the number of optimization variables great and is inefficient for computing in the optimization process. Based on the above consideration, ACO algorithm is more suitable for this multi-path planning problem, and the discrete path points distributed in 'AirMatrix' are to be optimized, which has fewer number of optimization variables compared to GA. According to the characteristic of the established multi-path planning model, a crowding mechanism is adopted to put the solutions into different sub-swarms, and they update themselves in different directions to make the final optimal solutions diverse. Then, the clustering algorithm is integrated into ACO algorithm, and some improvements and adjustments in the original ACO algorithm are made to enhance its performance in different phases of iterations.

A. Clustering algorithm in ACO based on the crowding mechanism

First, the cluster center is determined, and good initial cluster centers can both accelerate the convergence rate of algorithm and avoid the solution being trapped into local optimum. An intuitive idea is to select the cluster centers among several best initial solutions that are different with each other, and other initial solutions can be put into the corresponding cluster according to their distance to the cluster center. Note that there is only one center in each cluster, and one initial solution only belongs to a specific cluster. Assume that the number of initial solutions is N_A, the procedures of the clustering algorithm is shown in Fig. 6.



Initialize N_A solutions and sort them according to their fitness values. The best solution ranks first in the sequence.
o=1; c=0;
Has solution o not belonged to any cluster?
Yes
No
Is $o< N_A$?
No
The clustering process is completed
Yes
c=c+1;
p=o+1;
o=o+1;
No
Has solution p not belonged to any cluster?
No
Is $p< N_A$?
Yes
p=p+1;
Yes
Calculate the distance between solution o and p using Eq. (6).
Is $R_{ij} \le D_T$?
No
Yes
p joins cluster c.

Fig. 6. Procedures of the clustering algorithm

In Fig. 6, *o* and *p* are the serial numbers of the initial solution, and *c* is the serial number of the cluster. The crowding mechanism is used to generate the initial cluster centers in Fig. 6. First, the fitness values (the length of flight path plus the penalty item) of initial solutions are evaluated by Eq. (8), and the initial solutions are sorted in an ascending order according to their fitness values. The first solution is regarded as the center of the first cluster, and the distance between the first and the second solution is calculated using Eq. (6). If the distance is smaller than D_T, the two solutions are thought to be close, and the second solution will join the cluster lead by the first solution. The same operations will go on between the first solution and other solution, and the solutions belonging to the first cluster is determined in this way. The second solution will then be picked out, and a new cluster will be generated if it does not belong to the first cluster, i.e., the constraint in Eq. (7) is satisfied. With the same manner, a certain number of clusters can be produced.

The clusters have the following characteristic: Although a solution may be close to more than one cluster center, it only belongs to the cluster whose center is with smaller fitness value. Besides, the number of clusters is not decided in advance, and it is determined by the fitness values of solutions. Compared to the fixed number of clusters, the self-adaptive number of clusters can make each cluster distinctive from others. At last, the solutions which rank in the first few places are different and can be the cluster centers. This is beneficial for getting better solutions in the later iterative process as there is at least one good solution in each cluster.

B. Improved ACO for multi-path planning

As the whole swarm has been divided into a certain number of clusters, the strategies of updating solutions in each cluster will be developed based on ACO algorithm [36]. Although the ACO algorithm has been applied in many optimization problems in industry, it still suffers from the disadvantages of falling into local optimal solution easily and slower convergence rate. Under the framework of ACO algorithm, some improvements are introduced to remedy the above drawbacks. These improvements will be highlighted when describing the steps of ACO algorithm in solving the multi-path planning problem.

B1. *Initialization*

The solution in this problem is a combination of path points from the start point to the destination. Take the start point as an example, the next path point can be selected from 26 optional points shown in Fig. 3. Then each of them will be checked if the constraints in Eqs. (2)-(4) can be met, and the next path point is randomly generated from the optional points satisfying the constraints. The above process will be terminated if the destination is reached or the number of path points has been equal to N_{permit}. Eq. (8) is called to calculate the fitness values of solutions.

Other parameters, such as the number of ants N_A, the maximum times of iterations I_{max}, effective factor of pheromone α, effectiveness of heuristic factor β, the pheromone matrix τ, the pheromone constant Q and the evaporation of pheromone ρ, are also needed initializing. Note that the size of the pheromone matrix τ is as the same as the search space to record the influence of the traveled paths. Assume that the number of clusters is N_C, each cluster should have one independent pheromone matrix to follow by.

In the original ACO algorithm, the values of Q and ρ are kept unchanged during the iterations, which fails to adjust the local or global search ability adaptively. A smaller value of ρ will increase the possibility of searching previous solutions repeatedly. While a larger value of ρ will enhance the global search ability at the cost of slowing down the convergence rate. A good algorithm should have both a strong global search ability in the early stage of iteration and a powerful local search ability in the later stages. The following form of ρ is adopted:

$$\rho(g) = \rho_0 b^{g-1}, \quad g = 1,2,\dots,I_{max} \tag{15}$$

where ρ_0 is the base evaporation of pheromone, and b is the decay factor. In Eq. (15), ρ decreases exponentially to keep the exploration ability in the early iterations and enhances the exploitation ability later. The value of Q also has an influence on the search ability of ACO algorithm. At the beginning of exploration, the search space should be expanded to increase the probability of finding better solutions. Q must be set a relatively small value to avoid a local optimal solution. As the space has been basically explored with the increasing iteration times, there is no need for a large-scale exploration, and a local search is needed to further improve the solution quality. Q needs to be set to a large value in this stage. The value of Q varies according to the following rule:

$$Q(g) = Q_0 + ln(g)c, \quad g = 1,2,\dots,I_{max} \tag{16}$$

where Q_0 is the base pheromone constant, and c is the growth factor.

B2. *Selection of path points*

According to ACO algorithm, the selected probability of optional path points can be obtained as follows:

$$p_f = \frac{[\tau_f(g)]^{\alpha}\cdot(\eta_f)^{\beta}}{\sum_{h\in optional}[\tau_h(g)]^{\alpha}\cdot(\eta_h)^{\beta}}, g = 1,2,\dots,I_{max} \tag{17}$$

$$\eta_f = \begin{cases} \frac{1}{\sqrt{(x_f-x_{tar})^2+(y_f-y_{tar})^2+(z_f-z_{tar})^2}}, & f \neq tar \\ 1 & , f = tar \end{cases} \tag{18}$$

where p_f is the probability that the path point *f* is selected, and $\tau_f(g)$ is the pheromone of path point *f* in the *g*th iteration. η_f is the heuristic information of path point *f* that can be obtained by Eq. (18). *optional* is the set of optional path points, and (x_f, y_f, z_f) and $(x_{tar}, y_{tar}, z_{tar})$ are the positions of path point *f* and the target, respectively. Equation (17) enables the optional path points closer to the target selected with higher probability.

Selecting the path point by probability can avoid the solution being trapped into local optimum to some extent, but it also slows down the convergence rate. To deal with the above situation, a random number $rand \in (0,1)$ is introduced. When *rand* is greater than a certain value, the path point is selected by probability according to Eq. (17). Otherwise, the path point with the highest probability is selected, which makes the drone fly toward the target and accelerates the convergence rate. On the other hand, the path point is also selected by probability when *rand* is smaller than a certain value, and the solution can avoid being trapped into local optimum. Note that, when the ant has reached the destination, or the number of path points equals to N_{permit}, the process of selecting the path points is ended.

