Rolling Horizon Demand Capacity Balancing in UAM: Integrating Machine Learning for Conflict Resolution

Shulu Chen† and Peng Wei‡

George Washington University, Washington, DC, 20052, USA

James Jones§
MIT Lincoln Laboratory, Lexington, MA 02421

Antony D. Evans¶
Airbus UTM, Sunnyvale, CA, 94086, USA

As urban air mobility (UAM) promises to revolutionize urban transportation, air traffic management must evolve to accommodate increasing traffic density. Demand capacity balancing (DCB) strategically preconditions air traffic by analyzing capacity-constrained resources (CCRs) within structured airspace and determining optimal departure times to minimize conflicts and boost operational efficiency. However, uncertainties such as tactical maneuvers can disrupt pre-established plans by the DCB. To address this, we introduce a rolling-horizon DCB (RHDCB) approach, incorporating a learning-based ETA estimator to manage arrival time uncertainty. Our simulation results demonstrate that the RHDCB significantly reduces conflicts and enhances operational efficiency compared to single-round planning. Additionally, the decomposed mixed-integer linear programming (MILP) method ensures the approach meets real-time operational computational requirements.

I. Nomenclature

\(F_v\) = Set of all aircraft take off from vertiport \(v\).
\(F_r\) = Set of all aircraft on route \(r\).
\(V\) = Set of all departure vertiports \(v\).
\(R\) = Set of all routes \(r\).
\(K\) = Set of all time windows \(k\).
\(K_f\) = Set of flying time windows for flight \(f\).
\(I_r\) = Set of CCRs along route \(r\).
\(B\) = Set of all CCRs in the airspace.
\(D_o\) = Original requested departure queue.
\(D_a\) = Actual departure queue.
\(D_p\) = Period departure queue.
\(T_{max}\) = Operation time horizon
\(T_p\) = Length of the time period (50 seconds as defined in the paper).
\(L_{max}\) = Maximum number of aircraft proceeded in each period.
\(\hat{t}_f\) = Originally scheduled departure time for aircraft \(f\).
\(t_f\) = Optimized departure time for aircraft \(f\).
\(\omega_{k,f,i}\) = Binary variable indicating if aircraft \(f\) arrives at CCR \(i\) in time window \(k\).
\(d_{f,i}\) = Estimated flying time of aircraft \(f\) to capacity-constrained resource (CCR) \(i\).
\(s_k\) = Start timestamp of time window \(k\).
\(\Delta t\) = Average demand interval.

*This project is supported by the NASA Grant 80NSSC21M0087 under the NASA System-Wide Safety (SWS) program.
†Graduate Student, AIAA Student Member, shulu@gwu.edu.
‡Assistant Professor, AIAA Senior Member, pwei@gwu.edu.
§Technical Staff, Air Traffic Control Systems Group, AIAA Senior Member.
¶Traffic Management Systems Architect, AIAA Senior Member, tony.evans@airbus-sv.com.
\[ \Delta s = \text{Minimum temporal separation between departures.} \]
\[ l = \text{Length of the time window (200 seconds as defined in the paper).} \]
\[ \xi = \text{Maximum capacity of each CCR.} \]
\[ C_{b,k} = \text{Capacity of CCR } b \text{ during time window } k. \]

**II. Introduction**

**A. Motivation**

As urbanization intensifies, the demand for transportation is growing daily, leading to an increasingly severe mismatch between demand and the transport system’s capacity. In recent years, novel modes of travel, such as Connected and Autonomous Vehicles (CAVs), have been proposed and implemented [1]. Although they have been proven to alleviate congestion to some extent, they fundamentally do not increase transport capacity. Consequently, eyes are turning skyward, and the development of Urban Air Mobility (UAM) is being considered to alleviate the traffic burden on commuters. Generally, UAM includes electric vertical take-off and landing (eVTOL) aircraft and passenger air taxis, which could revolutionize our daily commutes.

Presently, many companies and research institutions, such as Joby, Archer, Wisk (Boeing), Airbus, and others, have achieved remarkable advancements in this field, developing advanced prototype vehicles. However, significant challenges remain from an air traffic management perspective. Before UAM becomes a reality, we must design a safe and efficient air traffic management system that people are willing to trust and adopt, bringing convenience and assistance to their lives.

The current scenario in traditional commercial airline traffic management faces a labor shortage, potentially threatening flight safety [2]. Utilizing conventional urban airspace management methods is likely to exacerbate this issue. Therefore, it is crucial to incorporate automated flight and safety assurance systems. Notably, in the domain of UAM, both the US Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA) have recently developed concepts for Unmanned Aircraft System (UAS) Traffic Management (UTM) [3–5]. Among the proposals, a primary challenge for an autonomous traffic management system is to mitigate conflicts in high-density traffic flows.

**B. Related Work**

Strategic conflict management in air traffic control involves actions such as ground delays, orchestrated by air traffic managers to harmonize traffic demand with the capacity of airspace bottlenecks like airport runways, merging points, and air route intersections. In traditional Air Traffic Management (ATM), these methodologies have been developed over the years, yielding tangible enhancements for airlines operating within the National Airspace System (NAS). Notably, Traffic Management Initiatives (TMIs) like the Ground Delay Program (GDP), Airspace Flow Program (AFP), and the Collaborative Trajectory Options Program (CTOP) serve as instrumental tools for air traffic flow managers in balancing demand with capacity in congested zones [6]. These initiatives have resulted in reduced delays and cancellations for airlines navigating the NAS while concurrently bolstering safety standards by minimizing the number of aircraft in the airspace, thus averting potential conflicts. Nonetheless, strategic conflict management presents an intricate challenge for UAM due to the elevated traffic density and the populous areas over which such traffic traverses [7]. Consequently, a deeper inquiry is warranted to explore and gauge the efficacy of strategic conflict management within the UAM landscape, particularly in tandem with the assimilation of tactical deconfliction technologies.

