Strategic Deconfliction of Unmanned Aircraft Based on Hexagonal Tessellation and Integer Programming

Yanchao Liu* and Zhenyu Zhou†
Wayne State University, Detroit, Michigan 48201
Waseem Naqvi‡
Raytheon Technologies, Marlborough, Massachusetts 01752
and
Jun Chen§
San Diego State University, San Diego, California 92182

https://doi.org/10.2514/1.G007459

Unmanned aircraft systems service suppliers adhere to interoperability standards that require unmanned aircraft operators to submit an operational intent, which describes the planned flight path in four-dimensional space. To ensure fairness, the central database follows a first-come, first-served approach, accepting new operational intents as long as they do not conflict with any active ones. However, creating a viable operational intent is challenging due to moving obstacles. This paper introduces an innovative optimization-based procedure to automate the intent filing process. It utilizes a stacked hexagonal tessellation to model the airspace, offering adjustable granularity. Path finding is accomplished using integer programming on the hex grid. The integer program is solved on a grid canvas that includes only necessary cells, striking a balance between computational efficiency and optimality. Simulation experiments demonstrate the procedure's effectiveness in generating feasible trajectories, even in scenarios with dense, omnidirectional air traffic. This procedure has the potential to become the foundational software core for low-altitude air traffic management systems, providing strategic deconfliction and constraint management services.

Nomenclature

$\mathcal{A}$	=	$\{(c,h,t): c \in \mathcal{C}, h \in \mathcal{H}, t \in \mathcal{T}$, cell c's layer h is available at time $t\}$
$\mathcal{C}$	=	index set of cells in the planning space on the map/canvas
c^0	=	index of the trip's origin cell, $\in \mathcal{C}$
c^*	=	index of the trip's destination cell, $\in \mathcal{C}$
$\mathcal{H}$	=	ordered index set of permissible altitude layers, $\{1,\ldots,k\}$
h^0	=	index of the trip's origin altitude, $\in \mathcal{H}$
h^*	=	index of the trip's destination altitude, $\in \mathcal{C}$
lock	=	the number of layers of cells surrounding the flight path to occupy
$\mathcal{N}_c$	=	set of neighboring cells of cell c, $c \in \mathcal{C}$
robust	=	the number of traversal time periods to occupy a cell for
$\mathcal{S}$	=	ordered index set of body segments, $\{-b,\ldots,-1, 0, 1,\ldots,b\}$
$\mathcal{T}$	=	ordered index set of time intervals in the planning space, $\{1, \ldots, T\}$
thickness	=	the number of layers of cells surrounding the shortest path to use in canvas
$w_{c,h,t}$	=	1 if $(x_{0,c,h,t} = 1$ and $\exists c' \in \mathcal{N}_c : x_{0,c',h,t-1} = 1)$
$x_{s,c,h,t}$	=	1 if segment s occupies cell c's altitude layer h at time t, $s \in \mathcal{S}$, $(c,h,t) \in \mathcal{A}$
$y_{c,h,t}$	=	auxiliary variable to enforce altitude reservation rules
$z_{s,c,t}$	=	1 if segment s occupies cell c's any layer at time t

Received 4 January 2023; revision received 17 June 2023; accepted for publication 5 October 2023; published online 31 October 2023.

*Associate Professor, Department of Industrial and Systems Engineering; yanchaoliu@wayne.edu (Corresponding Author).

†Research Assistant, Department of Industrial and Systems Engineering; currently R&D Analyst III, GEICO; zhezhou@geico.com.

‡Chief Technology Officer, Unmanned Systems, Raytheon Intelligence & Space; waseem.naqvi@raytheon.com.

§Assistant Professor, Department of Aerospace Engineering; jun.chen@sdsu.edu. Senior Member AIAA.

I. Introduction

CIVILIAN use of unmanned aircraft systems (UASs) has experienced a remarkable surge in recent years, leading to a substantial increase in the number of unmanned aircraft (UA) operating within commercial fleets across diverse sectors. This influx of UAs inevitably presents a pressing concern: the imminent crowding of the navigable airspace. Consequently, the effective management and coordination of high-density, heterogeneous, and unmanned air traffic have emerged as a shared challenge for the global air traffic management (ATM) community.

By the current interoperability standard [1], before each flight, the UAS operator in command (or its surrogate) needs to file a flight plan that includes a proposed departure time, a coarsely defined horizontal flight route (e.g., line segments connecting a series of navigation points), an intended altitude level, and an intended ground speed (as measured by GPS). The flight plan is then mapped to a geometric object in four-dimensional (4D) space. Such a flight plan, along with its 4D description, is called an operational intent (OI). A UAS service supplier (USS) that performs the strategic deconfliction service will then check if the current OI causes any conflict with other OIs already accepted in the system. If a conflict is detected, the OI cannot be accepted, and USS may optionally suggest some modifications to the proposed OI to restore its feasibility. It is our assumption that in transportation-purposed flights, only the origin and the destination waypoints, along with the intended departure time, are essential inputs, while the operator (increasingly likely to be a computer) is relatively unconcerned about the actual flight trajectory as long as it is safe and not overly inefficient (such as having long in-air holding or detours). In this case, the USS will be charged to generate a feasible trajectory for the operation based on the air traffic conditions at the time. The USS providing this service must perform two essential functions: 1) maintain a database that stores the 4D trajectory information of all accepted OI, and 2) plan conflict-free and efficient trajectories for new flight intentions based on relevant airspace constraint data sourced from the database.

In this paper, we present a new method for describing 4D volumes of trajectory-based OIs by leveraging a widely adopted, open-source geospatial indexing system with strong community support; therefore, our approach offers flexibility, scalability, and computational efficiency. Based on this, we introduce a mathematical optimization model that automates trajectory computation. Our optimization model utilizes well-established techniques in integer programming and benefits from a wide array of open-source and commercial solvers. With these advancements, we confidently assert that our proposed work can serve as the robust software core of a USS, delivering essential services for strategic deconfliction and constraint management.

Following a review of the UAS Traffic Management (UTM) development and the related literature in Sec. III.B, the data structure, optimization model, and solution processes will be described in Sec. III.C. Section IV presents numerical experiments and simulation analyses to validate the proposed system. Additional feasibility considerations are discussed in Sec. V. Section VI concludes the paper with pointers to follow-on research and development efforts.

II. Background and Related Literature

The UAS traffic management (UTM) concept was first conceived around 2015 to support the organization, coordination, and management of low-altitude UA operations, including beyond visual line-of-sight (BVLOS) operations. The UTM Concept of Operation (ConOps), developed by NASA researchers, proposes to integrate UAS into the national airspace system through fostering an innovative and competitive market of UAS service suppliers (USS) [2]. Subsequently, the UTM framework developed by the International Civil Aviation Organization (ICAO) suggested a number of services that are essential for the UTM ConOps, including activity reporting, airspace authorization, discovery, mapping, registration, restriction management, flight planning, conflict management and separation, identification, tracking and location, and meteorological services [3]. In the past few years, various new technologies have been developed, and existing technologies have been adapted and repurposed, to cater for the UTM ConOps.

The most important function of air traffic management is to ensure the safe separation of all aircraft while maximizing the overall traffic efficiency. To achieve effective separation, ICAO [4] adopts a three-layer scheme for conflict management: 1) strategic conflict management through airspace organization and management, demand and capacity balancing, and traffic synchronization; 2) separation provision, the tactical process of keeping aircraft away from hazards by at least the appropriate separation minima; and 3) collision avoidance, enabled by the traffic alert and collision avoidance system (TCAS) as a last defense against midair collisions. The corresponding three layers of traffic management in the UTM domain are called strategic deconfliction, tactical deconfliction, and detect and avoid (DAA). *Strategic deconfliction* deals with resolving a predicted conflict before departure. ICAO defines strategic deconfliction as a service consisting of the arrangement, negotiation, and prioritization of intended operational volumes, routes, or trajectories of UAS operations to minimize the likelihood of airborne conflicts between operations [3]. *Tactical deconfliction* deals with resolving impending conflicts among airborne aircraft using real-time information such as current location, heading, and speed. Such information may be available through the radar systems of the air traffic controller or through telemetry data sharing among UAS operators. DAA is an onboard capability (assuming no air traffic control separation services are provided to the UA) to avoid imminent midair collisions with objects in close proximity.

Most operations research on aircraft collision avoidance attempted to address the tactical deconfliction problem. A plethora of methodologies have emerged in the literature, including game theory and reinforcement learning [5], mixed integer programming [6], quadratically constrained quadratic programming [7], nonlinear programming [8,9], Markov decision processes [10], and rule-based approaches [11]. A recent survey of collision avoidance research can be found in [12].

