3D Drone Path Planning Algorithms with Collision Avoidance: Collision Resolution vs. Prevention

Rutuja Shivgan, *Student Member, IEEE,* Jorge Medina, *Student Member, IEEE,*
Ziqian Dong, *Senior Member, IEEE,* and Roberto Rojas-Cessa, *Senior Member, IEEE*

***Abstract*—Because of their limited flight range, multiple drones are often deployed simultaneously to perform complex tasks. The flight path is planned for each drone to follow to complete the job before task execution. However, multi-drone path planning places drones at risk of in-flight collisions. To overcome this problem, we model the multi-drone path planning problem as a multi-vehicle routing problem that maximizes job coverage subject to collision-free paths. We propose three 3D collision-free path planning algorithms, namely, XTRACT, 3DETACH, and ASCEND. XTRACT and 3DETACH provide collision-free paths by setting partial paths at different altitudes while ASCEND prevents intersecting paths at the planning phase by selecting different altitudes. We limit the search processes to two altitudes to demonstrate sufficiency with the lowest height complexity. Through exhaustive evaluations, we compare the performance of the proposed schemes and show the trade-offs between resolving and preventing collisions from path planning. We identify the best-performing strategy by using a profit model to evaluate the plethora of applicable performance metrics.**

***Index Terms*—Unmanned aerial vehicles, in-flight collisions, multiple depot vehicle routing problems, path planning, optimization, collision avoidance, 3D path planning, altitude planning.**

I. Introduction

The applications of Unmanned Aerial Vehicles (UAVs) or drones span from aerial photography, surveillance, search and rescue, delivery, agriculture, and many other domains in recent years [1]–[3]. These drones are often small in size and with limited battery capacity and flight range. Therefore, complex jobs may require the coordination of multiple drones for their completion. In such an operation, each drone covers a portion of the task. Paths are determined as a set of waypoints that a drone visits. Drones fly to a waypoint, perform the required task, and continue to the next waypoint, until completing the task or exhausting the battery. In such cases, each drone visits a unique set of waypoints and each waypoint is visited by a single drone. The drone range determines the trajectory, the number of waypoints, and the path followed, including the departure and return to the drone's depot.

In this paper, we focus on a multi-drone path planning problem where drones are tasked to collect data from a set of static waypoints that are dispersed over a determined large area. The flight paths of such drones define a multi-variable optimization problem where range and coverage are considered. However, the asynchronous nature of the problem also raises the risk of in-flight collision, which may compromise the complex task.

Different from existing work, we consider the operation of drones in realistic airspace for designing an in-flight collision-free path planning algorithm that makes use of three-dimensional (3D) space to maximize the number of waypoints covered by each drone. The flight path conservatively considers eliminating collisions while aiming at the largest waypoint coverage. This objective is translated as resolving edge intersections in a directed graph in 3D space. Solving the multi-drone path planning problem and the potential of in-flight drone collision raises the following question: is planning a collision-free flight path more cost-effective than removing possible collisions from a flight path in a scenario with a minimal number of altitudes in 3D airspace? To answer this question, we focus on designing a set of efficient collision-avoidance path planning algorithms for multi-drone path planning where drones are commissioned to cover the largest number of waypoints in a designated ground area.

The contributions of this paper are the formulation of a multi-depot vehicle routing problem (MDVRP) with the objective of maximizing the number of waypoints visited by drones in a 2D area while the drone is permitted to travel in a 3D space. We propose three path planning algorithms, namely XTRACT, 3DETACH, and ASCEND, to generate collision-free paths. XTRACT and 3DETACH remove the intersecting paths generated using a greedy collision-agnostic nearest neighbor algorithm, while ASCEND performs path planning by avoiding intersecting paths as a preventive measure. The intersection removal algorithms use different altitudes to resolve any crossing path and thus to avoid a possible collision. Here, we test two altitudes as a proof-of-concept to compare the collision prevention and removal approaches. We model the three proposed path planning algorithms and evaluate their performance through computer simulation in terms of the number of waypoints covered, the number of drones used, the traveled distance, and cost-effectiveness based on our proposed profit model. We compare the performance of these three algorithms with an algorithm that adopts a greedy collision-agnostic approach for different waypoint density scenarios. We show that ASCEND outperforms XTRACT and 3DETACH in both the number of waypoints covered and the traveled distance while achieving a profit similar to that of the collision-agnostic greedy path planning approach.

R. Shivgan, and Z. Dong are with the Department of Electrical and Computer Engineering, New York Institute of Technology, New York, NY, 10023.
E-mail: {rshivgan, ziqian.dong}@nyit.edu.
J. Medina and R. Rojas-Cessa are with the Networking Research Laboratory, Helen and John C. Hartmann Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 07102.
Email: {jmc237, rojas}@njit.edu

2023 IEEE 9th World Forum on Internet of Things (WF-IoT) | 979-8-3503-1161-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/WF-IOT58464.2023.10539596

The remainder of the paper is organized as follows. Section II presents the related work on collision-avoidance path planning. Section III introduces the multi-depot vehicle routing problem formulation. Section IV presents the proposed collision-avoidance path planning algorithms. Section V presents the simulation setup, and performance metrics, and compares the results of the three proposed path planning algorithms. Section VI concludes the paper.