B3. *Update of the pheromone*

After all the ants have finished their travels in one iteration, numerous flight paths are generated. These flight paths must be evaluated by calculating their fitness values, and the better flight path can extract more pheromone on the path point it passed by. The strategy of updating the pheromone in ACO algorithm is governed by the following equations:

$$\Delta\tau_a = \frac{Q(g)}{J_{path}(a)}, a = 1,2,\ldots,N_A \quad (19)$$

$$\tau_f(g+1) = \big(1-\rho(g)\big)\tau_f(g) + \Delta\tau_a, f \in \Gamma_a \quad (20)$$

where $J_{path}(a)$ is the fitness value corresponding to the flight path of ant a, and $\Delta\tau_a$ is the pheromone increment produced by ant a. In Eq. (20), Γ_a is the combination of path points traveled by ant a in the gth iteration, and $\tau_f(g+1)$ is the updated pheromone. The pheromone matrix is used to connect the solutions in different iterations. A good strategy of updating pheromone matrix is beneficial to the fast convergence of ACO algorithm. In this paper, to further strengthen the influence of excellent solutions, only the pheromone corresponding to the best solution in the current iteration is added to the pheromone matrix.

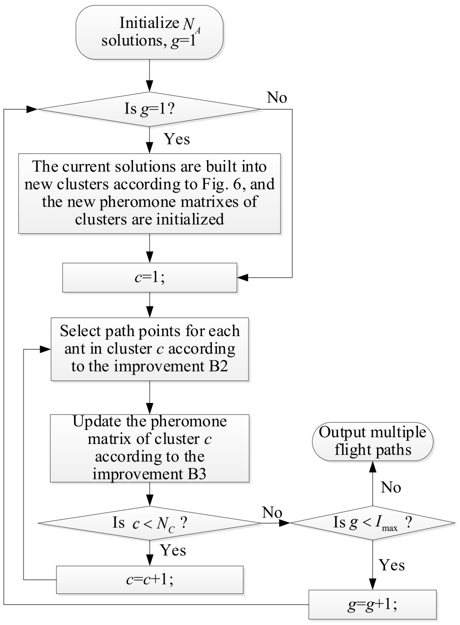
Besides, the pheromone may be accumulated at some path points after many times of update using Eqs. (19) and (20), which will lead to local optimum. The most direct way of preventing pheromone accumulation is to set its minimum and maximum value, as expressed by

$$\tau_f(g) = \max\left(\tau_{min}, min\left(\tau_f(g), \tau_{max}\right)\right) \quad (21)$$

where τ_{min} and τ_{max} are the lower and upper values of pheromone, respectively. Eq. (21) can be executed after the pheromone is updated by Eq. (20).

C. The flow of the clustering improved ACO multi-path planning algorithm

The clustering algorithm and the improved ACO (IACO) algorithm are depicted in detail in Sections III.A and III.B. In one iteration, these two parts are integrated together to form a complete loop. The flow of the clustering IACO (CIACO) multi-path planning algorithm is shown in Fig. 7.



Initialize N_A solutions, g=1
Is g=1?
No
Yes
The current solutions are built into new clusters according to Fig. 6, and the new pheromone matrixes of clusters are initialized
c=1;
Select path points for each ant in cluster c according to the improvement B2
Update the pheromone matrix of cluster c according to the improvement B3
Output multiple flight paths
No
Is $c < N_C$?
No
Is $g < I_{max}$?
Yes
Yes
c=c+1;
g=g+1;

Fig. 7. Flow of the CIACO algorithm for multi-path planning (N_C: Number of clusters)

where g is the current times of iteration, and c is the serial number of the cluster. In Fig. 7, each cluster should have one independent pheromone matrix to follow. By comparing to the standard ACO algorithm, a number of flight paths is required to be included into clusters in the initialization before the iteration process begins. The crowding mechanism shown in Fig. 7 is different from that in the continuous multi-path planning problem, and only one time of clustering process is needed in CIACO algorithm. Compared to many other swarm-based algorithms, the excellent solutions in the iterative process are not reserved directly according to the principle of ACO algorithm, and the information of the excellent solutions are passed on utilizing the pheromone matrix. In view of the above characteristic, many crowding mechanisms in the continuous algorithms cannot be used in ACO algorithm directly, or the frequent actions of clustering and information sharing will make the algorithm divergent.

After the maximum iteration time is reached, the best flight path in each cluster will be picked out to make the alternative flight paths of a drone. If the number of clusters is smaller than the required number of alternative flight paths, i.e., $N_C < M_{max}$, the missing $(M_{max} - N_C)$ alternative flight paths are supplemented by the N_Cth paths. On the contrary, if $N_C > M_{max}$, the best M_{max} paths will be selected. In this way, multiple paths with difference are generated, and they are the best paths in each cluster and are regarded as the alternative flight paths for the task scheduling of drones.

V. TASK SCHEDULING BASED ON GENETIC ALGORITHM

After multiple flight paths are generated for each drone, the task scheduling algorithm will work. Assume that the delay of departure time is an integral number of seconds, the decision variable in the task scheduling problem, i.e., the delays of departure time, the flight velocities of drones and the serial numbers of flight paths, are all discrete variables. As it is a problem with large scale and complicated constraints, evolution-based algorithms can find satisfactory schedules in a short computation time. GA is the most popular evolution algorithm and can be applied both in continuous and discrete cases, which is suitable for solving this task scheduling problem. Since the GA is proposed in 1975 [37], many new operators, such as insert [38, 39], perturb [40], delete [40] and swap [40], have been developed and added to original GA to increase the probability of obtain better solutions. In this paper, instead of following the new operators, some adjustments to the original operators, i.e., selection, crossover and mutation are made based on the characteristic of the task scheduling model. The key step in GA is to calculate the fitness value of each solution. In this paper, the 'distributed-centralized' framework is applied, i.e., the drones with different priorities are dealt with separately and the drones with the same priority are considered together. In the 'centralized' process, a 'cross-off' strategy is designed to determine the value of the three-level fitness function for the drones with the same priority. The total value of the three-level fitness function for all the drones is obtained by combining the results in each priority comprehensively.

A. Approach to calculate the fitness values

According to the established task scheduling model, after the delay of departure time, flight velocity and flight path number of each drone is given, the 4D flight paths of drones are determined. However, the maximum number of permitted flights still needs to be judged. A 'cross-off' strategy is developed to calculate the fitness value, and an example is given, as shown in Fig. 8.

Drone number	1	2	3	4	5
1		Y	N	Y	N
2	Y		Y	N	N
3	N	Y		N	N
4	Y	N	N		Y
5	N	N	N	Y	

Y: Conflict exists
N: No conflict

Fig. 8. A table to show the conflict among the drones

The conflict exists between two drones when one or more constraints in Eqs. (9), (10) and (11) are not satisfied, and the row number and column number corresponding to the two drones are marked by 'Y'. In Fig. 8, the goal is to find the maximum number of permitted flights. In other words, the number of drones should be reduced as small as possible to ensure that there is only 'N' left in the table. Assume that the number of drones in Fig. 8 is N_{ch}, the specific steps of calculating the fitness value are listed as follows.