Within the domain of strategic deconfliction, recent research has devoted considerable attention to the exploration and evaluation of various architectures and system designs. For instance, Jang et al. [13] conceived a lane-based system for UAS traffic in urban areas, in which all UA are assumed to follow predefined sky lanes (that weave through high-rise building blocks) and maintain a sufficient following distance to avoid crashes. Russell et al. [14] employed the autorouter idea in the design of multilayer printed circuit boards (PCB) to plan deconflicted flight paths for dense air traffic in urban environments. A hybrid search technique combining grid search (exact but slow) and probe (inexact but fast) was used to identify viable 2D paths, and heuristic treatments were applied to deconflict crossing and colinear paths. Chin et al. [15] suggested that not all operators would be willing to share the flight intent information due to privacy concerns, and hence congestion management mechanisms should handle scenarios with limited information sharing. The authors then proposed a rules-of-the-road approach for airspace access to address such challenges. Yang and Wei [16] formulated a problem that allocated more aircraft to a given airspace volume under strategic deconfliction using a multi-agent Markov decision process and solving with a Monte Carlo tree search algorithm. Egorov et al. [17] used high-fidelity simulation in combination with a collection of UTM services to evaluate the functional and performance requirements for strategic deconfliction. Despite the advances in research, a unanimous consensus regarding the optimal approach in practical applications has yet to be reached.

Of particular relevance to our paper is the Standard Specification for UTM USS Interoperability [1] (referred to as the Standard Specification hereafter) put forward by the F38.02 Flight Operations subcommittee in 2021. The Standard Specification addresses the performance and interoperability requirements for a set of UTM roles performed by USSs, including strategic coordination, conformance monitoring, and constraint management and processing. Regarding strategic deconfliction, the Standard Specification suggests, in simplified terms,[¶] OIs be addressed by the responsible USS in a one-by-one, first-come, first-served manner. Even so, finding a feasible 4D trajectory in the presence of other prescheduled trajectories is still a challenging task, which this paper aims to address.

III. Method

A. Definitions

Relevant terminology definitions are transcribed from the Standard Specification [1]. A *3D volume* is a volume of airspace defined in terms of latitude, longitude, and altitude; a *4D volume* is a 3D volume plus a start and end time for the volume; an *operational intent* (*OI*) is a volume-based representation of the intent for a UAS operation. An OI comprises one or more overlapping or contiguous 4D volumes, where the start time for each volume is the earliest entry time and the stop time for each volume is the latest exit time. A trajectory-based OI consists of a series of volumes that follow the desired flight path and overlap in space and time. A *conflict* is a situation where two OIs intersect both in space and time. To intersect in space and time, at least one constituent 3D volume of an OI must share at least one point with a 3D volume of another OI, and there must be an intersection between the start/end time ranges for those two volumes.

B. Hexagonal Grid Model

We divide the navigable airspace into different layers by altitude and partition each layer into hexagonal cells of equal size. We model the trajectory-based OI as a reservation of a selected subset of the cells, with each cell being reserved for a period (or multiple disjunctive periods) of time marked by a (set of) start and end time points. Cells that cover the intended flight path should be reserved, and the start and end time of a cell's reservation should cover the time period that the aircraft is expected to be (i.e., passing through or lingering) in

[¶]More detailed specifications governing how a USS should create and update an OI can be found in Sec. 5.4.2 of the Standard Specification [1].

the cell. If the operation involves visiting the same location multiple times, such as in a looping trajectory, then the same cell can be reserved multiple times, each with a separate pair of start and end time designations.

Errors can arise from various sources, including path definition, georeferencing, flight management systems, altitude and positioning systems, remote pilot proficiency, departure timing, and weather conditions. Along the intended flight path, the aircraft's locational uncertainty (in the 4D sense) is much more pronounced in the longitudinal direction than in the lateral direction, because inaccuracies in speed and departure timing primarily affect the longitudinal direction. The boundaries of the reserved 4D volume must be constructed to buffer the intended operation.

To account for the longitudinal uncertainty, cells along the flight path can be reserved for a longer period than would be necessary under the perfect condition. A straightforward approach is to start the cell reservation a number of time intervals earlier and end the cell reservation a number of time intervals later than the expected entry and exit time points of the cell. See Fig. 1 for a demonstration. In the figure, the aircraft intends to pass through four adjacent cells labeled 1–4. Since the aircraft will be in one cell at any moment, the most frugal reservation is to reserve the four cells in the exact timing sequence; e.g., the reservation of cell 2 starts at the same time as the reservation of cell 1 ends. However, conformance to such an OI could be a problem in the actual operation, since there is no room for timing errors. A more robust reservation is shown in the lower part of the figure, in which the reserved time period for each cell is extended to start before the expected entry time and to end after the expected exit time. At any moment, multiple contiguous cells may be reserved, which effectively hedges against operational uncertainty. The *robust* value shown in Fig. 1 is equal to $|\mathcal{S}|$, or $b + 1$, as defined in the Nomenclature section and further explained in Sec. III.G.

C. Lateral Separation Bounds

In theory, the aircraft is allowed to be anywhere within the reserved cell (including at the cell's boundary) during the reserved time period. However, if two aircraft's reserved cells overlap in time and have a common boundary, the separation between the two aircraft cannot be ensured. This issue can be addressed in two ways.

The first method is to require that the aircraft navigate along the centerline of the reserved corridor. That is, the nominal flight path should follow the line segments connecting the center points of successive reserved cells (as shown in the blue arrow segments in Fig. 2). In this way, the separation buffer between any two aircraft is at least the width of the cell. The problem with this method is also obvious: 1) the reconstructed flight path may contain many unnecessary navigation (direction adjustment) points; 2) the trip's origin and destination points are unlikely to fall on the center points of the corresponding cells, which means that during takeoff and landing, the aircraft may have insufficient separation from some other aircraft operating in an adjacent cell.

The second method is to, in addition to reserving the cells the aircraft intends to pass through, reserve all the neighboring cells as well. See Fig. 2 for an illustration: the blue cells (lock = 1) encompass the intended flight path, while the yellow cells (lock = 2) provide extra buffer for lateral separation. In this way, as long as the actual flight path remains within the area covered by the blue cells, the aircraft's distance to the boundary of its reserved operational area is no smaller than the edge length of a cell. The red curve in the figure

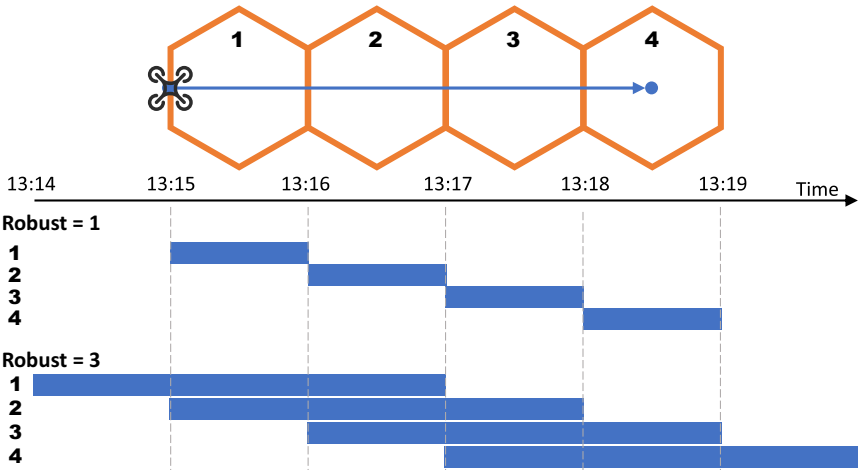

Fig. 1 Cell-based OI representation.

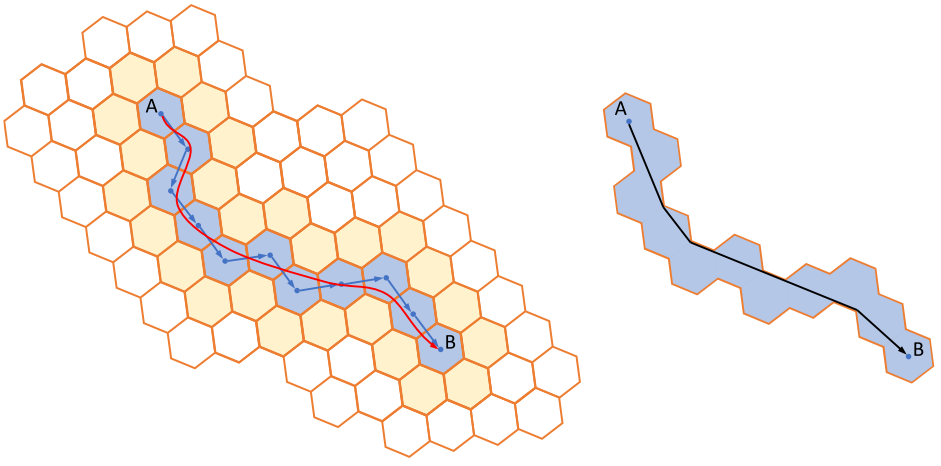

Fig. 2 Cell reservation schemes.

illustrates a smooth flight path of this type. Via postprocessing, the shortest path between the origin and destination can be easily constructed within the reserved operational area, as illustrated in the right part of Fig. 2.