II. Related work

Drones have been widely adopted in applications in disaster relief and efficient smart cities [4]–[8]. Many of these applications require multiple drones to overcome infrastructure failure or to supplement the existing ones. To provide reliable services, these drones must work collaboratively where each drone is assigned a clearly defined task and a path to traverse without potential collisions.

Path planning for drones has been extended from existing solutions for vehicle routing problems to avoid obstacles on the planned path [9]–[16]. These graph-based approaches model the environment and obstacles as graphical shapes, the targets and waypoints as vertices, and paths as edges. To find the optimal collision-free path for multiple drones, constrained optimization problems were formulated to seek an optimal solution among all the possible paths from a source to a destination for an individual or multiple drones [10], [11].

These graph-based path planning algorithms use 2D graphs with a single altitude [9], [14], [17]–[21]. Free-space algorithms were proposed where paths are generated with the same distance from the edges of the obstacles to avoid collision [17], [18]. However, these generated paths may not be optimal. An artificial bee colony algorithm was proposed that uses concave polygon convex decomposition to find a global optimal route more efficiently [19]. Because these vehicle routing problems are known to be NP-hard, heuristic algorithms, such as the Dijkstra algorithm (DA), genetic algorithm (GA), and particle swarm optimization (PSO) have been also considered for vehicle path planning [14], [20]–[22]. Ant colony algorithms have been proposed [13], [14]. Here each vehicle is considered an agent and the swarm of vehicles forms the colony. These bio-inspired algorithms mimic the behavior of ants to find an optimal path based on the highest reward. The optimization constraints are based on the task requirements such as maximizing the coverage area or minimizing the mission completion time.

Other collision-avoidance methods adopt a coordinated approach through a communication network among a swarm of drones [12], [15], [23]. These approaches consider the real-time location of each drone and use this location information to calculate and decide a safe path for each drone to follow. This coordinated approach depends on a reliable communication network and is prone to jamming attacks on the communication network. An exploration of taking flight paths at different altitudes to avoid collision on the intersected path was proposed [16]. Another collision-avoidance approach proposed a separation maintenance method to avoid collision between drone paths in both space and time [24]. A safety distance and a safety margin time were manually set to avoid drone collision. Any violation of the preset safety distance and time is added as a penalty to the cost of the planned paths. However, the added penalty does not guarantee a collision-free path, which could render an unfinished job.

In our previous work, we proposed a collision-agnostic multi-drone path planning algorithm ORBIT and two graph-based collision-free multi-drone path planning algorithms, DETACH and STEER, in a 2D space [25]. In this paper, we look into a 3D space with three proposed collision-free multi-depot drone path planning algorithms, XTRACT, 3DETATCH, and ASCEND, and compare their performance.

TABLE I: Variables Description

Variable	Description
D	The set of depots
V	The set of waypoints
r	Radius of a drone's flight range
H	Height of the elevated edge
Q	Vehicle distance capacity
L	Minimum number of waypoints in a route
$\mathbf{M}$	Distance matrix
R_k	Route for drone k
$\|R_k\|$	The number of waypoints in R_k
$\mathbf{R}$	All the paths for drones
$\mathbf{R}_d$	All the collision-free paths for drones
$d_{n_w,k}$	The back tour distance from current waypoint n_w to depot k
$d_{cum,k}$	Cumulative distance traveled by drone k

III. Multi-Depot Vehicle Routing Collision Avoidance Problem Formulation

We formulate the multi-drone path planning problem as a multi-depot vehicle routing problem. Drones are tasked with surveying a number of waypoints in a set 3D area. The following assumptions are made for the path-planning problem formulation:

- The depots are uniformly distributed in the area.
- Each drone is allocated to one depot, as its starting and ending point.
- Each drone visits waypoints within a circular area centered at its depot with radius r.
- The drones are homogeneous, each with the same vehicle distance capacity, Q.
- A drone is used only if it covers a minimum number of L waypoints in its route.

Table I outlines the variables used to describe the proposed algorithms. The set of waypoints is denoted as V and the set of depots is denoted as D. Each depot is denoted as k where $k \in D$. Here, a node is defined as either a depot or a waypoint. (a_i, b_i, c_i) and (a_j, b_j, c_j) denote the coordinates of nodes i and j, respectively. The Euclidean distance between nodes i and j, d_{ij}, is defined in (1). The pair-wise distance among all nodes forms the distance matrix, $\mathbf{M}$. R_k denotes the route for the drone starting at depot k. The number of waypoints in R_k is denoted as $|R_k|$. $\mathbf{R}$ denotes the set of all

paths for drones while $\mathbf{R_d}$ denotes the set of all collision-free paths for drones. $d_{n_w,k}$ denotes the distance of the return trip from the last waypoint in the route to depot k. $d_{cum,k}$ denotes the cumulative distance traveled by drone k.

$$d_{ij} = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2 + (c_i - c_j)^2} \tag{1}$$

The binary decision variable $x_{ijk} \in \{0, 1\}$ in (2) denotes whether an edge between nodes i and j for depot k is selected or not.

$$x_{ijk} = \begin{cases} 1, & \text{if edge } (i,j) \text{ is selected by drone } k, \\ 0, & \text{the edge } (i,j) \text{ is not selected by drone } k. \end{cases} \tag{2}$$