Step 1: Check whether all the drones have conflicts with each other and record their numbers. If all drones are safe, the process of calculating the fitness values ends, or N_{re}=1.

Find out the drones with 'Y', and they are the candidates to be crossed off. In Fig. 8, all the five drones are the candidates to be rejected.

Step 2: Try to cross off N_{re} drones from the candidates (try all possible scenarios) and repeat *step* 2 if there are still 'Y' in the table, $N_{re} = N_{re} + 1$.

If there are scenarios which can wipe out all 'Y', stop and record all of them. In this case, the maximum number of permitted flights are found. In Fig. 8, all 'Y' cannot be wiped out by only crossing off one drone.

Step 3: The maximum number of permitted flights are found, and N_{re} drones are rejected.

In Fig. 8, all 'Y' can be wiped out by rejecting drones No. 2 and No. 4, and the maximum number of permitted flights is 3.

Another point that must be explained is that even a scenario which can wipe out all 'Y' is found, the remaining possible scenarios that rejecting the same number of drones still need to be found out. The reason is that when more than one scenario can result in the same maximum number of permitted flights, the best scenario should be found by comparing their second or even the third layer of fitness values (the total departure delays or the sum of operators' flight path number) in Eqs. (13) and (14). The second and the third layer of fitness values can be calculated easily as the permitted flights has been determined by the 'cress-off' strategy, and the departure delay, the flight velocity and the flight path number of each drone is provided by the GA.

The above approach to calculate the fitness value is carried out among the drones with the same task priority, and the drones are scheduled in a centralized way. When considering the drones with different task priorities, they are scheduled in a distributed way. The group with the highest task priority will be arranged first, and the flight information of drones will not be changed once it has been permitted flying. The group which is arranged later will be checked whether the drone has conflicts with the permitted ones, and it will be rejected when the conflict is detected. The 'distributed-centralized' scheduling strategy decomposes a large-scale optimization problem into smaller ones, and the constraints are also decomposed correspondingly, which reduces the difficulty of solving the problem. Besides, the drone with higher task priority will consider fewer constraints when it is being scheduled, which increases the probability of acceptance.

B. Task scheduling based on GA

In GA, the number of initial solutions is assumed as N_I. Three vectors, i.e., $D_{1\times N_U}(s)$, $V_{1\times N_U}(s)$ and $F_{1\times N_U}(s)$ $(s = 1,2,\ldots,N_I)$ are defined to present the decision variables, i.e., delayed departure times, velocities and serial numbers of flight paths of drones. After the decision variables are initialized, the fitness value can be calculated. The procedures of solving the task scheduling problem based on GA are depicted below:

1. *Selection operator*

The selection operator is introduced to choose a certain number of solutions, and those solutions will be brought into the following operators. Roulette wheel selection is used in the original GA. For a solution, the probability being selected equals to the ratio of its fitness value in the sum of the fitness values of all solutions. However, it is not suitable for this problem as the three-layer fitness function is adopted, and the roulette wheel selection can only reflect the first layer of fitness value of a solution. Here the stochastic tournament strategy is adopted. Two different solutions make up a pair, and the better one in each pair is chosen to go to the next step. Therefore, $\frac{N_I}{2}$ solutions are selected. Compared to the roulette wheel selection, the qualities of two solutions can be easily judged in the stochastic tournament, which makes the selection process more straightforward and effective.

2. *Crossover operator*

A pool is defined to hold the selected $\frac{N_I}{2}$ solutions, and the best and the worst solution in the current pool are picked out to execute the crossover operation. As the information of every drone should be involved in the crossover process, uniform crossover is adopted between two solutions, i.e., each corresponding element of the two solutions will be exchanged with the same probability p_c. Then two new solutions are generated after updating the information of every drone. Comparisons are made among two original solutions and two new solutions, and the two with better fitness values will be remained.

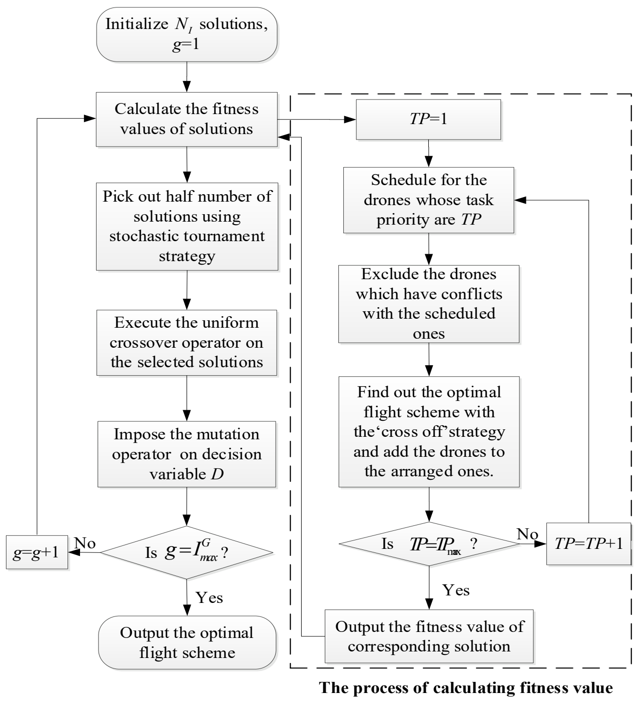
3. *Mutation operator*

As there are only several allowed values for the decision variables V and F, the effect of mutation operator will not be so obvious to the two decision variables. Therefore, the mutation operator is imposed on the decision variable D only. The delay departure time of each drone fluctuates in a small range with the probability p_m. After the mutation operation, the better solution between the original and updated ones will be remained.

After executing the above three operators, one iteration ends, and the fitness values of the updated solutions will be calculated again.

C. Flow of GA-based task scheduling algorithm

In every iteration, GA starts by computing the fitness value. The three operators are then conducted to update the solutions. When the maximum iteration times (I^G_{max}) is reached, the optimal solution will be obtained. The flow of GA-based task scheduling algorithm is shown in Fig. 9.



Initialize N_I solutions, g=1
Calculate the fitness values of solutions
Pick out half number of solutions using stochastic tournament strategy
Execute the uniform crossover operator on the selected solutions
Impose the mutation operator on decision variable D
Is $g=I^G_{max}$?
No
g=g+1
Yes
Output the optimal flight scheme
TP=1
Schedule for the drones whose task priority are TP
Exclude the drones which have conflicts with the scheduled ones
Find out the optimal flight scheme with the 'cross off' strategy and add the drones to the arranged ones.
Is TP=TP_{max} ?
No
TP=TP+1
Yes
Output the fitness value of corresponding solution
The process of calculating fitness value

Fig. 9. Flow chart of GA-based task scheduling algorithm

In Fig. 9, the modules in the dotted box are the process of calculating the fitness value, where TP is an index from 1 to 5 denoting the task priority, and TP_{max}=5 in this problem. By using the 'cross off' strategy, the maximum number of no-conflict drones with the same task priority can be found in a centralized way, and they make up the permitted flights in a solution.