Note that Fig. 2 is an aggregated spatial view over the entire time span of the OI, i.e., the flight from point A to point B. The temporal reservation of the cells will evolve along the timeline in a way that is depicted in the lower part of Fig. 1. This explains why the flight path (the string of blue cells) is not a near-straight corridor from A to B—the 4D cell reservation of other OIs is assumed to have prevented the aircraft from flying along the straight line.

D. Cell Size, Speed, and Time Interval

Let us denote the edge length of a hexagonal cell by a, then the distance an aircraft needs to travel to traverse the cell (i.e., the distance between the entry point and the exit point) is bounded in this interval: $[a, 2a]$. The lower bound is due to the absence of a knot in the trajectory; i.e., in order to go from the last visited cell to the next cell, the aircraft must at least travel an edge-long distance to traverse the current cell, since there is no shorter path between the last cell and the next cell. The upper bound is simply the maximum distance between two points in the same cell. In practice, there is no incentive to traverse a cell along a longer path than necessary, so the distance between a pair of parallel edges, or equivalently the center-to-center distance between two adjacent cells, is representative of the nominal distance needed to traverse a cell. This distance is $\sqrt{3}a$ and will be used in travel time calculations.

While all OIs use the same discretized geospatial indexing system for location referencing, the temporal dimension remains continuous, meaning that the reservation start and end time of a cell can be placed anywhere along the timeline, not restrained to a discrete set of time points. In planning the 4D reservation for an OI, we propose an integer programming model, where time is discretized into equal-length intervals, with the interval length representing the expected time taken to traverse a cell. Specifically, the interval length is equal to the nominal cell size, $\sqrt{3}a$, divided by the ground speed of the aircraft. In this way, a binary state can be associated with each 4D voxel (i.e., a hexagonal cell at a particular altitude layer during a particular time interval), indicating whether the voxel is reserved or not.

E. Vertical Separation

To avoid trespassing unavailable airspace voxels in the flight path, an aircraft has three options: change speed, change direction, and change altitude. In practice, the vertical separation buffer is much smaller than the horizontal separation buffer, e.g., 1000 feet vertical versus (up to) 5 nautical miles horizontal in certain air traffic control scenarios [18]. In low-altitude airspace, a similar contrast between horizontal and vertical separation distances is likely to apply. Therefore, for lightweight UA, changing altitude is an easy and efficient maneuver and should be leveraged in OI planning whenever feasible. We consider this an advancement in comparison to our earlier work, in which altitude changes were omitted [8,9].

The navigable airspace is divided into N vertical layers, and the reservation of each geospatial cell is associated with an altitude index to indicate which vertical layer is being reserved. As with the geospatial indexing system used for partitioning the Earth surface, the vertical division scheme must be agreed upon by all users of the shared airspace. In mathematical modeling, only the layer indices are essential, whereas specific details regarding altitude divisions, including layer boundary values and thicknesses, are unnecessary for the purposes of modeling and computation.

We assume that altitude changes happen simultaneously along with the aircraft's traversal of a reserved cell in the horizontal dimension and do not take a substantial amount of additional time beyond the time taken for the horizontal movement. If a change in altitude is planned within a cell, all altitude layers between the starting and ending altitude layers must be available and reserved for the cell. This is illustrated in Fig. 3.

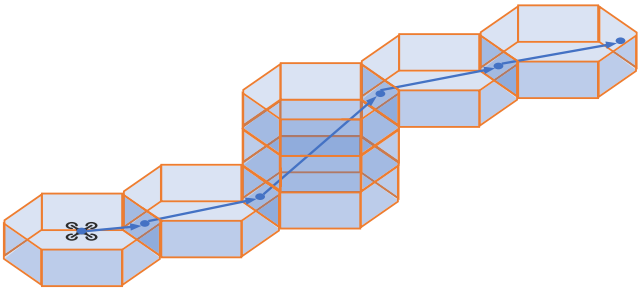

Fig. 3 Altitude reservations.

In OI planning, the permissible altitude range (consisting of a contiguous block of altitude layers) is provided as input, and the path planning process will only consider the altitudes in range. For a transportation-purposed OI, the desired altitude is usually the lowest layer within the permissible range. Altitude changes are mainly for deconfliction with other prescheduled OIs. The greater the range of altitudes allowed for a trip, the simpler it becomes to schedule a direct path (viewed from the top) between the starting point and the destination. On the other hand, frequent changes in altitude should be avoided, as it would fragment the navigable space for other operations that might be scheduled in the same temporal and spatial region. Spurious altitude consumption should be properly penalized to ensure efficiency.

F. OI Filing Process

An OI based on trajectory can be broken down into a series of waypoints. These waypoints can be categorized into two types based on their importance for the intended operation: essential waypoints and unessential waypoints. For instance, in a transportation-purposed flight, only the origin and the destination are essential waypoints; in a reconnaissance task, all target locations to be surveyed are essential waypoints, though their visitation sequence may be up for optimization. If there is no a free-flight path between two successive essential waypoints, additional waypoints may be added to the flight plan to aid navigation. We call the latter type *navigation waypoints*, which are deemed nonessential because there are numerous ways in which such waypoints can be added to achieve a feasible outcome, and minimizing their usage is usually desired. Strategic deconfliction aims to plan all navigation waypoints ahead of the flight, while tactical deconfliction calculates the navigation waypoints progressively on-the-fly. Traditional air traffic control systems depend on the attention and judgment of human controllers to manage air traffic. Additionally, they rely on human pilots to verbally receive and execute navigation commands from the controllers. However, these mechanisms are constrained by the cognitive limitations of humans.

When dealing with an operation that involves visiting multiple essential waypoints, the planning problem can be divided into two hierarchical levels. The first level focuses on optimizing the sequence in which the essential waypoints are visited. This optimization can be formulated as a shortest path problem, traveling salesman problem, vehicle routing problem, or orienteering problem, depending on the specific flight mission. At this level, factors such as the air traffic condition, including rapidly changing variables like the flight trajectories of other operations, are disregarded. Once the first-level decision is made, the second level comes into play. The objective of the second level is to determine a *feasible* and efficient flight trajectory between each successive pair of essential waypoints. Here, feasible means that the trajectory is achievable within the capability of the aircraft, does not conflict with any other OI's trajectory, and does not violate any airspace restriction, e.g., no-fly zones, convective weather zones, and altitude limitations. This is accomplished by taking into account the dynamic air traffic condition in the 4D space. In some cases, the second-level problem may not have a feasible solution. When this occurs, it becomes necessary to adjust the first-level decision, deviating from its initial optimality, in order to enable a feasible solution at the second level.

The proposed integer program solves the second-level problem, i.e., planning the 4D trajectory of a flight between two essential waypoints. User-supplied inputs to the problem consist of 1) a pair of origin and destination points specified by their 3D coordinates, $(\text{Lat}_o, \text{Lon}_o, \text{Alt}_o)$ and $(\text{Lat}_d, \text{Lon}_d, \text{Alt}_d)$; 2) the desired start time of the flight, denoted by t^0; 3) the altitude range in which the navigation waypoints can be located, $[\text{Alt}^{\min}, \text{Alt}^{\max}]$, determined by the aircraft capability or by mission requirements; 4) the horizontal cruising speed (ground speed), v^H, and the maximum climb and descent rates, v^V, of the aircraft; and 5) the temporal buffer size, denoted by b, for the airspace volume reservation during the operation, specified in terms of the number of time intervals before (and symmetrically after) the time interval in which the aircraft is expected to be in a geospatial cell.