The path planning problem was formulated as follows:

$$\max_{j} \sum_{i \in D \cup V} \sum_{j \in V} \sum_{k \in D} x_{ijk} \tag{3}$$

Subject to:

$$\sum_{i \in D \cup V} \sum_{k \in D} x_{ijk} \leq 1, \quad \forall\ j \subseteq\ V \tag{4}$$

$$\sum_{i \in V} \sum_{j \in D} x_{ijk} = 1, \quad \forall\ k \in\ D \tag{5}$$

$$\sum_{i \in D} \sum_{j \in V} d_{ij} \sum_{k \in D} x_{ijk} \leq r, \quad k = i \tag{6}$$

$$\sum_{i \in D \cup V} \sum_{j \in D \cup V} d_{ij} \sum_{k \in D} x_{ijk} \leq Q_k \tag{7}$$

$$\sum_{i \in D \cup V} \sum_{j \in V} x_{ijk} \geq L, \quad \forall k \in D \tag{8}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1, \quad \forall\ S \subseteq\ V, |S| \geq 2 \tag{9}$$

$$(x_{ijk} x_{pql}) H_{ijk,pql} = 0,\ i \neq p,\ j \neq q,\ k \neq l \tag{10}$$

The objective function is to maximize the number of waypoints visited as denoted in (3). Constraint (4) ensures that waypoints are visited at most once. Constraint (5) ensures that each drone goes back to its depot. Constraint (6) states that the visited waypoints are within the radius r from the depot. Constraint (7) ensures the drone travels within its distance capacity. Constraint (8) sets the minimum number of waypoints for each drone to cover for efficient use, L. Constraint (9) eliminates the subtours for each drone. Constraint (10) ensures the paths are collision-free [16]. Here, $H_{ijk,pql}$ is a binary variable for collision check. $H_{ijk,pql} = 0$ indicates that there is no collision between edge (i, j) for drone k and path (p, q) for drone l. $H_{ijk,pql} = 1$ indicates that there is a collision and a disjoint of the intersected edges is needed. In (10), x_{ijk} represents the edge on a different altitude between edge (i, j) for drone k.

IV. Proposed Path Planning Algorithms

The ORBIT algorithm [25] uses a greedy approach to find the nearest neighbor to be included in the routes for every node. However, ORBIT neglects the risk of collision and therefore becomes collision prone. ORBIT follows similar constraints defined in Section III, except for Constraint 10. We use ORBIT as a baseline for our proposed collision-free path planning algorithms.

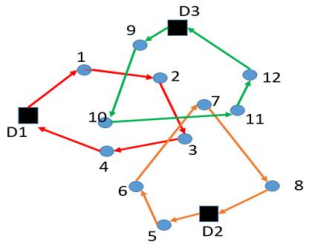


Fig. 1: ORBIT path planning example.

Fig.1 shows an example of path planning generated by ORBIT. Here, the three generated routes R_{D1} = $[D1 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow D1]$, R_{D2} = $[D2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow D2]$ and R_{D3} = $[D3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow D3]$ intersect with each other. Because ORBIT is agnostic to route collisions, it does not guarantee collision-free path planning. We use this case to demonstrate how XTRACT and 3DETACH remove the intersected routes in the following sections.

A. XTRACT ALGORITHM

XTRACT is a graph-based algorithm that removes the route intersections generated with ORBIT by sending one of the drones with the intersecting path to another altitude. Here, routes are interpreted as the edges of a polygon, and drones are assumed to initiate the flight at the same altitude. Whenever XTRACT detects an intersection among the routes generated by ORBIT, it sends one of the drones on the intersecting routes to another altitude or an elevated route. If the new route intersects with an already elevated route, the new route is discarded, leaving all the waypoints in the new route unvisited. We refer to the unvisited waypoints as orphan nodes in this paper.

Algorithm 1 describes the XTRACT algorithm. The input of XTRACT requires the list of the routes generated with ORBIT R, the coordinates of the depots and waypoints, the distance capacity of the drones Q, the minimum number of waypoints L, and the drone height H and returns the list of collision-free routes R_d as output. First, XTRACT calculates the distance matrix $\mathbf{M}$. Then, it performs sequential pairwise intersection checks among all the generated routes. For example, for a pair of routes (R_i and R_j), it checks for edge intersections between R_i and R_j. It adds any edge of R_j intersecting R_i to an array I_e, moves the drone in R_j to an altitude H, and then reduces the drone's total travel distance capacity by twice the travel altitude H. Once all the edges of R_j are checked for intersections with R_i, the total traveled distance of R_j is calculated. If the drone's distance capacity Q for R_j is satisfied, then R_j is stored in R_d. Otherwise, the last visited waypoint in R_j is removed. This process continues until the drone's distance capacity Q for R_j is satisfied.

If the intersecting edge of R_j intersects with any edge in the I_e array, then R_j is removed from the array and is no longer considered. Moreover, all the nodes in R_j are marked as orphans. On the contrary, if the intersecting edge of R_j does not intersect with any edge in the array, then R_j is kept in R_d. After all the routes have been considered for intersection checks, XTRACT returns the routes in R_d that meet the requirement for the minimum number of waypoints L.