VI. Simulation Studies

To investigate the rationality and superiority of the proposed 4D path planning method for drone operations, simulations under different path planning algorithms and various settings are conducted. In the first group, the standard ACO algorithm, IACO algorithm and CIACO algorithm are used to generate multiple flight paths for each drone, and comparison between the three algorithms are made. In the second scenario, the results of task scheduling with different number of drones are presented, and the relationship between the number of drones and the acceptance rate of flights are analyzed. Comparison between the 'cross-off' strategy and other relevant approaches in calculating the fitness value is also made. At last, to explore the way to further enhance the acceptance rate of flights, the influence of different number of alternative flight paths on the fitness value are discussed.

There are 300 buildings randomly distributed in 'AirMatrix'. The ranges of 'AirMatrix' along OX, OY and OZ axes are [0, 1800] m, [0, 1800] m, and [0, 90] m respectively, and the edge length of cube is set as 30 m. The optional flight velocities for drones are $V_H = 15\ \text{m/s}$, $V_M = 10\ \text{m/s}$ and $V_L = 5\ \text{m/s}$. Each flight path is divided into ten pieces, i.e., $N_{path} = 10$, and the threshold to judge whether the two flight paths are different is set to $D_T = 4000\ \text{m}$ considering the range of 'AirMatrix'. The values of N_{path} and D_T are determined by trial and error to ensure that the initial solutions can be put into the clusters which are significantly different, thus making the clusters search the path from different directions. Smaller values of N_{path} and D_T will make the difference among the clusters not so obvious. Although the number of clusters is greater, the search directions of sub-swarms show a smaller gap, which is bad for generating different paths. On the contrary, there will be fewer clusters if N_{path} and D_T are set to greater values, which may lead to insufficient number of different paths. The safety time interval Δt between two drones is set as 5s. All the results are obtained by running the programs on a desktop with Intel Core i7-3370 3.40 GHz. Note that, as the drone can take-off and land vertically and the path of the two processes are relatively fixed and cannot be further optimized, they are not considered in the multi-path planning problem. In other words, the start point and the destination of drone can be in the air in the simulations.

A. Results of multi-path planning using different forms of ACO algorithms

Three different forms of ACO algorithms are used to calculate the flight paths, and comparisons are made to demonstrate the superiority of the proposed CIACO algorithm. The parameters settings in ACO algorithms are listed in TABLE I.

TABLE I. SETTING OF PARAMETERS IN ACO ALGORITHMS

Item	N_A	I_{max}	α	β	c
Value	500	200	0.5	1	10
Item	ρ_0	b	τ_{min}	τ_{max}	
Value	0.7	0.95	0	15	

Take the start point (750, 810, 60) and destination (1770, 1710, 90) as an example, the pheromone constant Q_0 is set as Q_0 = (1770-750) + (1710-810) + (90-60), which is the sum of position differences between the start and end point along OX, OY and OZ axes, respectively. All the elements in pheromone matrix τ are initialized as 1. With the above settings, the variations of fitness values in ACO, IACO and CIACO algorithms are shown in Fig. 10.

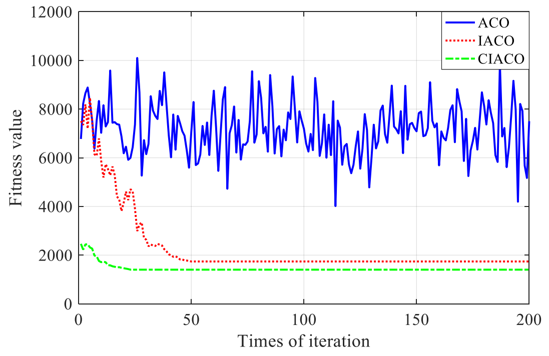


Fig. 10. Convergence curves of three algorithms

The IACO algorithm indicates that all the improvements mentioned in Section III.B are made on the original ACO algorithm. The fitness value of CIACO in each iteration corresponds to the best solution considering all the sub-swarms. The following results can be got from Fig. 10.

(a) After 200 iterations, ACO algorithm is still oscillating, and the fitness values are greater than the other two algorithms.

(b) IACO and CIACO algorithms both converge quickly, and CIACO algorithm even converges after 20 iterations. The final fitness values of IACO and CIACO algorithms are 1739.2 and 1402.3, respectively.

From the above convergence curves, ACO algorithm shows the slowest convergence rate. With the modifications in Section III.B, both the solution quality and the convergence rate are enhanced in IACO algorithm, which demonstrate the effectiveness of the proposed improvements. In addition, the solution quality can be improved by using CIACO algorithm because each cluster evolves independently towards different directions, which increases the probability of obtaining better solutions. The best four flights generated by IACO and CIACO algorithms are presented in Fig. 11 to Fig. 13.

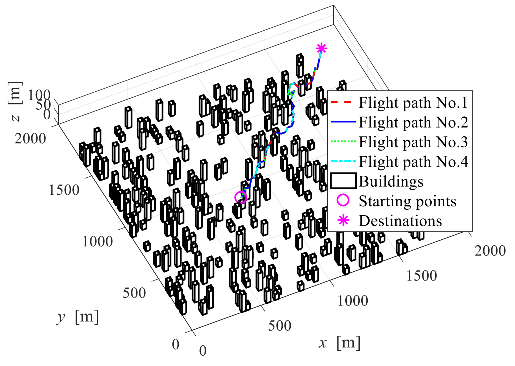


Fig. 11. Flight paths generated by IACO algorithm (3D view, and the buildings are presented by cuboids)

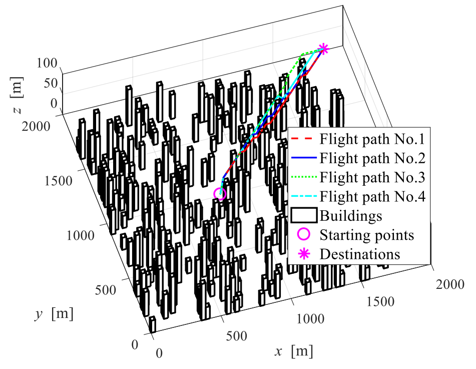


Fig. 12. Best Flight paths obtained by CIACO algorithm (3D view, and the buildings are presented by cuboids)

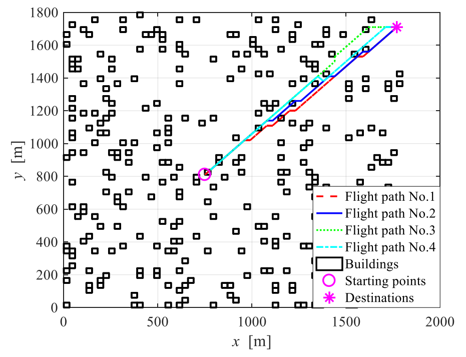


Fig. 13. Flight paths obtained by CIACO algorithm (top view of Fig. 12)

In Fig. 11, the flight paths are totally coincident, which fails to satisfy the requirements of generating multiple different flight paths. For the IACO algorithm without the clustering mechanism, the solutions are likely to be identical after many times of iterations. In CIACO algorithm, the solutions are initially updated in different directions, and it is with low probability that the best solutions in each cluster are all identical. In Fig. 12 and Fig. 13, there are still some flight path segments overlapped. This is because under the same fitness function, some flight path segments are regarded as optimal ones by many clusters. To show more results of CIACO algorithm, the fitness values of each cluster in CIACO algorithm are presented in Fig. 14.