Given these inputs, the planning process proceeds as follows. The origin and destination cell indices c^0 and c^* are obtained via querying the global geospatial indexing system using the (Lat, Lon) coordinates of the trip's origin and destination, respectively. Adopting a predetermined altitude discretization scheme, the operating altitude range is mapped to the index set $\mathcal{H} \coloneqq \{1, \ldots, k\}$, where index 1 represents the lowest layer and the index k represents the highest layer in the user-supplied altitude range. The length of the unit time interval is calculated as the cell size (i.e., center-to-center distance) divided by the horizontal cruising speed of the aircraft, i.e., $\Delta t \coloneqq \sqrt{3}a/V^H$. An overestimate of the number of time intervals taken to complete the flight from the origin to the destination in heavy traffic is calculated as follows:

$$T = \beta \cdot \frac{D((\text{Lat}_o, \text{Lon}_o), (\text{Lat}_d, \text{Lon}_d))}{V^H} \tag{1}$$

where $D(o, d)$ is a function that returns the great circle distance between points o and d expressed in the (Lat, Lon) format, and the multiplier $\beta \geq 1$ accounts for the expected delay caused by traffic. The index set of time intervals $\mathcal{T}$ is therefore set as $\{1, \ldots, T\}$, whereas the starting point of the first time interval (i.e., $t = 1$) is aligned with the operation start time t^0. Here, each index represents a time interval of the same length and adjacent indices represent adjacent time intervals. For example, the end time point of the interval i is the start time point of the interval $i + 1$, for each $i \in \mathcal{T} \setminus \{T\}$. Now, the trip origin $(\text{Lat}_o, \text{Lon}_o, \text{Alt}_o, t^0)$ can be mapped to the 4D voxel indexed by $(c^0, h^0, 1)$. The set of cells $\mathcal{C}$ to be used in the model, called the *horizontal canvas*, can be instantiated with flexibility. At the minimum, the canvas should contain a path that links the origin cell c^0 and the destination cell c^*, and the canvas should span a single connected area. Even though the final flight trace (i.e., the optimal 4D trajectory projected onto the 2D plane) may not coincide with the 2D shortest path due to deconfliction with other OIs, it is recommended that the shortest path in 2D between c^0 and c^* is always included in the canvas to avoid prematurely excluding any optimal solution. In general, a bigger canvas provides a higher chance of finding feasible 4D trajectories and provides more room for optimization but, in the meantime, increases the computational burden.

G. Integer Programming Model

$$\text{Maximize} \quad \sum_{t \in \mathcal{T}} x_{0,c^*,h^*,t} - \alpha \sum_{c,h,t} x_{0,c,h,t} \tag{2}$$

$$\text{s.t.} \quad x_{s,c,h,t} = x_{s-1,c,h,t-1} \qquad \forall s \in \mathcal{S} \setminus \{-b\}, (c,h,t) \in \mathcal{A}, t \neq 1 \tag{3}$$

$$w_{c,h,t} \leq x_{0,c,h,t} \qquad \forall (c,h,t) \in \mathcal{A}, t \neq 1 \tag{4}$$

$$w_{c,h,t} \leq \sum_{c' \in \mathcal{N}_c} x_{0,c',h,t-1} \qquad \forall (c,h,t) \in \mathcal{A}, t \neq 1 \tag{5}$$

$$x_{0,c,h,t} \leq x_{0,c,h,t-1} + \sum_{c' \in \mathcal{N}_c} x_{0,c',h,t-1} + \sum_{h' \in \mathcal{H} \setminus \{h\}} w_{c,h',t} \qquad \forall (c,h,t) \in \mathcal{A}, t \neq 1 \tag{6}$$

$$z_{s,c,t} \leq \sum_{h \in \mathcal{H}} x_{s,c,h,t} \qquad \forall s \in \mathcal{S}, (c,t) \in \mathcal{A}' \tag{7}$$

$$z_{s,c,t} \geq x_{s,c,h,t} \qquad \forall s \in \mathcal{S}, (c,h,t) \in \mathcal{A} \tag{8}$$

$$\sum_{c \in \mathcal{C}} z_{s,c,t} = 1 \qquad \forall s \in \mathcal{S}, (c,t) \in \mathcal{A}', t \neq 1 \tag{9}$$

$$y_{c,h,t} \geq x_{0,c,h,t} - x_{0,c,h-1,t} \qquad \forall (c,h,t) \in \mathcal{A}, h \neq 1 \tag{10}$$

$$y_{c,h,t} \geq x_{0,c,h-1,t} - x_{0,c,h,t} \qquad \forall (c,h,t) \in \mathcal{A}, h \neq 1 \tag{11}$$

$$x_{0,c,1,t} + \sum_{h \in \mathcal{H} \setminus \{1\}} y_{c,h,t} \leq 2 \qquad \forall (c,t) \in \mathcal{A}' \tag{12}$$

$$x_{s,c,h,t} = 0 \qquad \forall s \in \mathcal{S}, (c,c',h,t) \in \mathcal{E} \tag{13}$$

$$x_{s,c^0,h^0,1} = 1 \qquad \forall s \in \mathcal{S} \tag{14}$$

$$x_{s,c,h,t} \in \{0,1\} \qquad \forall s \in \mathcal{S}, (c,h,t) \in \mathcal{A} \tag{15}$$

$$z_{s,c,t} \in \{0,1\} \qquad \forall s \in \mathcal{S}, (c,t) \in \mathcal{A}' \tag{16}$$

$$y_{c,h,t} \geq 0 \qquad \forall (c,h,t) \in \mathcal{A} \tag{17}$$

$$w_{c,h,t} \geq 0 \qquad \forall (c,h,t) \in \mathcal{A} \tag{18}$$

The objective (2) has two terms. The first term is to maximize the time the vehicle spends at its destination cell, i.e., to minimize the time it spends on the way. The second term minimizes the total size of the 4D volume reserved, which suppresses spurious reservation of altitude layers. The constraints are annotated below.

Equation (3) ensures that the vehicle occupies each cell in its path for a contiguous block of time intervals.

Equations (4) and (5) define the variable $w_{c,h,t}$, which indicates whether a 4D voxel (c, h, h) has the following properties in the solution: a) it is to be occupied by the aircraft, and b) one of its same-layer neighbors is also to be occupied by the aircraft. This variable is used for modeling the aircraft's motion while altitude adjustment is undertaken.

In Eq. (6), a 3D voxel (c, h) can be occupied at time t by the aircraft if one of the following three conditions is met: a) the aircraft was in the voxel at $t - 1$, b) the aircraft was in a neighboring voxel of the same layer at $t - 1$, and c) the aircraft was in a neighboring voxel of a different layer at $t - 1$ and will perform an altitude change into the current voxel. The three conditions correspond to the three terms, respectively, on the right-hand side of this constraint.

Equations (7) and (8) define the variable $z_{s,c,t}$, which models the whereabouts (in the 2D horizontal dimension) of the body segment s at time t. The term "body segment" refers to an occupied cell in the local reference frame, relative to the reference cell (with $s = 0$) that contains the location of the aircraft.

In Eq. (9), each body segment must be in one and only one cell at any time. Each c uniquely identifies a hexagonal geographic area, and the index s represents the position of a body segment relative to the whole body; $z_{s,c,t}$ links the two elements with time. Figure 4 gives an illustration.

Equations (10) and (11) define the variable $y_{c,h,t}$ as the absolute value of the difference between $x_{0,c,h,t}$ and $x_{0,c,h-1,t}$, used for modeling the altitude change rule.

In Eq. (12), at most one contiguous block of altitude layers can be reserved for the same horizontal cell in a time interval. In other words, all altitude layers between the start and the end layer must be reserved by the aircraft. This is illustrated in Fig. 3.

In Eq. (13), a cell cannot be taken if its neighbor is unavailable. The set $\mathcal{E}$ is defined as $\{(c, c', h, t) \in \mathcal{C}^2 \times \mathcal{H} \times \mathcal{T} : (c, h, t) \in \mathcal{A}, (c', h, t) \notin \mathcal{A}, c' \in \mathcal{N}_c\}$. This constraint implements the second method of ensuring lateral separation as discussed in Sec. III.C.

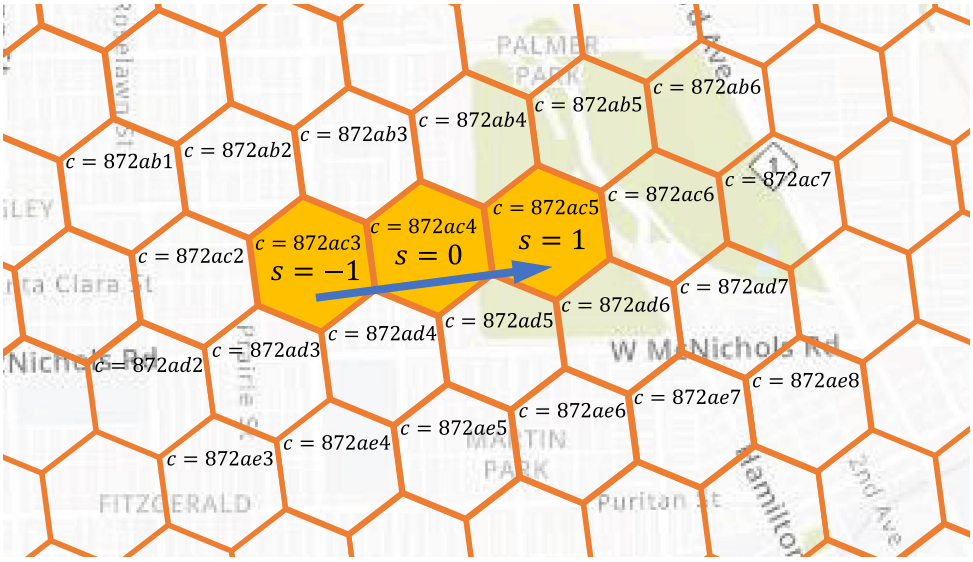

Fig. 4 Relationship between index of cells c and body segments s.