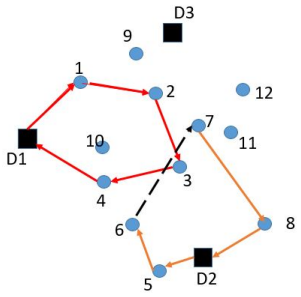


Fig. 2: XTRACT path planning example by removing intersected paths for D_3 in Figure 1.

Figure 2 shows an example of the path planning results using XTRACT for the path generated using ORBIT shown in Figure 1. The elevated edges are represented by the black dotted line in the figure. The path planning results using XTRACT contains R_{D1} and R_{D2}. XTRACT chooses to move the drone in R_{D2} to altitude H because the edge between waypoints 6 to 7 of R_{D2} insects R_{D1} at the edges between waypoints 3 to 4 and waypoints 2 to 3. XTRACT discards R_{D3} because this route intersects R_{D1} and R_{D2} and moving the drone in R_{D3} to avoid the insertion with R_{D1} would intersect the elevated edge of R_{D2}.

B. 3DETACH ALGORITHM

3DETACH algorithm is another graph-based 3D approach that removes the route intersections generated by ORBIT. Similar to XTRACT, in 3DETACH, routes are considered the edges of a polygon, and drones are initially considered to fly at the same height. A drone path is moved to another altitude to resolve any edge intersections between the generated routes. The difference between XTRACT and 3DETACH lies in the strategy to resolve the edge intersections, as described below.

Algorithm 2 describes the proposed 3DETACH algorithm. The input of the 3DETACH algorithm includes the coordinates of the depot and waypoints, the distance capacity Q, the minimum number of waypoints L, the drone height H, and the routes from the ORBIT algorithm. The output returns the collision-free routes in R_d. For every pair of routes (R_i and R_j), 3DETACH performs the edge intersection check between R_i and R_j. The intersecting edges of R_j are stored in I_e. The drone in R_j is chosen to be elevated for the intersection edges and the total travel distance capacity is reduced by two times the travel altitude H for every intersected edge. If the

Algorithm 1 XTRACT Algorithm

```
Obtain routes R from ORBIT
Input: D, V, R, height, L , Q
Output: R_d
for Sequential pair of R_i and R_j in R do
    I_e = []
    if R_i and R_j are intersected then
        Record the intersection edges in the array I_e from route R_j
        Calculate the totaldistance = totaldistance + 2H
        if totaldistance ≤ Q then
            Store the route in R_d
        else
            Remove the last visited waypoint from the R_j
        end if
    else
        Store the route in R_d
    end if
    if Edge of R_j intersects with elevated edge of R_i then
        Consider all the waypoints in route R_j as orphans
    end if
end for
for path in R_d do
    if length of path in R_d ≥ L then
        Store that path in R_d in R_d
    end if
end for
return R_d
```

total distance of R_j is less than Q, the last visited waypoint in R_j is removed and marked as an orphan. The process repeats until exhausting Q, after that R_j is stored in R_d. If the intersecting edge of R_j intersects with any of the edges in I_e, the conflicting edges of R_j are removed, and R_j is updated with the remaining waypoints in the route. if the travel distance of R_j is within capacity, R_j is kept in R_d. Finally, 3DETACH returns the routes in R_d that satisfies the minimum number of visited waypoints L.

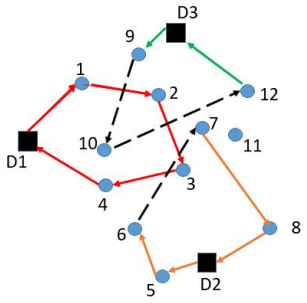


Fig. 3: 3DETACH path planning example.

Figure 3 shows the results of path planning using 3DETACH for the routes generated with ORBIT, as shown in Figure 1. For 3DETACH, the path planning result consists of three

routes $R_{D1} = [D1 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow D1]$, $R_{D2} = [D2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow D2]$, and $R_{D3} = [D3 \rightarrow 9 \rightarrow 10 \rightarrow 12 \rightarrow D3]$. 3DETACH chooses to elevate the drone in R_{D2} at the edge between waypoints 6 to 7 to resolve the intersections with R_{D1} and R_{D2} and elevate the drone in R_{D3} to resolve the two collisions with R_{D1}. However, elevating the drone in R_{D3} creates a collision with R_{D2} at the edge between waypoints 6 to 7. To remove the collision, 3DETACH removes the intersecting edges of R_{D3} with R_{D2} and updates the route of R_{D3} with the remaining waypoints in the route.

Algorithm 2 3DETACH Algorithm

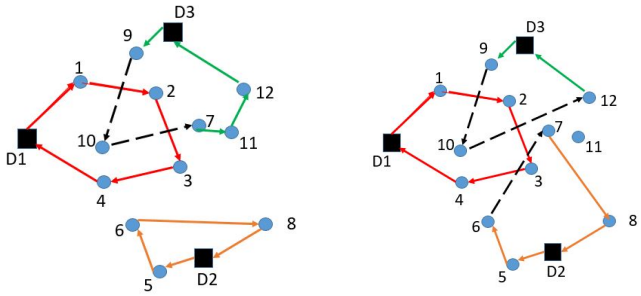
Obtain routes $\mathbf{R}$ from ORBIT
Input: D, V, $\mathbf{R}$, height, L , Q
Output: $\mathbf{R}_d$
for Sequential pair of R_i and R_j in $\mathbf{R}$ **do**
  I_e = []
  if R_i and R_j are intersected **then**
    Record the intersection edges in the array I_e from route R_j
    Calculate the totaldistance = totaldistance + 2H
    if totaldistance $\leq Q$ **then**
      Store the route in R_d
    else
      Remove the last visited waypoint from the R_j
    end if
  else
    Store the route in R_d
  end if
  if Edge of R_j intersects with elevated edge of R_i **then**
    Eliminate that waypoint from the route R_j and make that waypoint as orphan
  end if
end for
for path in R_d **do**
  if length of path in $R_d \geq L$ **then**
    Store that path in R_d in $\mathbf{R}_d$
  end if
end for
return $\mathbf{R}_d$