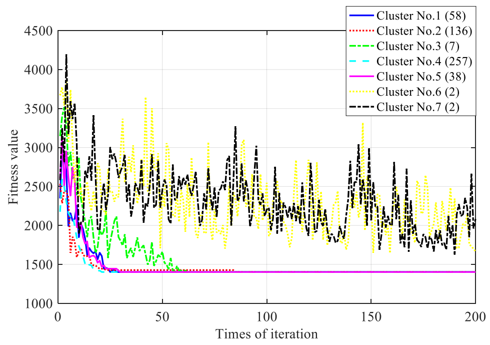


Fig. 14. Convergence curves of clusters in CIACO algorithm

The solutions fall into 7 clusters in this example, and the number in the bracket denotes the number of solutions in the corresponding cluster. In Fig. 14, the convergence curves of cluster No. 6 and No. 7 are oscillating as there are only two individuals in each cluster. The fitness values of the first five clusters are convergent with different rates, and their final fitness values are all 1402.3 but corresponding to different flight paths, as shown in Fig. 13.

The results demonstrate that the proposed CIACO algorithm is validated for generating multiple alternative flight paths for each drone, and the length of flight path can also be shortened by improving the standard ACO algorithm. Those 3D flight paths for drones can be the initial inputs for the further task scheduling process.

B. Task scheduling with different number of drones: comparison and discussion

In this section, each drone will be allocated three alternative flight paths for the task scheduling algorithm. Other inputs contain the requested departure times of drones, which are set to random integers between 0 s and 300 s, the latest arrival times of drones and the task priorities of drones. In GA, the number of chromosomes is set to $N_I = 300$, and the probabilities of executing the crossover and mutation operators are $p_c = 0.2$ and $p_m = 0.1$, respectively. GA runs for 200 iteration times, i.e., $I_{max}^G = 200$, and the best solution will be outputted as the optimal flight scheme. The convergence of the GA-based task scheduling algorithm is discussed first. Different number of drones are set to make a comparison. Note that the simulations are conducted with an increasing number of drones, and some common data are used. For example, when the number of drones is 100, the initial settings of the first half are the same with those when the number is 50. In this way, the comparison can be more reliable, and the influence of random operator information can be reduced. As the fitness function of task scheduling has three layers, the convergence curves of each layer are shown in Fig. 15 to Fig. 17.

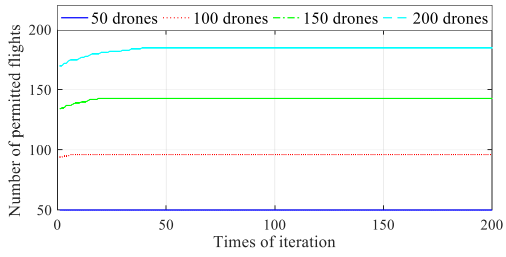


Fig. 15. Convergence curve of permitted flights

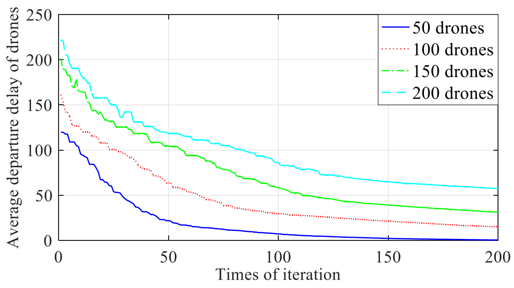


Fig. 16. Convergence curve of average departure delay of drones

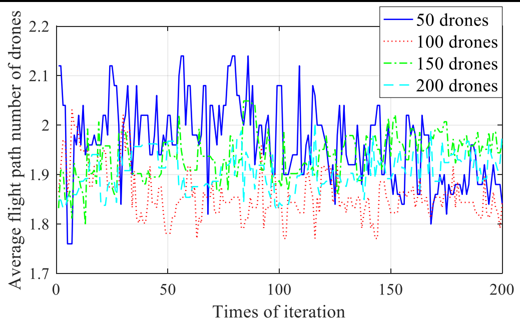


Fig. 17. Convergence curve of average flight path number of drones

In Fig. 15, the permitted flights are all convergent in the four cases, and the convergence rate is faster with a smaller number of drones. When there are 50 drones, it converges even in the first iteration. In the second layer of fitness function, the convergence rate become slower, and a greater number of drones also corresponds to a slower convergence rate and a longer average departure delay. When the number of drones is 150 or 200, the average departure delay increases in individual iterations (for example the 10th iteration and the 30th iteration in the above two cases) because the accepted number of flights has not been convergent in the corresponding iterations in Fig. 15. In Fig. 17, the third layer of fitness function is not convergent after 200 times of iteration in all the four cases. In actual situations, the goal of task scheduling is to obtain a satisfactory solution within limited computation time. In this problem, the airspace situation is reflected in the first layer of fitness function, which are the most important information in traffic management of drones. As for the third layer of fitness function, it focuses on the satisfactory degree of operators, which has a small influence on the airspace situation. Therefore, 200-time iteration is sufficient in this example although the third layer of fitness function is still not convergent. Next, the complete results of task scheduling with different number of drones are presented in TABLE II.

TABLE II. COMPLETE RESULTS OF TASK SCHEDULING WITH DIFFERENT NUMBER OF DRONES

Number of drones	Accepted number (rate of flights)	Average departure delay (sec)	Average path number	Average velocity (m/s)	Computing time (min)
20	20 (100%)	0	1.3	10.3	0.1
30	30 (100%)	0	1.7	10.2	0.3
40	40 (100%)	0	1.6	10.3	0.6
50	50 (100%)	0.44	1.9	9.9	1.1
60	60 (100%)	2.1	1.9	10	2.0
70	69 (98.6%)	5.1	1.9	9.3	3.1
80	79 (98.8%)	4.8	1.8	10	4.4
90	89 (98.9%)	9.9	1.9	10.5	5.9
100	96 (96%)	15.0	1.9	10.5	7.7
110	106 (96.4%)	23.2	1.9	9.9	9.9
120	115 (95.8%)	22.3	2.0	10.4	12.5
130	125 (96.2%)	29.9	1.9	10.4	15.2
140	135 (96.4%)	27.1	1.9	10.2	18.0
150	143 (95.3%)	30.7	2.0	9.9	21.1
160	151 (94.4%)	34.7	2.0	10.1	24.4
170	161 (94.7%)	44.7	2.0	10.5	27.9
180	169 (93.9%)	58.5	1.8	10.4	31.6
190	177 (93.2%)	54.8	1.9	10.2	35.5
200	185 (92.5%)	57.3	1.9	10.0	40.1

In general, with a greater number of drones, the accepted rate of flights decreases, and the average departure delay of drones increases. The average flight path number fluctuates, and its mean value is about 2, which implies that the second flight path is selected with higher probability. Note that the accepted rate of flight and the average departure delay sometimes show abnormal changes, such as when the number of drones is 80 and 130. This is because with random flight information, the flight paths of new drones may have no conflict with the ahead ones, which makes the accepted rate of flight increase and the average departure delay decrease. Besides, the average flight velocities of drones are about 10 *m*/*s* regardless of the number of drones. Moreover, the computing time shows a rapid growth with the increasing number of drones. This is because the number of decision variables is larger when there are more drones, and it will take GA more time to operate on these decision variables. Moreover, the computation load of the 'cross-off' strategy also increases rapidly when calculating the fitness value as each pair of drones must be checked to determine whether they have a conflict with each other. The proposed GA-based task scheduling algorithm is applied in offline computing, and the computing time can be further reduced by the workstation with high configuration.