In Eq. (14), the vehicle, along with all body segments of its instantaneous trace, is at the trip's origin cell at $t = 1$.

Equations (15–18) are variable-type constraints. Note that the variable x is defined over $\{0, 1\}^{\mathcal{S}\times\mathcal{A}}$ instead of $\{0, 1\}^{\mathcal{S}\times\mathcal{C}\times\mathcal{T}}$, implying that only available 4D voxels can be reserved.

H. Flight Path Construction

In this step, a shortest path from the origin to the destination is constructed. In doing so, static obstacles such as no-fly zones (NFZ) are considered, while moving obstacles such as the flight traces of other OIs are ignored. If the straight-line path between the origin and the destination is blocked by some obstacle(s), a variety of 2D path planning algorithms can be applied to find the shortest path. For example, one can apply the A* algorithm on the visibility graph [19,20]; if the obstacles can be enveloped by a circular hull, nonlinear optimization approaches, non, [8,9], can be applied.

The length of the shortest path will serve as the congestion-free benchmark for the final OI, whose construction must take the traffic situation (other OIs) into account. In addition, volume cells that are passed through by the shortest path, as well as their adjacent cells, are highly probable to be utilized by the final OI. Therefore, the canvas $\mathcal{C}$ for the optimization model should include, and center around, those cells.

IV. Implementation and Experiments

In the software implementation, we used Uber's H3 hexagonal hierarchical geospatial indexing system. The system partitions the Earth's surface into hexagonal cells using central place indexing (CPI) [21]. Table 1 summarizes the H3 grid resolution.

The H3 core library is written in C, and its Application Programming Interface (API) is available for more than 20 programming languages and frameworks. In the computational experiments, we adopted H3 Resolution 7, which gives an average hexagon edge length of 1.22 km. As suggested in [1], implementations of the 4D volume reservation must balance between false conflicts that can arise from overly coarse volume descriptions and unnecessary computation that can result from overly granular characterization of OIs. The 16 levels of granularity provided by the H3 library can support such a tradeoff, and we will leave an in-depth analysis of efficiency versus safety for future work.

All numerical experiments and simulations were performed on a MacBook Pro with a 2.3 GHz 8-Core Intel Core i9 CPU and 16 GB of RAM. The MIP model instances were solved by the CPLEX solver (version 20.1) via the GAMS Python API (GAMS version 36.2.0). A simple user interface was programmed using Python 3.9 to facilitate customized experimentation. In all experiments, we set the robustness level to 3 (i.e., $b = 1$ in the definition of body segments $\mathcal{S}$), meaning that a string of 3 cells (one at, one before, and one after the expected whereabouts of the aircraft) is reserved at any time when the aircraft is in motion, as illustrated in Fig. 1.

A. Stylized Scenario for Congestion Tests

In this section, we validate the effectiveness of the proposed model using a stylized resource-constrained scenario. Specifically,

Table 1 H3 resolution summary

H3 resolution	Average hexagon area, km^2	Average hexagon edge length, km	No. of unique indexes
0	4,250,546.85	1,107.71	122
1	607,220.98	418.6760055	842
2	86,745.85	158.2446558	5,882
3	12,392.26	59.81085794	41,162
4	1,770.32	22.6063794	288,122
5	252.9033645	8.544408276	2,016,842
6	36.1290521	3.229482772	14,117,882
7	5.1612932	1.220629759	98,825,162
8	0.7373276	0.461354684	691,776,122
9	0.1053325	0.174375668	4,842,432,842
10	0.0150475	0.065907807	33,897,029,882
11	0.0021496	0.024910561	237,279,209,162
12	0.0003071	0.009415526	1,660,954,464,122
13	0.0000439	0.003559893	11,626,681,248,842
14	0.0000063	0.001348575	81,386,768,741,882
15	0.0000009	0.000509713	569,707,381,193,162

we generate six flight plans, all to start at the same time. The six flights are in three pairs, and within each pair, the origin of one flight is the destination of the other, and vice versa. Moreover, all the straight-line paths connecting each pair of origin and destination have a common intersection point, and so congestion is bound to occur. The flight plan data are given in Table 2 and demonstrated in Fig. 5.

Using the proposed method, we plan the 4D trajectories of the six flights sequentially from 1 to 6. After generating each plan, the volume reservation information of the OI is appended to a database, the entries of which will form the constraints for subsequent runs. We have experimented with opening one, two, and three altitude layers (i.e., $k = 1, 2, 3$ or $\mathcal{H} = \{1\}, \{1, 2\}, \{1, 2, 3\}$) for use for these flights, and experimented with two canvas size settings (i.e., thickness values). Specifically, thickness = 1 means to use the shortest path (in terms of a string of hex cells) between the origin and the destination as the 2D grid canvas, i.e., $\mathcal{C}$, for the MIP model, and thickness = 2 means to thicken the shortest path with an extra layer of hex cells on each side of the path and use the resulting set of cells as the grid canvas for optimization. The planning canvas (the grid of hollow red cells) shown on the right of Fig. 5 shows the set $\mathcal{C}$ with thickness = 2 used for planning the trajectory of Flight 5 (or 6).

The experiment results are shown in Table 3. For each flight, the following metrics are listed: the actual flight duration in minutes, the number of 4D voxels occupied in the operation, the number of times the aircraft has to move from one altitude layer to an adjacent layer (Alt Chg), and the solution time of the MIP model (CPU). We can see that, in such a resource-constrained situation, OIs that are scheduled late (i.e., latecomers) suffer more of the consequences of congestion in terms of increased flight duration, more 4D volume occupation, and more altitude change maneuvers. For instance, in the right subfigure of Fig. 5, which captures a temporal snapshot of the volume reservation, the slightly darker cell in the threesome reserved by flight No. 4 signals an altitude change (i.e., occupying more than one altitude layer in the same horizontal cell) at the moment. Such altitude changes can also be seen in flights No. 2 and No. 6, when they are trying to avoid a head-on collision with previously scheduled flights (i.e., No. 1 and No. 5, respectively) along the same flight paths. In the

Table 2 Input data of the stylized test case

No.	Origin latitude	Origin longitude	Destination latitude	Destination longitude	Distance, km	Speed, m/s	Duration, min
1	43.5346	−83.3883	43.1731	−82.9646	52.8	15	58.7
2	43.1731	−82.9646	43.5346	−83.3883	52.8	15	58.7
3	43.5744	−83.0127	43.1250	−83.2571	53.7	15	59.7
4	43.1250	−83.2571	43.5744	−83.0127	53.7	15	59.7
5	43.2892	−83.4851	43.3631	−82.8026	56.0	15	62.2
6	43.3631	−82.8026	43.2892	−83.4851	56.0	15	62.2

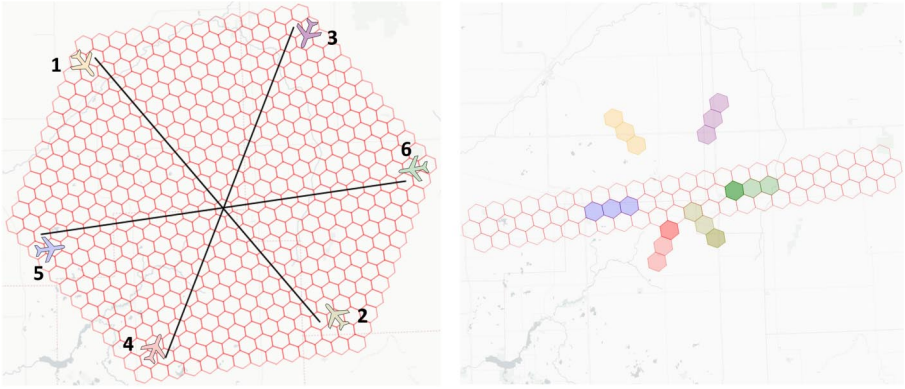

Fig. 5 Test area and OIs. Left: starting position of six flights. Right: a temporal snapshot of voxel reservations.