C. ASCEND ALGORITHM

ASCEND performs the same intersection checks as XTRACT and 3DETACH do, but it does not consider the routes generated by ORBIT as input. Instead, ASCEND checks for an intersection each time it adds a waypoint to the route. If there is an edge intersection, the other altitude is used to continue with the path. However, if the elevated edge intersects with another elevated edge, ASCEND looks for the next nearest neighbor from the last visited waypoint to continue the search.

Algorithm 3 describes the ASCEND algorithm. The input of ASCEND is the coordinates of the depot and waypoints, the drone's distance capacity Q, the radius r, the minimum number of waypoints L, and the altitude of the elevated edge H. The output is the array R_d with the collision-free routes. At first, ASCEND randomly chooses one of the depots, namely depot k, finds all the waypoints that are within a radius r, and stores the waypoints within r in T_k. It then checks if depot k has enough neighbors in T_k, i.e., $|T_k| \geq L$. Then, ASCEND checks if the traveled distance capacity Q for the obtained route is also satisfied. The waypoints within r for depot k are added sequentially as long as the capacity of the visiting drone is not exceeded. No intersection checks are required for the first route.

For the second route, a depot is chosen randomly from the remaining depots. The procedure is repeated as in the case of finding the first route. However, for the second and following routes, ASCEND performs intersection checks every time a new waypoint is added to the route and checks if the distance capacity constraint is satisfied. If it finds any edge intersection between the new route and any existing routes flying at the base altitude, then the drone is elevated at the intersecting edge. Moreover, if there is an intersection between the edges of a new route with the elevated edges, ASCEND will select the nearest waypoint that satisfies the distance capacity constraint and ensures no intersection with any elevated edge. The waypoint meeting those conditions is added to R_k. After no more routes can be formed, ASCEND returns the routes that meet the requirement of the minimum number of waypoints L from $\mathbf{R}_d$.



(a) Example 1 (b) Example 2

Fig. 4: ASCEND path planning examples.

Figure 4(a) shows one of the possible path planning results obtained using ASCEND for the example shown in Figure 1. In the figure, the dotted black represents the intersecting edges elevated in a route. In this example, it is considered that ASCEND randomly chooses the drone at depot $D1$ and that waypoints 1, 2, 3, 4, and 10 are within the radius r from $D1$. The waypoints within r for $D1$ are then stored in T_k. ASCEND finds that the nearest waypoint to $D1$ is waypoint 1 and adds it to the route after the distance capacity and intersection checks. The next step is to find the nearest waypoint from waypoint 1 that satisfies all the conditions. The process is repeated, and ASCEND finds for D1 the route $R_{D1} = [D1 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow D1]$. The next unvisited depot D3 is chosen. The same procedure is

followed for depot D3 as for D1. The route for depot D3 is $R_{D3} = [D3 \rightarrow 9 \rightarrow 10 \rightarrow 7 \rightarrow 11 \rightarrow 12 \rightarrow D3]$. The edge between waypoints 9 to 10 and the edge between waypoints 10 to 7 are stored in $Intersecting_{edge}$ because these edges intersect with the edges of R_{D1}. For $D2$, the route $R_{D2} = [D2 \rightarrow 5 \rightarrow 6 \rightarrow 8 \rightarrow D2]$ is found, and because it does not intersect with any of the edges of both R_{D1} and R_{D3}, no drone elevation is needed. ASCEND returns the three routes because all of them meet the requirement of the minimum number of waypoints L. Figure 4(b) shows that the path planning output that ASCEND obtains after $D2$ is chosen before D3. In this example, R_{D2} is elevated because there is an intersecting edge with R_{D1}.

Algorithm 3 ASCEND Algorithm

Input: D, V, Q, L, r, $\mathbf{M}$
Output: $\mathbf{R}_d$
for k in D **do**
  T_k = [ ]
  Add all the unvisited waypoints in T_k within coverage range r from k
  I_e = []
  if $|T_k| \geq L$ **then**
    n = k
    $d_{cum,k}$ = 0
    Find the minimum distance d_{min} from n to nearest waypoint (n_w) in T_k
    if $\overrightarrow{n, n_w}$ intersects with any edges in I_e **then**
      look for the next nearest waypoint n_w for n
    else
      Go to step 16
    end if
    if $\overrightarrow{n, n_w}$ intersects with any edges (i.e. not in I_e) in $\mathbf{R}_d$ **then**
      if n_w satisfies $d_{cum,k} + d_{min} + d_{n_w,k} \leq Q$ **then**
        Store the edge $\overrightarrow{n, n_w}$ in I_e
        Add n_w to R_k; mark n_w as unavailable
        $d_{cum,k} += d_{min}$;
        n = n_w
      else
        Discard n_w and set it as unavailable
      end if
    else
      Discard n_w and set it as unavailable
    end if
  end if
  if $|R_k| \geq L$ **then**
    Mark all waypoints in R_k as visited
    Add k to R_k, then add R_k to $\mathbf{R}_d$
  end if
end for