C. Comparison between the 'cross-off' strategy and three other approaches when calculating the fitness value in task scheduling

To further demonstrate the advantage of the proposed 'cross-off' strategy in calculating the fitness value of task scheduling, the approach in Ref. [41] is introduced to make a comparison first. In Ref. [41], the task priority of drone is not considered, and an array is defined to record the serial number of drones (called as the sequence of drones for convenience) according to the first-come-first-served basis. The drones are selected one by one from the second element of the array to check the conflict with the ahead drones. If there is a conflict, only the selected drone will be rejected. In this way, the accepted number of drones can be calculated, and the second or even the third layer of fitness values (the total departure delays or the sum of operators' flight path number) can be determined. To sum up, the proposed 'cross-off' strategy is centralized, and the strategy in Ref. [41] is distributed. It is evident that the drone placed in the front of the array will have a lower probability to be rejected. In this study, when determining the sequence of drones with the same task priority, the drone with shorter flight time will be arranged in the front position of the array, and the reason is explained as follows. The drone with shorter flight time takes fewer resources of the low-altitude airspace and can free up more airspace for other drones, and they should have a higher probability to be accepted and be placed in the front position of the array. Based on the above considerations, the results of task scheduling for 200 drones under the framework of GA with centralized and distributed strategies of calculating the fitness value are shown in TABLE III.

TABLE III. COMPARIS BETWEEN THE CENTRALIZED AND DUSTRIBUTED STRATEGT IN CALCULATING THE FITNESS VALUE (200 DRONES)

Strategy	Accepted number of flights	Average departure delay (sec)	Average path number	Average velocity (m/s)	Computing time (min)
Cross-off	185	57.3	1.9	10.0	40.1
Distributed	185	64.5	2.0	10.5	35.5

In TABLE III, with the distributed strategy, it takes fewer time to obtain the result (reduced by 11.5%). It is reasonable that the times of checking the conflict between two drones are less when calculating the fitness value by the distributed strategy. The drone which has been rejected will not be checked with the drone being placed behind it in the sequence of drones, thus reducing the computing time. With the centralized strategy, the average departure delay of drones is reduced, and the average flight path number is also smaller than that in the distributed strategy, which makes the drones fly with a lower velocity. The reasons can be explained as follows. Although both the 'cross-off' strategy and the distributed strategy can check the conflicts among drones and calculate the maximum number of accepted flights, there are usually more than one combination of drones corresponding to the maximum number of accepted flights. In the distributed strategy, there is no mechanism to record all the possible combinations of drones, and only one combination is recorded according to the sequence of drones. The combination of drones recorded in the distributed strategy may not result in the minimum departure delay of drones. In the 'cross-off' strategy, all the possible combinations of drones are recorded and are further checked to determine the minimum departure delay of drones, which can ensure that the best combination of drones cannot be missed. In the offline task scheduling, the computing time of algorithm is not the main concern, and the gap of computing time between the two strategies is not too much as the number of rejected flights is small. The above results and analysis demonstrate that the proposed 'cross-off' strategy is superior to the distributed strategy in calculating the fitness value for the task scheduling problem.

Besides, as mentioned above, the proposed 'cross-off' strategy is centralized, and there are also other centralized approaches which can calculate the fitness value of this task scheduling problem, such as the simulated annealing (SA) algorithm and estimation of distribution algorithm (EDA) belonging to the meta-heuristic algorithms. To make a comparison, the results of 'cross-off' (CO) strategy, SA algorithm and EDA under different number of drones are shown in TABLE IV. Note that, GA is not used as the outer loop to obtain the results in this comparison, i.e., the departure delay, the flight velocity and the flight path number of each drone is fixed, and the reason will be explained when analyzing the results.

TABLE IV. COMPARIS AMONG THE 'CROSS-OFF' STRATEGY, SA ALGORITHM AND EDA IN CALCULATING THE FITNESS VALUE

Number of drones	Algorithm	Maximum number of drones without conflicts	Average departure delay (sec)	Computing time (sec)
	CO	25	0	0.011

25	SA	25	0	38.7
	EDA	25	0	40.7
50	CO	50	0.74	0.016
	SA	50	0.74	38.9
	EDA	50	0.74	41.2
75	CO	74	7.6	5.8
	SA	74	9.2	38.1
	EDA	74	8.6	40.5
100	CO	95	13.7	69.6
	SA	94	14.1	38.5
	EDA	94	12.9	40.8

With the increasing number of drones, the percentage of drones without conflicts is reduced, and the average departure delay get greater in all the three approaches. Compared to the 'cross-off' strategy, the results of SA algorithm and EDA are worse, and the gap grows with the increasing number of drones. As for the computing time, it increases greatly when the number of drones is greater in the 'cross-off' strategy, and the computing time for SA algorithm and EDA roughly keeps the same.

The reasons for the above results can be explained as follows. The 'cross-off' strategy is an enumeration method with some search skills in essence, so the optimal solution can be guaranteed. SA algorithm and EDA algorithm are heuristic algorithms, and the optimal solution cannot be always obtained. The computing time increases greatly in the 'cross-off' strategy because the times of enumerations get larger when there are more drones. However, the computing time fluctuates in a small range in SA algorithm and EDA as the scale of the problem does not change much. In general, in the task scheduling problem, the number of drones which have conflicts with others is usually small, and the 'cross-off' strategy is more effective and suitable for the offline scheduling after a comprehensive consideration of solution quality and computing time.

Furthermore, calculating the fitness value by the 'cross-off' strategy is only a part of work in solving the task scheduling problem. As the task scheduling is already an optimization problem which is solved by the GA-based algorithm, it will become a two-level optimization problem if the fitness value is calculated by SA algorithm or EDA in the inner loop. The above operations will make the problem complicated and reduce the computation efficiency, which is unnecessary especially when the 'cross-off' strategy can obtain the optimal solution without spending too much computing time. This is also the reason why the GA is not used in this comparison.

D. Results with different number of alternative flight paths and discussions

To further explain the rationality of submitting multiple alternative flight paths, simulations with different numbers of alternative flight paths for each drone are carried out. Take 200 drones as an example, drones with different number of alternative flight paths are considered respectively, as shown in TABLE V.

TABLE V. RESULTS OF TASK SCHEDULING WITH DIFFERENT NUMBER OF ALTERNATIVE FLIGHT PATHS (200 DRONES)

Number of flight paths	Accepted number/ rate of flights	Average departure delay (sec)	Average path number	Average velocity (m/s)	Computing time (min)
1	181 (90.5%)	57.0	1	10.1	39.1
2	185 (92.5%)	72.2	1.5	10.4	39.5
3	185 (92.5%)	57.3	1.9	10.0	40.1
4	184 (92%)	57.7	2.3	10.2	40.9
5	184 (92%)	64.2	2.6	10.6	41.8

In TABLE V, with the increasing number of alternative flight paths, the computing time does not increase so much as the number of decision variables is not changed, and only the optional values of the flight path number increase in GA. The accepted rate of flights is not always improved, which is different from the expected results. The best result is obtained when there are three alternative flight paths. The reasons can be tracked after a deep analysis of the rejected drones' data, as presented in TABLE VI.