Table 3 Stylized test results with planning sequence 1–6

		Thickness = 1				Thickness = 2			
H	No.	Duration, min	4D volume	Altitude change	CPU	Duration, min	4D volume	Altitude change	CPU
$H = 1$	1	61.3	75	0	0.2	61.3	78	0	0.9
	2	— —	— —	— —	0.1	64.0	81	0	1.0
	3	72.4	87	0	0.2	72.4	90	0	1.1
	4	— —	— —	— —	0.1	78.0	96	0	1.1
	5	83.5	99	0	0.2	83.5	102	0	1.1
	6	— —	— —	— —	0.1	80.8	99	0	1.2
$H = 2$	1	61.3	75	0	0.8	61.3	78	0	4.2
	2	61.3	81	2	0.3	61.3	84	2	3.6
	3	64.0	84	2	0.9	64.0	93	4	5.4
	4	78.0	99	2	0.7	66.8	84	0	4.2
	5	75.2	96	2	0.7	72.4	96	2	6.2
	6	83.5	105	2	0.4	75.2	93	0	4.0
$H = 3$	1	61.3	75	0	1.0	61.3	78	0	21.7
	2	61.3	81	2	1.3	61.3	84	2	22.6
	3	64.0	84	2	1.3	64.0	87	2	26.7
	4	64.0	90	4	1.3	64.0	93	4	9.0
	5	69.6	96	4	1.4	69.6	99	4	42.0
	6	75.2	90	0	1.1	69.6	99	4	24.2

more extreme case, when all flights are confined to the same altitude layer ($H = 1$) and no horizontal leeway is permitted (thickness = 1), the latecomer in each conflicting pair, e.g., flight No. 2 in the pair (1, 2), cannot even get a feasible trajectory. Such an infeasible situation is alleviated either by opening more altitude layers or by planning a bigger grid canvas (i.e., increasing thickness), but both methods would result in a larger problem instance with increased computing time.

It is worth noting the differences between the actual flight duration and the theoretically best duration listed in Table 2. When there is no congestion, the slightly longer flight duration is attributed to the additional lock of adjacent volume cells to combat uncertainty, whereas in congested situations, the actual flight duration is longer because of trajectory zigzagging or in-air waiting for deconfliction with previously scheduled OIs.

B. Simulation

In this section, we conduct a simulation study to demonstrate the performance of the proposed OI filing process in potentially dense, omnidirectional urban air traffic. The airspace in the simulation covers the Detroit–Ann Arbor metropolitan area (see Fig. 6). The no-fly zone of each airport in the area is delineated by a surrounding polygon. We randomly generate 30 flight requests of arbitrary origin and destination locations lying in the area. The intended ground speed of each operation is sampled in the set $\{10, 15, 20\}$ m/s, and the operation start time is sampled in a 10-minute interval, i.e., offset in the range $[0, 600]$ s. The flight request data are listed in Table 4 and plotted in the right subfigure of Fig. 6. The column header "OD" represents the straight-line travel time between origin and destination, and "Best" represents the best possible travel time between the OD pair considering NFZ along the route.

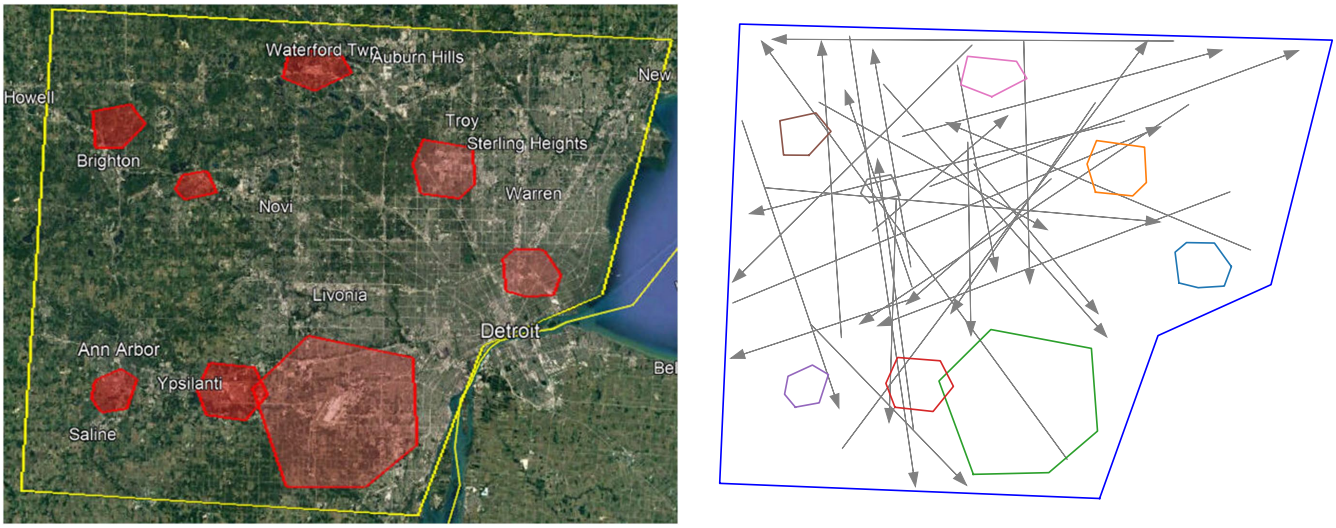

Fig. 6 Test area and OIs.

Table 4 Flight intention data for simulation experiments

No.	OLat	OLng	DLat	DLng	Distance, km	Speed, m/s	Start, s	OD, min	Best, min
1	42.3514	−83.5580	42.2615	−83.8905	29.2	10	318	48.6	58.5
2	42.4324	−82.9231	42.5984	−83.5008	50.9	20	183	42.4	50.1
3	42.6783	−83.4830	42.3935	−83.3986	32.4	15	46	36.0	44.6
4	42.4895	−83.5902	42.1796	−83.5911	34.4	15	439	38.3	50.1
5	42.7202	−83.0769	42.7064	−83.8326	61.9	20	21	51.6	56.4
6	42.6324	−83.0457	42.3150	−83.6513	61.0	20	366	50.8	60.6
7	42.7070	−83.4565	42.3677	−83.8917	52.0	20	157	43.3	52.2
8	42.1416	−83.6766	42.7196	−83.1263	78.6	20	294	65.5	77.3
9	42.5244	−83.3005	42.3445	−83.5652	29.6	10	319	49.3	58.5
10	42.3964	−83.5592	42.6964	−83.6428	34.0	15	522	37.8	47.3
11	42.5419	−83.4463	42.3391	−83.2061	30.0	10	340	49.9	54.3
12	42.3130	−83.7439	42.0955	−83.4430	34.7	15	409	38.5	44.6
13	42.3384	−83.8903	42.6006	−83.0978	71.4	20	230	59.5	68.9
14	42.1352	−83.2565	42.7032	−83.8529	79.9	20	554	66.6	85.6
15	42.5088	−83.5280	42.7109	−82.8415	60.6	20	349	50.5	58.5
16	42.5920	−83.8846	42.1974	−83.6855	46.8	15	203	52.0	64.0
17	42.4446	−83.6328	42.6101	−83.3859	27.4	10	433	45.6	58.5
18	42.7141	−83.6876	42.0918	−83.5383	70.2	20	448	58.5	71.0
19	42.6489	−83.6210	42.3068	−83.1886	52.1	20	79	43.4	45.9
20	42.6318	−83.2215	42.3337	−83.4814	39.4	15	164	43.8	50.1
21	42.2953	−83.6849	42.7057	−83.7404	45.8	15	152	50.9	64.0
22	42.5720	−83.4592	42.3038	−83.4428	29.8	10	193	49.7	62.7
23	42.3686	−83.5557	42.6385	−83.6926	32.0	15	488	35.6	44.6
24	42.1710	−83.5451	42.5446	−83.6277	42.1	15	566	46.7	58.5
25	42.6211	−83.7410	42.4536	−83.3055	40.3	15	155	44.8	55.7
26	42.5000	−83.8384	42.4684	−83.0942	61.3	20	307	51.1	58.5
27	42.7143	−83.3593	42.3784	−83.3353	37.4	15	122	41.5	52.9
28	42.5772	−83.5820	42.7093	−82.9855	51.1	20	188	42.6	48.0
29	42.5125	−82.9642	42.3132	−83.6175	58.2	20	346	48.5	56.4
30	42.6073	−83.1656	42.4632	−83.8645	59.6	20	253	49.7	54.3

Table 5 UTM analyze results

Lock	Thickness	Altitude layer	Delay, min	Extra	CPU	Success
1	1	1	10.0	249	0.2	0.83
		2	4.0	246	0.7	1.00
		3	1.3	210	1.3	0.97
		4	0.4	171	1.8	0.97
	2	1	11.0	414	1.1	1.00
		2	1.5	216	7.3	1.00
		3	0.6	195	28.9	1.00
		4	0.4	207	44.6	1.00
2	1	1	8.6	114	0.2	0.37
		2	8.1	240	0.8	0.63
		3	4.2	237	1.4	0.70
		4	2.3	267	2.2	0.73
	2	1	27.7	552	1.6	0.50
		2	19.9	648	8.4	0.70
		3	6.4	300	58.2	0.67
		4	4.3	330	52.2	0.70

In each experiment, we run the trajectory-planning MIP sequentially for the 30 flights, in the order listed in Table 4. The experiment is run 16 times, composing a complete factorial design over the parameters: lock is selected in the set {1, 2}, thickness in {1, 2}, and altitude layer in {1, 2, 3, 4}. The parameter lock indicates the number of layers of cells surrounding the body segments to be considered occupied in the model. For instance, lock = 1 means that only the body is considered occupied, and lock = 2 means that all cells that are adjacent to a body cell are also considered occupied. A higher value of lock corresponds to more spatial buffer for separation.