V. Simulation Results and Analysis

A. Simulation setup

We implemented the algorithms in C++ and evaluated their performance through computer simulation. The flight zone is set as a 4×4 km^2 area with 25 depots and different waypoint densities from 50 to 500 waypoints with an increment of 50 waypoints. The number of depots is determined by the upper bound of the number of depots needed to cover this area. The waypoints are randomly distributed inside the area. The locations of the depots are evenly distributed in the area and are fixed for all the experiments. Each depot is home to one drone, the algorithms randomly choose the depots from the given set of 25 depots to generate routes. We set the distance capacity of the drones to that of commercial ones, or 7 km. The range of the drone is a circle with a radius of 2 km, where the center of the circle is its depot. The minimum number of visited waypoints within each route is set to at least 3% of the total number of waypoints in the area. The simulation is run 10,000 times for each scenario. Each time, the depots are randomly selected with a randomly generated set of waypoints. The three algorithms were evaluated using the same topology. We record the mean and standard deviation of the number of orphan nodes, the number of drones used, and the cumulative route distance. We also use our previously proposed profit model to evaluate the cost-effectiveness of the three algorithms.

B. Performance evaluation

We compare the performance of the proposed 3D collision-free path planning algorithms in terms of the number of drones used, the number of waypoints covered, the number of orphan nodes, and the cumulative covered distance for a target area with waypoint densities that vary from 50 to 500 waypoints in the area. We use ORBIT as a baseline to compare the performance of our proposed algorithms. We also evaluate the profit ratio achieved with each algorithm to measure the cost-effectiveness of each approach. The profit ratio indicates the revenue generated from the number of covered waypoints, the cost of investment (i.e., drones), and usage, which is proportional to the drone covered distance.

Number of drones used. Figure 5 shows that the number of drones needed increases as the number of waypoints increases. Here, ASCEND requires the largest number of drones because it is optimized to maximize the number of visited waypoints in the presence of edge intersections with elevated routes. On the other hand, XTRACT requires the smallest number of drones because the algorithm discards entire routes whenever it finds a collision with an elevated route. Similar to XTRACT, 3DETACH detangles the routes generated by ORBIT with less restrictive conditions. Therefore, 3DETACH uses more drones to cover more waypoints than XTRACT.

Number of orphan nodes. Figure 6 shows the number of orphan nodes increases as the number of waypoints increases because high-density waypoints make it more difficult to find routes without collisions. ORBIT leaves out the smallest number of orphan nodes because it neglects intersection checks

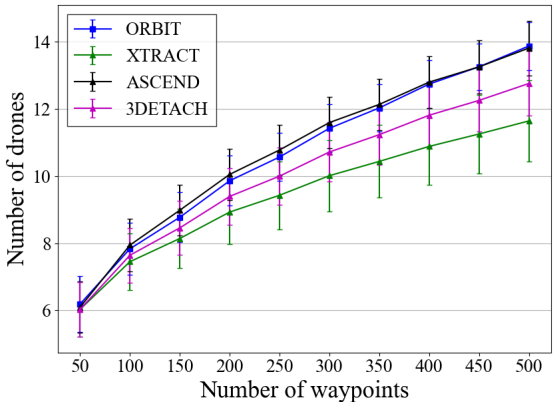


Fig. 5: Number of drones vs. number of waypoints.

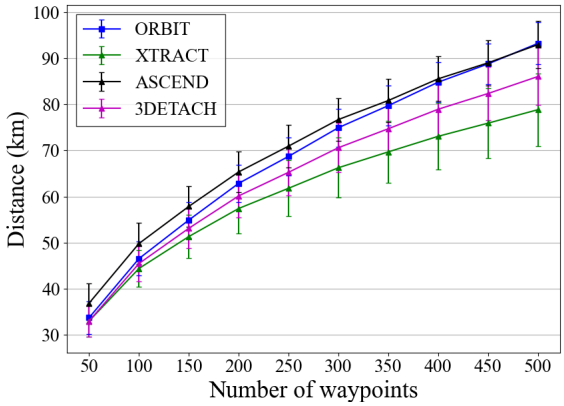


Fig. 7: Cumulative distance vs. number of waypoints.

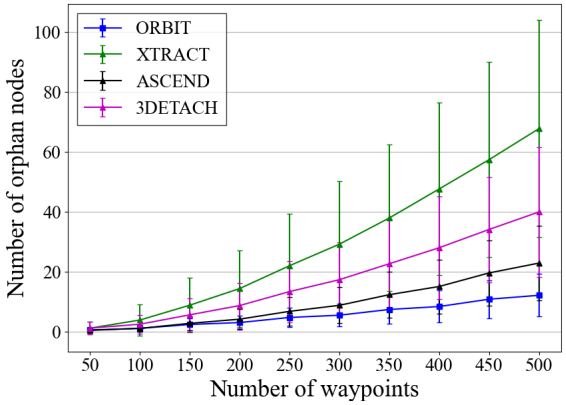


Fig. 6: Number of orphans vs. number of waypoints.