TABLE VI. NUMBER AND TASK PRIORITY OF REJECTED DRONES

Number of alternative flight paths	Number of drones (*the number in the bracket denotes the task priority of the corresponding drone*)
1	1(5), 3(4), 38(5), 62(3), 64(5), 75(5), 81(1), 84(5), 88(3), 98(2), 100(1), 105(4), 111(3), 137(4), 146(1), 160(2), 174(5), 182(4), 192(4)
2	3(4), 38(5), 64(5), 81(1), 84(5), 88(3), 98(2), 100(1), 111(3), 137(4), 146(1), 171(2), 174(5), 182(4), 192(4)
3	3(4), 33(4), 38(5), 64(5), 81(1), 84(5), 88(3), 91(1), 98(2), 137(4), 149(3), 171(2), 174(5), 182(4), 192(4)
4	3(4), 33(4), 38(5), 84(5), 96(3), 98(2), 104(3), 111(3), 116(3), 146(1), 152(4), 154(1), 171(2), 174(5), 182(4), 192(4)
5	3(4), 33(4), 38(5), 64(5), 84(5), 98(2), 104(3),111(3), 116(3), 152(4), 153(5), 154(1), 160(2), 174(5), 182(4), 192(4)

With the increasing number of alternative flight paths (for example, the number of alternative flight paths increases from 3 to 4), the previously rejected drones with higher task priority may be accepted, such as drones No. 81 and No. 91. While the previously accepted drones with lower task priority may be influenced by the above operations and be rejected, such as drones No. 96 and No. 152. Therefore, it is not always true that a greater number of alternative flight paths will result in a better flight scheme when the task priority is considered. To further explore the relationship between the rejected drones and their task priorities, the results in TABLE VI are further concluded in TABLE VII.

TABLE VII. NUMBER OF REJECTED DRONES CATEGORIZED BY DIFFERENT TASK PRIORITIES

The number of rejected drones		Task priority				
		1	2	3	4	5
Number of alternative flight paths	1	3	2	3	5	6
	2	3	2	2	4	4
	3	2	2	2	5	4
	4	2	2	4	5	3
	5	1	2	3	5	5

In general, with the increasing number of alternative flight paths, the drones with higher task priorities (1 and 2) are less likely to be rejected. In other words, when the number of alternative flight paths is greater, the main beneficiaries are the drones with higher task priorities but there is an influence on the drones with lower task priorities. When the accepted number of flights with high task priority is smaller than the rejected number of flights with low task priority, the strategy of increasing the number of alternative flight paths makes no sense. Another way to further improve the accepted number/rate of flights is to set different numbers of alternative flight paths for drones with different task priorities.

VII. Conclusion

The 4D path planning problem for drone operations in urban environments is studied in this paper to ensure the airspace safety and improve the airspace operation efficiency. Literature investigation shows that the coordinated path planning problems for UAVs are solved from the operators' standpoint, which lacks a global consideration of airspace situation. The viewpoint from air traffic controller is introduced to coordinate the drones and determine their flight scheme.

First, the concept of 'AirMatrix' is used to describe the urban environments and flight rules of drones. In the multi-path planning level, the constraints on a single flight path and the difference of two flight paths are considered, and the shortest flight path is the fitness function. In the task scheduling level, with the known task priority, the departure delay, the flight velocity and the flight path number of each drone are regarded as the inputs, and the conflicts between two flight paths are modeled. A three-layer fitness function is established to reflect the solution quality from the perspective of airspace situation and operators' demand.

To solve the established models, CIACO algorithm is proposed to generate multiple flight paths for each drone. In this algorithm, the crowding mechanism is applied in the clustering process, and the number of clusters is determined by the solution quality rather than a specified number. Besides, several strategies are developed to improve the exploration and exploitation ability of the basic ACO algorithm in different phases of iteration. In the task scheduling problem, under the 'distributed-centralized' scheduling strategy, GA-based algorithm is designed to obtain the optimal flight scheme, and a 'cross off' approach is proposed to calculate the complicated three-layer fitness value.

In the simulation studies, the CIACO multi-path planning algorithm and the GA-based task scheduling algorithm are both validated. The CIACO algorithm is able to generate multiple alternative flight paths for each drone, and the length of flight path also can be shortened compared to the standard ACO algorithm. In GA-based task scheduling algorithm, with the increasing number of drones, the accepted rate of flight decreases, and the average delay of drones rises. The superiority of the proposed 'cross-off' strategy is verified by comparing with the distributed strategy, SA algorithm and EDA. Besides, the flight scheme is not always getting better with a greater number of alternative flight paths. The results can provide the suggestions for the design of airspace capacity and operator's preference. In the future, the path planning problem of other forms of air transportation, such as air taxi, surveillance and emergency rescue can be modeled, and other advanced bioinspired algorithms are expected to improve the safety and efficiency of drone operations.

Acknowledgements

This research work is supported by the Chongqing Research Program of Basic Research and Frontier Technology with the grant numbers of cstc2020jcyj-msxmX0602 and cstc2018jcyjAX0127, Fundamental Research Funds for the Central Universities with the project reference number of 2020CDJ-LHZZ-066, China Scholarship Council with the project reference number of 201906055030.

This collaborative research is also supported by the Ministry of Education (MOE, Singapore) Tier-1 project research grant (Project ID: 2018-T1-002-124) and the UAS Program on "Urban Aerial Transport Traffic Management and Systems" in the ATMRI, NTU, Singapore.

Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not reflect the views of any of the above-mentioned funding agents.

REFERENCES

[1] Gallacher, D. (2016). Drones to manage the urban environment: Risks, rewards, alternatives. *Journal of Unmanned Vehicle Systems*, *4*(2), 115-124.

[2] Ramana, M. V., Varma, S. A., & Kothari, M. (2016). Motion planning for a fixed-wing UAV in urban environments. *IFAC-PapersOnLine*, *49*(1), 419-424.

[3] Yao, J., & Ansari, N. (2020). Online Task Allocation and Flying Control in Fog-Aided Internet of Drones. I*EEE Transactions on Vehicular Technology*, *69*(5), 5562-5569.

[4] Meng, W., He, Z., Su, R., Yadav, P. K., Teo, R., & Xie, L. (2016). Decentralized multi-UAV flight autonomy for moving convoys search and track. *IEEE Transactions on Control Systems Technology*, *25*(4), 1480-1487.

[5] Zhang, Z., Zheng, L., & Guo, Q. (2018). A varying-parameter convergent neural dynamic controller of multirotor UAVs for tracking time-varying tasks. *IEEE Transactions on Vehicular Technology*, *67*(6), 4793-4805.

[6] Watkins, S., Burry, J., Mohamed, A., Marino, M., Prudden, S., Fisher, A., ... & Clothier, R. (2020). Ten questions concerning the use of drones in urban environments. *Building and Environment*, *167*, 106458.

[7] Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., & Robinson, J. E. (2016). Unmanned aircraft system traffic management (UTM) concept of operations. *In 16th AIAA Aviation Technology, Integration, and Operations Conference*, Reston, VA: 3292.

[8] Air, A. P. (2015). Revising the airspace model for the safe integration of small unmanned aircraft systems. *Amazon Prime Air.*

[9] Yan, S., & Tseng, C. H. (2002). A passenger demand model for airline flight scheduling and fleet routing. *Computers & Operations Research*, 29(11), 1559-1581.