The simulation results are summarized in Table 5. The total *Delay* is calculated as the sum, over all successfully scheduled OIs, the actual flight time minus the shortest-path flight time, in minutes. The *Extra* number 4D voxels reserved is calculated in a similar way. The CPU column records the average computing time (including constructing the modeling canvas, sourcing in previously scheduled trajectories as constraints, and solving the MIP model), and the *Success* column indicates the proportion of successfully scheduled operations. If a feasible trajectory cannot be found (i.e., the MIP model returning infeasible), the OI is marked unsuccessful, in which case the operator might consider delaying the operation start time and file again.

The simulation results suggest that the proposed computational method is able to handle cases simulated to resemble reality. The computing time, though heavily dependent on the parameter setting (which determines the model size), is well under one minute for most cases tested. Aligned with the earlier observations, all three parameters have distinguishable impacts on the scheduling performance. Their impacts on *Delay*, the number of *Extra* voxels reserved, and the computing time are compared in Figs. 7–9, respectively.

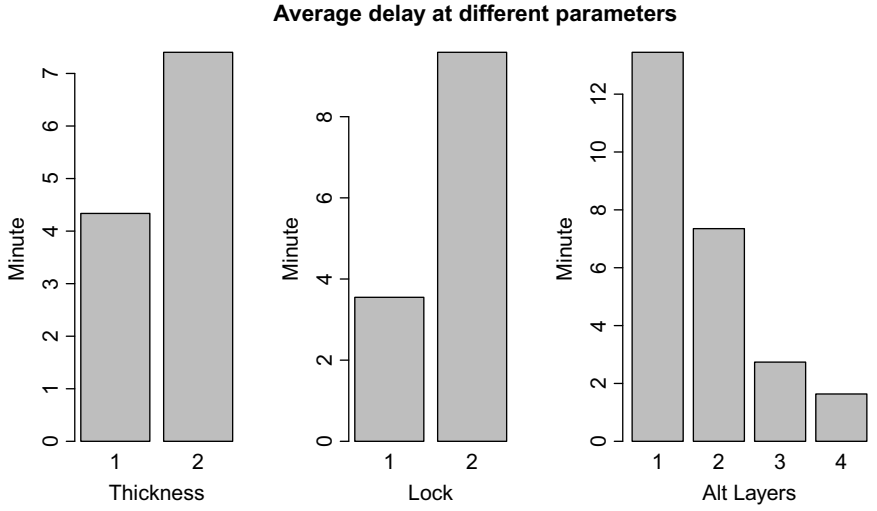

Fig. 7 Effects of thickness, lock, and altitude layer on the average delay.

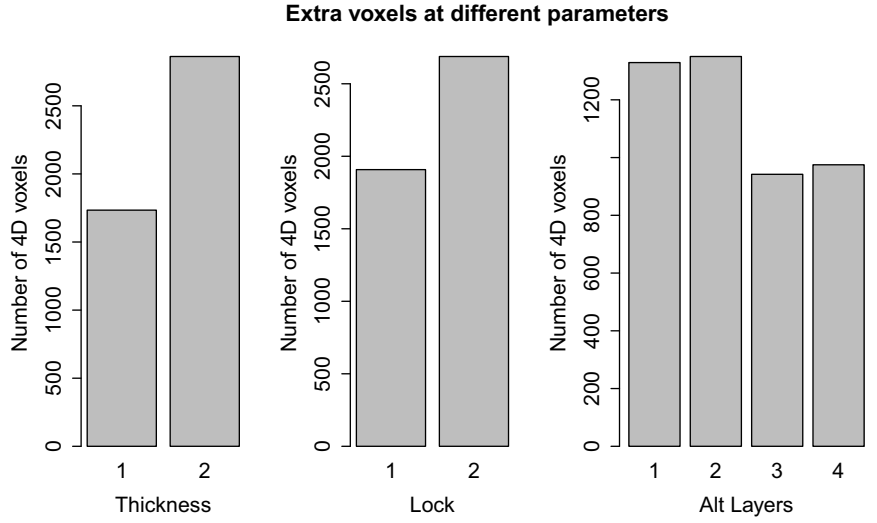

Fig. 8 Effects of thickness, lock, and altitude layer on the extra voxels taken.

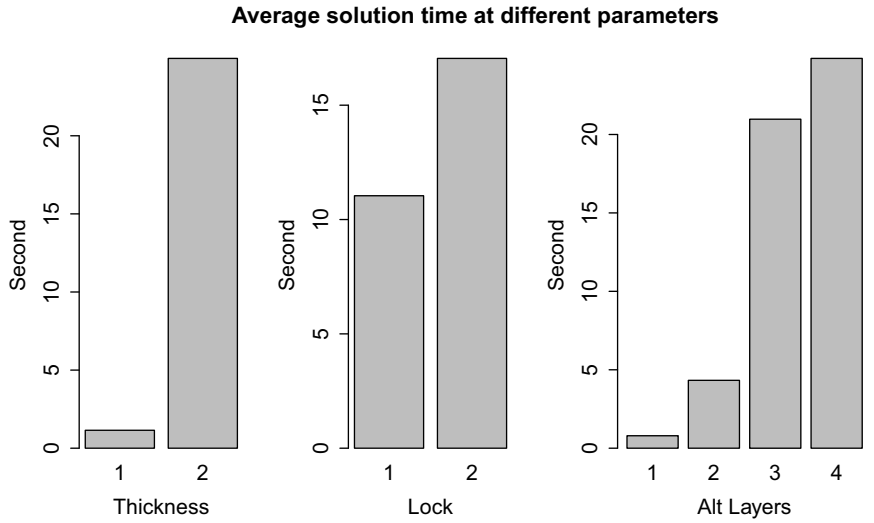

Fig. 9 Effects of thickness, lock, and altitude layer on computing time.

V. Discussion and Future Work

The H3 geospatial indexing system maps the Earth's surface, presumably at sea level, to a hex grid of indexed cells. At a higher altitude, the grid covers a larger spherical area, and thus the cell size will be proportionally larger. Considering that the globally averaged Earth radius is about 6371 km, each 1000 m increase in altitude will increase the edge length of a cell by about $1/6371 \approx 0.16\%$ compared to the nominal values listed in Table 1. Therefore, the effect of altitude on cell traversal time is negligible for most low-altitude operations.

In the IP model and the numerical experiments, it was assumed that altitude changes could be completed within the horizontal footprint of one cell. This assumption can be relaxed to requiring that climbing/descending to an adjacent altitude layer should be completed within the horizontal span of one cell, which is a much weaker assumption that can be satisfied by choosing an appropriate grid resolution commensurate with the aircraft's speed and climb rate properties. To limit the vertical span of slow-climbing aircraft to K layers, for example, we could add this constraint to the IP model:

$$\sum_{h\in\mathcal{H}} x_{s,c,h,t} \le K, \qquad \forall s \in \mathcal{S}, (c,t) \in \mathcal{A}' \tag{19}$$

where K is chosen according to the aircraft's climbing capability.

The usable altitude layers for an operation, as mapped from the user input [$\text{Alt}^{\min}$, $\text{Alt}^{\max}$], reflect the aircraft capability or mission requirements. Additional constraints on altitude availability due to proximate airspace structure, i.e., switching between Class B and Class C airspace, can be modeled by "masking" certain 3D voxels as unavailable for the IP model. In this way, dynamic airspace changes can be easily implemented by changing the voxel reservation mask.

The proposed framework does not require all OIs to adopt the same ground speed. However, the IP model does require the aircraft to maintain the same cell traversal time throughout the execution of its OI trajectory. This means that in a wind field the aircraft might need to adjust its airspeed or change its cell traversal pattern (i.e., use a curvy path instead of the direct center-to-center path) when flying in different directions. Reserving voxels for a longer time before and after the planned visitation time (i.e., increasing the robustness; see Fig. 1 for an illustration) can also help smooth out the airspeed variation caused by varying flight directions. Under light traffic, the actual flight direction will largely agree with the OD direction, which serves as a reference for calculating the required airspeed in windy conditions.

In heavy air traffic, it is inevitable for an aircraft to take some off-the-straight-line detour or reduce speed momentarily in order to avoid conflict with other aircraft, but such actions should be limited. If the trajectory contains too many detours, altitude adjustments, and speed changes just to deconflict with other operations, the operator might consider trying an earlier or later start time (if the business case allows), in the hope of finding a more efficient trajectory for the operation. Let us formalize this practical consideration in terms of trajectory shape requirements.

For a simple A-to-B trip, the following constraints are necessary to ensure efficiency:

1) The trajectory should not contain a loop. In other words, a cell should not be visited twice at two noncontiguous time points.