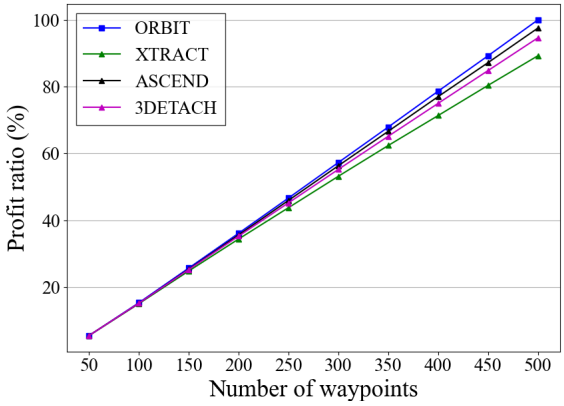


Fig. 8: Profit ratio vs. number of waypoints.

between the generated routes, which results in more visited waypoints but with colliding routes. ASCEND generates fewer orphan nodes than XTRACT and 3DETACH because it chooses alternative collision-free routes at the planning phase upon detection of route intersection and provides a more efficient coverage. This strategy reduces the number of orphan waypoints. XTRACT leaves more orphan nodes than 3DETACH because its strategy to resolve route collisions is more restrictive than that of 3DETACH. With a density of 500 waypoints in the area, ASCEND leaves approximately 60% and 50% fewer orphan nodes than XTRACT and 3DETACH, respectively.

Cumulative route distance. Figure 7 shows the cumulative travel distance for the path planning of each considered approach as the number of waypoints increases. Again, ASCEND outperforms XTRACT and 3DETACH with the longest cumulative travel distance with XTRACT covering the least distance due to its restrictive route selection.

Profit ratio. To make a fair and complete comparison between the profit of each approach considering the cost and maintenance of drones, and revenue from the number of covered waypoints, we use the profit model proposed in [25]:

$$P = N_{waypoint} \cdot \gamma - D \cdot \epsilon_{distance} - N_{drone} \cdot \epsilon_{drone} \quad (11)$$

Here, $N_{waypoint}$ denotes the number of visited waypoints, γ is the reward for each visited waypoint, D is the total distance, after subtracting the travel altitude H in the case of the elevated routes, $\epsilon_{distance}$ is the unit cost for the distance covered by the drone, and N_{drone} is the number of drones used, each with a unit price ϵ_{drone}. We assume the initial unit costs, for example, $\gamma = 50$, $\epsilon_{distance} = 5$, and $\epsilon_{drone} = 185$. The profit ratio is calculated as the profit achieved with each approach divided by the maximum profit of ORBIT for 500 waypoints.

Figure 8 shows the profit ratio as a function of the number of waypoints. It is clear that the profit ratio of ASCEND is the closest to that of ORBIT, which indicates its effectiveness in finding routes with a large number of visited nodes as ORBIT but without route collisions. XTRACT and 3DETACH achieve a smaller profit ratio than ASCEND and ORBIT because they visit fewer waypoints, translating to a lower achievable profit.

VI. Conclusions

In this paper, we tackle the collision-avoidance multi-drone path planning problem in a 3D space. We formulate the problem as a multi-depot vehicle routing problem and proposed three heuristic algorithms, XTRACT, 3DETACH, and ASCEND. We compared the performance of these three algorithms with the collision-agnostic ORBIT algorithm. XTRACT and 3DETACH take the path generated by ORBIT and remove the intersecting paths by taking one of the paths at a different altitude. ASCEND plans the path by selecting the nearest neighbors with a non-intersecting path. We consider only two altitudes in this paper to compare the collision prevention and removal approaches with minimal height complexity. We evaluate the three proposed algorithms through computer simulation and the results show that ASCEND not only covers as many waypoints as ORBIT but also generates 60% fewer orphan nodes than XTRACT under a high-density waypoint scenario in an area with 500 waypoints. In general, ASCEND outperforms XTRACT and 3DETACH and also achieves higher cost-effectiveness in the long term. These results suggest that collision prevention is more effective than collision removal.

Acknowledgment

This material is based upon work supported by the United States National Science Foundation under Grant No. 1841558 and NJIT Faculty Seed Grant Award 2023.

References

[1] L. Ding, D. Zhao, H. Ma, H. Wang, and L. Liu, "Energy-Efficient Min-Max Planning Of Heterogeneous Tasks with Multiple UAVs," in *2018 IEEE 24th International Conference On Parallel And Distributed Systems (ICPADS)*, 2018, pp. 339–346.

[2] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "UB-ANC planner: Energy efficient Coverage Path Planning With Multiple Drones," in *2017 IEEE International Conference On Robotics And Automation (ICRA)*. IEEE, 2017, pp. 6182–6189.

[3] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle Routing Problems For Drone Delivery," *IEEE Transactions On Systems, Man, And Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2017.

[4] J. Qiu, D. Grace, G. Ding, M. D. Zakaria, and Q. Wu, "Air-ground heterogeneous networks for 5G and beyond via integrating high and low altitude platforms," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 140–148, 2019.