[10] Hu, H., Wu, Y., Xu, J., & Sun, Q. (2019). Cuckoo search-based method for trajectory planning of quadrotor in an urban environment. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, *233*(12), 4571-4582.

[11] Wu, J., Wang, H., Li, N., Yao, P., Huang, Y., & Yang, H. (2018). Path planning for solar-powered UAV in urban

environment. *Neurocomputing*, *275*, 2055-2065.

[12] Zhang, J., Yu, W., & Qu, X. (2019). A trajectory planning model of tiltrotor considering multi-phase and multi-mode flight. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(16), 6019-6031.

[13] Wu Y. (2021). A survey on population-based meta-heuristic algorithms for motion planning of aircraft. *Swarm and Evolutionary Computation*, *62*, 100844.

[14] Zhou, D.., Li, X., Zhang, K., & Pan, Q. (2015, July). Multiple routes planning based on particle swarm algorithm and hierarchical clustering. In *2015 34th Chinese Control Conference (CCC)* (pp. 42-46). IEEE.

[15] Li, F., Hao, B., Zhao, J., & Xue, L. (2013). Cruise missile multiple routes planning based on hybrid particle swarm optimization. *Journal of Beijing Institute of Technology*, 3, 12.

[16] Dong, K., Huang, H., Huang, C., & Zhang, Z. (2017). Trajectory online optimization for unmanned combat aerial vehicle using combined strategy. *Journal of Systems Engineering and Electronics*, *28*(5), 963-970.

[17] Primatesta, S., Guglieri, G., & Rizzo, A. (2019). A risk-aware path planning strategy for uavs in urban environments. *Journal of Intelligent & Robotic Systems*, *95*(2), 629-643.

[18] Wang, X., & Meng, X. (2019, October). UAV Online Path Planning Based on Improved Genetic Algorithm with Optimized Search Region. In *2019 IEEE International Conference on Unmanned Systems (ICUS)* (pp. 1-6). IEEE.

[19] Savuran, H., & Karakaya, M. (2016). Efficient route planning for an unmanned air vehicle deployed on a moving carrier. *Soft Computing*, *20*(7), 2905-2920.

[20] Feng, Y., Cheng, J., Li, T., Chen, B., & Tang, K. (2019, October). Path Planning of Uninhabited Aerial Vehicle Added the Guiding Factor. In *2019 IEEE International Conference on Unmanned Systems (ICUS)* (pp. 866-870). IEEE.

[21] Yue, L., & Chen, H. (2019). Unmanned vehicle path planning using a novel ant colony algorithm. *EURASIP Journal on Wireless Communications and Networking*, *2019*(1), 136.

[22] Hu, Y., Yao, Y., Ren, Q., & Zhou, X. (2020). 3D multi-UAV cooperative velocity-aware motion planning. *Future Generation Computer Systems*, *102*, 762-774.

[23] Hu, C., Zhang, Z., Yang, N., Shin, H. S., & Tsourdos, A. (2019). Fuzzy multiobjective cooperative surveillance of multiple UAVs based on distributed predictive control for unknown ground moving target in urban environment. *Aerospace Science and Technology*, *84*, 329-338.

[24] Yao, P., Wang, H., & Ji, H. (2017). Gaussian mixture model and receding horizon control for multiple UAV search in complex environment. *Nonlinear Dynamics*, *88*(2), 903-919.

[25] Shim, D. H., & Sastry, S. (2008, August). A dynamic path generation method for a UAV swarm in the urban environment. In *AIAA Guidance, Navigation and Control Conference and Exhibit* (p. 6836).

[26] Wu, Y., Wang Y., Qu, X., & Sun, L. (2019). Exploring mission planning method for a team of carrier aircraft launching. *Chinese Journal of Aeronautic*s, *32*(5), 1256-1267.

[27] Wu, J., Wang, H., Li, N., Yao, P., Huang, Y., Su, Z., & Yu, Y. (2017). Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm. *Aerospace Science and Technology*, *70*, 497-510.

[28] Yao, P., Wang, H., & Ji, H. (2016). Multi-UAVs tracking target in urban environment by model predictive control and Improved Grey Wolf Optimizer. *Aerospace Science and Technology*, *55*, 131-143.

[29] San Juan, V., Santos, M., & Andújar, J. M. (2018). Intelligent UAV map generation and discrete path planning for search and rescue operations. *Complexity*, 2018, Article ID 6789419.

[30] Wu, Y. & Low, K. H. (2020). An Adaptive Path Replanning Method for Coordinated Operations of Drone in Dynamic Urban Environments, *IEEE Systems Journal*, DOI: 10.1109/JSYST.2020.3017677.

[31] Mohamed Salleh, M. F. B., Wanchao, C., Wang, Z., Huang, S., Tan, D. Y., Huang, T., & Low, K. H. (2018). Preliminary concept of adaptive urban airspace management for unmanned aircraft operations. In *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, AIAA SciTech Forum 2018, Florida, USA. DOI: 10.2514/6.2018-2260.

[32] Mohamed Salleh, M. F. B., & Low, K. H. (2017). Concept of operations (ConOps) for traffic management of Unmanned Aircraft Systems (TM-UAS) in urban environment. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, AIAA 2017-0223, 5 Jan 2017. DOI: https://doi.org/10.2514/6.2017-0223.

[33] Wang, J., Cui, N., & Wei, C. (2019). Rapid trajectory optimization for hypersonic entry using convex optimization and pseudospectral method. *Aircraft Engineering and Aerospace Technology*, *91*(4), 669-679.

[34] Blot, A., Kessaci, M. É., & Jourdan, L. (2018). Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *Journal of Heuristics*, *24*(6), 853-877.

[35] Dal Sasso, V., Fomeni, F. D., Lulli, G., & Zografos, K. G. (2019). Planning efficient 4D trajectories in Air Traffic Flow Management. *European Journal of Operational Research*, *276*(2), 676-687.

[36] Zhang, H., Wang, H., Li, N., Yu, Y., Su, Z., & Liu, Y. (2018). Time-optimal memetic whale optimization algorithm for hypersonic vehicle reentry trajectory optimization with no-fly zones. *Neural Computing and Applications*, 1-15.

[37] Parpinelli, R. S., & Lopes, H. S. (2011). New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, *3*(1), 1-16.

[38] Qingtian, H., Wenjing, C., & Jia, C. (2012, March). Research of route planning based on genetic algorithm. In *2012 International Conference on Computer Science and Electronics Engineering* (Vol. 2, pp. 199-202). IEEE.

[39] Savuran, H., & Karakaya, M. (2016). Efficient route planning for an unmanned air vehicle deployed on a moving carrier. *Soft Computing*, *20*(7), 2905-2920.

[40] Kok, J., Gonzalez, L. F., & Kelson, N. (2012). FPGA implementation of an evolutionary algorithm for autonomous unmanned aerial vehicle on-board path planning. *IEEE transactions on evolutionary computation*, *17*(2), 272-281.

[41] Tan, Q., Wang, Z., Ong, Y. S., & Low, K. H. (2019, June). Evolutionary Optimization-based Mission Planning for UAS Traffic Management (UTM). In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 952-958), IEEE. DOI: 10.1109/ICUAS.2019.8798078.