2) The trajectory should not contain a "knot": at any cell along the trajectory, the next cell to visit should not be a neighbor of the last cell visited. Trajectory search space that conforms to this constraint is illustrated in Fig. 10. At time = 2, neighbors of the cell in which the aircraft was at time = 1 are excluded from the search space for the next cell to visit.

While both of these requirements can be described via adding additional constraints and auxiliary variables in the integer programming model, they are unessential for the proof-of-concept work presented in this paper. Therefore, we leave a detailed treatment of these constraints as an optional for the software implementation work in the future.

The research also leaves a number of other issues for future investigation. First, in the current work, we have adopted an invariable granularity level for the planning grid, i.e., resolution 7 in the H3 system (see Table 1). In situations where UA differ dramatically in size and speed, choosing the cell size flexibly, e.g., in the H3 resolution range of 4–12, would enable better computational efficiency for long-haul flights and improve the airspace efficiency for short-range flights of small UA. Second, it is worth extending the optimization model to incorporate multi-USS, simultaneous trajectory planning in a shared hexagonal grid. Third, characterizing the quantitative relationship between the scheduled traverse time across a hexagonal cell and the actual flight time in continuous space is also of significant importance. This understanding will aid in effectively managing estimation errors over time and enable the implementation

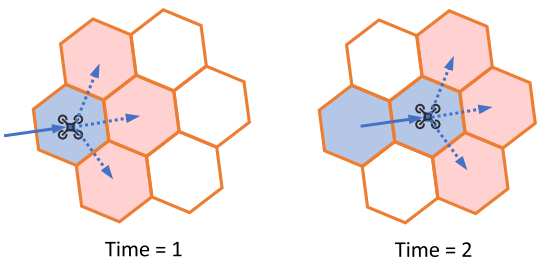

Fig. 10 The trajectory should not contain a knot.

of corrective measures such as speed adjustments or trajectory smoothing to mitigate their propagation.

VI. Conclusions

With the growing integration of UAS into the national airspace, efficient and optimized air traffic management becomes increasingly important. The OI scheduling process is critical to realize effective and equitable allocation of resources among competing airspace users in congested, resource-constrained environments.

In this paper, we have proposed a strategic deconfliction approach for unmanned air traffic management by generating 4D OIs as flight plans for each aircraft. We have developed an integer programming model for optimized trajectory planning based on a stacked hexagonal grid airspace model. We have run a number of validation experiments to demonstrate the effectiveness of the model, as well as simulation experiments to reveal the relationship between the parameter setting and the OI planning outcome. These findings provide empirical guidance to adjust the model setting for different specific application scenarios. The proposed approach can be implemented in a USS to handle strategic deconfliction and constraint management tasks within the UTM framework.

Acknowledgment

The first and second authors are supported by the National Science Foundation under the grant Division of Civil, Mechanical and Manufacturing Innovation (CMMI) 1944068.

References

[1] "Standard Specification for UAS Traffic Management (UTM) UAS Service Supplier (USS) Interoperability," F38.02, A. C., 2021, https://www.astm.org/f3548-21.html.
https://doi.org/10.1520/F3548-21

[2] Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., and Robinson, J. E., III, "Unmanned Aircraft System Traffic Management (UTM) Concept of Operations," *16th AIAA Aviation Technology, Integration, and Operations Conference*, AIAA Paper 2016-3292, 2016.
https://doi.org/10.2514/6.2016-3292

[3] "Unmanned Aircraft Systems Traffic Management (UTM)–A Common Framework with Core Principles for Global Harmonization," 3rd ed., International Civil Aviation Organization, 2020, https://www.icao.int/safety/UA/Documents/UTM%20Framework%20Edition%203.pdf [retrieved 31 Dec. 2022].

[4] "Global Air Traffic Management Operational Concept," International Civil Aviation Organization, 2005, https://www.icao.int/Meetings/anconf12/Document%20Archive/9854_cons_en%5B1%5D.pdf [retrieved 31 Dec. 2022].

[5] Musavi, N., Onural, D., Gunes, K., and Yildiz, Y., "Unmanned Aircraft Systems Airspace Integration: A Game Theoretical Framework for Concept Evaluations," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 1, 2017, pp. 96–109.
https://doi.org/10.2514/1.g000426

[6] Pallottino, L., Feron, E. M., and Bicchi, A., "Conflict Resolution Problems for Air Traffic Management Systems Solved with Mixed Integer Programming," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 3, No. 1, 2002, pp. 3–11.
https://doi.org/10.1109/6979.994791

[7] Frazzoli, E., Mao, Z.-H., Oh, J.-H., and Feron, E., "Resolution of Conflicts Involving Many Aircraft via Semidefinite Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 1, 2001, pp. 79–86.
https://doi.org/10.2514/2.4678

[8] Liu, Y., "A Progressive Motion-Planning Algorithm and Traffic Flow Analysis for High-Density 2D Traffic," *Transportation Science*, Vol. 53, No. 6, 2019, pp. 1501–1525.
https://doi.org/10.1287/trsc.2019.0903

[9] Liu, Y., "A Multi-Agent Semi-Cooperative Unmanned Air Traffic Management Model with Separation Assurance," *EURO Journal on Transportation and Logistics*, Vol. 10, Jan. 2021, Paper 100058.
https://doi.org/10.1016/j.ejtl.2021.100058

[10] Zhang, N., Zhang, M., and Low, K. H., "3D Path Planning and Real-Time Collision Resolution of Multirotor Drone Operations in Complex Urban Low-Altitude Airspace," *Transportation Research Part C: Emerging Technologies*, Vol. 129, Aug. 2021, Paper 103123, https://www.sciencedirect.com/science/article/pii/S0968090X2100142X.
https://doi.org/10.1016/j.trc.2021.103123

[11] Hwang, I., Kim, J., and Tomlin, C., "Protocol-Based Conflict Resolution for Air Traffic Control," *Air Traffic Control Quarterly*, Vol. 15, No. 1, 2007, pp. 1–34.
https://doi.org/10.2514/atcq.15.1.1

[12] Huang, S., Teo, R. S. H., and Tan, K. K., "Collision Avoidance of Multi Unmanned Aerial Vehicles: A Review," *Annual Reviews in Control*, Vol. 48, Jan. 2019, pp. 147–164, https://www.sciencedirect.com/science/article/pii/S1367578819300598.
https://doi.org/10.1016/j.arcontrol.2019.10.001

[13] Jang, D.-S., Ippolito, C. A., Sankararaman, S., and Stepanyan, V., "Concepts of Airspace Structures and System Analysis for UAS Traffic Flows for Urban Areas," *AIAA Information Systems-AIAA Infotech @ Aerospace*, AIAA Paper 2017-0449, 2017.
https://doi.org/10.2514/6.2017-0449

[14] Russell, D., Liquori, M., Lu, C.-T., Sun, D., and Meyer, F., "Multi-Dimensional, Multi-Agent Pathfinding for Autonomous Flight Planning with Airspace Deconfliction," *AUVIS Xponential 2020*, TN, 2020, pp. 1–21.

[15] Chin, C., Gopalakrishnan, K., Balakrishnan, H., Egorov, M., and Evans, A., "Protocol-Based Congestion Management for Advanced Air Mobility," *14th USA/Europe Air Traffic Management Research and Development Seminar (ATM2021)*, 2021.

[16] Yang, X., and Wei, P., "Scalable Multi-Agent Computational Guidance with Separation Assurance for Autonomous Urban Air Mobility," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 8, 2020, pp. 1473–1486.
https://doi.org/10.2514/1.g005000

[17] Egorov, M., Evans, A., Campbell, S., Zanlongo, S., and Young, T., "Evaluation of UTM Strategic Deconfliction Through End-to-End Simulation," *14th USA/Europe Air Traffic Management Research and Development Seminar (ATM2021)*, 2021.

[18] "FAA Order JO 7110.65AA–Air Traffic Control," Federal Aviation Administration, 2023, https://www.faa.gov/air_traffic/publications/atpubs/atc_html/.

[19] Patel, A., "Amit's A* Pages," 2022, http://theory.stanford.edu/amitp/GameProgramming/ [retrieved 31 Dec. 2022].

[20] Xiang, J., Amaya, V., and Chen, J., "Dynamic Unmanned Aircraft System Traffic Volume Reservation Based on Multi-Scale A* Algorithm," *AIAA Scitech 2022 Forum*, AIAA Paper 2022-2236, 2022.
https://doi.org/10.2514/6.2022-2236

[21] Sahr, K., "Central Place Indexing: Hierarchical Linear Indexing Systems for Mixed-Aperture Hexagonal Discrete Global Grid Systems," *Cartographica: The International Journal for Geographic Information and Geovisualization*, Vol. 54, No. 1, 2019, pp. 16–29.
https://doi.org/10.3138/cart.54.1.2018-0022