[5] A. Saif, K. Dimyati, K. A. Noordin, S. H. Alsamhi, and A. Hawbani, "Multi-UAV and SAR collaboration model for disaster management in B5G networks," *Internet Technology Letters*, p. e310, 2021.

[6] S. H. Alsamhi, F. Almalki, O. Ma, M. S. Ansari, and B. Lee, "Predictive Estimation of Optimal Signal Strength from Drones over IoT Frameworks in Smart Cities," *IEEE Transactions on Mobile Computing*, 2021.

[7] M. Kothari, I. Postlethwaite, and D.-W. Gu, "Multi-UAV Path Planning in Obstacle Rich Environments Using Rapidly-exploring Random Trees," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 3069–3074.

[8] P. K. Esubonteng and R. Rojas-Cessa, "RESTORE: Low-Energy Drone-Assisted NLoS-FSO Emergency Communications," *IEEE Access*, vol. 10, pp. 115 282–115 294, 2022.

[9] B. Zhang, W. Liu, Z. Mao, J. Liu, and L. Shen, "Cooperative And Geometric Learning Algorithm (CGLA) For Path Planning of UAVs With Limited Information," *Automatica*, vol. 50, no. 3, pp. 809–820, 2014.

[10] L. Mejias, S. McNamara, J. Lai, and J. Ford, "Vision-Based Detection And Tracking Of Aerial Targets For UAV Collision Avoidance," in *2010 IEEE/RSJ International Conference On Intelligent Robots And Systems*. IEEE, 2010, pp. 87–92.

[11] Y. K. Kwag and C. H. Chung, "UAV Based Collision Avoidance Radar Sensor," in *2007 IEEE International Geoscience And Remote Sensing Symposium*. IEEE, 2007, pp. 639–642.

[12] K.-Y. Kim, J.-W. Park, and M.-J. Tahk, "UAV Collision Avoidance Using Probabilistic Method In 3-D," in *2007 International Conference On Control, Automation And Systems*. IEEE, 2007, pp. 826–829.

[13] C. Xiong, D. Chen, D. Lu, Z. Zeng, and L. Lian, "Path Planning Of Multiple Autonomous Marine Vehicles For Adaptive Sampling Using Voronoi-Based Ant Colony Optimization," *Robotics And Autonomous Systems*, vol. 115, pp. 90–103, 2019.

[14] G.-Z. Tan, H. Huan, and A. Sloman, "Ant Colony System Algorithm For Real-time Globally Optimal Path Planning Of Mobile Robots," *Acta Automatica Sinica*, vol. 33, no. 3, pp. 279–285, 2007.

[15] Y. Lin and S. Saripalli, "Sampling-based Path Planning For UAV Collision Avoidance," *IEEE Transactions On Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[16] S. Ahmed, A. Mohamed, K. Harras, M. Kholief, and S. Mesbah, "Energy Efficient Path Planning Techniques For UAV-based Systems With Space Discretization," in *2016 IEEE Wireless Communications And Networking Conference*. IEEE, 2016, pp. 1–6.

[17] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path Planning For Mobile Robot Navigation Using Voronoi Diagram And Fast Marching," in *2006 IEEE/RSJ International Conference On Intelligent Robots And Systems*. IEEE, 2006, pp. 2376–2381.

[18] P. Bhattacharya and M. L. Gavrilova, "Voronoi Diagram in Optimal Path Planning," in *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*. IEEE, 2007, pp. 38–47.

[19] Z. Li, Z. Zhang, H. Liu, and L. Yang, "A New Path Planning Method Based On Concave Polygon Convex Decomposition And Artificial Bee Colony Algorithm," *International Journal Of Advanced Robotic Systems*, vol. 17, no. 1, p. 1729881419894787, 2020.

[20] H. I. Kang, B. Lee, and K. Kim, "Path Planning Algorithm Using The Particle Swarm Optimization And The Improved Dijkstra Algorithm," in *2008 IEEE Pacific-Asia Workshop On Computational Intelligence And Industrial Application*, vol. 2. IEEE, 2008, pp. 1002–1004.

[21] Y. Liang and L. Xu, "Global Path Planning For Mobile Robot Based Genetic Algorithm And Modified Simulated Annealing Algorithm," in *Proceedings Of The First ACM/SIGEVO Summit On Genetic And Evolutionary Computation*, 2009, pp. 303–308.

[22] M. A. Mohammed, M. S. Ahmad, and S. A. Mostafa, "Using Genetic Algorithm In Implementing Capacitated Vehicle Routing Problem," in *2012 International Conference On Computer Information Science (ICCIS)*, vol. 1, 2012, pp. 257–262.

[23] J.-W. Park, H.-D. Oh, and M.-J. Tahk, "UAV Collision Avoidance Based On Geometric Approach," in *2008 SICE Annual Conference*. IEEE, 2008, pp. 2122–2126.

[24] L. Yang, X. Zhang, Y. Zhang, and G. Xiangmin, "Collision Free 4D Path Planning For Multiple UAVs Based On Spatial Refined Voting Mechanism And PSO Approach," *Chinese Journal Of Aeronautics*, vol. 32, no. 6, pp. 1504–1519, 2019.

[25] K. Shen, R. Shivgan, J. Medina, Z. Dong, and R. Rojas-Cessa, "Multidepot Drone Path Planning With Collision Avoidance," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 297–16 307, 2022.