The evolution of aviation patterns in urban airspaces, such as UAM for passenger commuting [7] and UAS for delivery services [8], is introducing new challenges. These challenges extend beyond the scope of traditional Air Traffic Management (ATM) and include ensuring safety for high-density operations over urban areas, as well as adapting systems to the unique dynamics of UAM and UAS. To address these challenges, several studies have been conducted. The study in [9] explores traffic flow management within UTM contexts, organizing flights into different airspace layers. Similarly, [10] introduces a distributed framework based on column generation to optimize trajectories for large-scale commercial flights, considering the presence of unmanned aircraft. Focusing specifically on the complexity of UAM flow networks, [11] presents an advanced flow management method. This algorithm efficiently allocates traffic flows across various air routes, enhancing the management and safety of UAM traffic. Additionally, [12] examines UAS traffic flow management, highlighting the balance between total delay costs and fairness in airspace utilization. Complementing this, [13] proposes an optimization model integrated with a First-Come, First-Served approach, aiming to balance...
efficiency and fairness in strategic conflict resolution for air traffic. Moreover, [14] proposes a two-layered framework for strategic deconfliction and battery prognostics-based UAS decision-making, primarily focusing on ensuring safety in terms of available battery energy. Finally, [15] and [16] introduce a framework called V-HATT to manage UAM vertiport operations by controlling the departure and arrival times.

C. Demand Capacity Balancing for UAM

The notion of Demand Capacity Balancing (DCB) has been explicitly underscored by the FAA as a potentially essential element to realize UAM operations as the volume of operations burgeons [5]. DCB may emerge as an effective tool to attain a harmonious equilibrium between operational efficiency and safety levels amidst significant operational uncertainties. Within this framework, DCB functions as a strategic conflict management tool that defines airspace capacity and modulates demand for limited resources. The study by [17] defines the fundamental airspace structure for UAM and employs DCB to optimize slot allocation and trajectory selection. Significantly, it typically abstains from explicit strategic deconfliction measures such as allotting specific flyover times to each waypoint for every aircraft, a practice that could impair operational efficiency and present challenges in managing uncertainties [18]. Conversely, DCB operates to precondition the traffic, delegating the tasks of deconfliction and separation assurance to pilots, air traffic controllers, and their supplementary decision-support systems.

Prior research has delineated the means to identify Critical Convergence Regions (CCRs) within structured airspace for UAM [7], and the integration of DCB with tactical deconfliction to ensure both safety and operational efficiency [18]. However, single-round DCB models, which operate on a deterministic basis, face challenges in the dynamic nature of airspace management:

- **Flight time uncertainty.** One of the primary challenges is the uncertainty in aircraft arrival times at CCRs, often introduced by maneuvers such as speed changes from tactical deconfliction. As aircraft navigate through increasingly complex airspace over extended durations, these arrival time discrepancies can accumulate, impacting the DCB’s pre-planning and potentially leading to overloads in certain time intervals.
- **Large-scale operations.** The single-round DCB processes all aircraft requests and solves the optimization problem over the entire time horizon. As the number of aircraft and the length of the time horizon increase, the computational time rises significantly, which is not suitable for real-time operations.

D. Contributions of this Paper

Addressing these challenges, this paper introduces a rolling horizon demand capacity (RHDCB) algorithm, which provides strategic conflict management with the flexibility to re-evaluate and adjust departure times, accommodating arrival uncertainties more effectively. Moreover, the rolling horizon methodology enables the decomposition of the complex airspace management problem into multiple computational rounds, enhancing the handling of large-scale challenges in a more manageable and computationally efficient manner. This methodology is designed to improve the accuracy of arrival time predictions and manage airspace demand and capacity amidst growing operational complexities.

III. Objectives

This study is structured around three principal objectives:

1) **Development of a RHDCB Algorithm:** We propose an algorithm designed to calculate optimal ground delays for individual aircraft, factoring in predicted traffic density across CCRs within structured airspace and accounting for the basic departure intervals at each vertiport. The algorithm operationalizes this by segmenting the time horizons into discrete intervals. At each interval, the problem is modeled as a mixed-integer linear programming (MILP) challenge and solved using an advanced optimization solver. The algorithm aims to maintain traffic density below a specified threshold, minimize ground delay, and deliver solutions in a time frame that supports real-time decision-making.

2) **A Learning-Based Approach to Arrival Time Prediction:** To enhance the RHDCB’s efficacy, precise traffic density forecasts for each airspace bottleneck are required. This study addresses the issue by developing a predictive model for the arrival times of both airborne and grounded aircraft. We introduce a machine learning algorithm that incorporates variables such as location, altitude, velocity, heading, and data on proximate aircraft. The objective is to refine the accuracy of arrival time estimates at critical junctures.

3) **Simulation-Based Validation and Assessment:** The study’s proposed algorithms will undergo rigorous testing within a simulation environment modeled after the Dallas-Fort Worth airspace. Employing a simplified
rendition of the X4 simulation scenarios from NASA [19], we assess safety, operational efficiency, robustness to uncertainties, and computational expediency. Our benchmarks include both a no-intervention scenario and a deterministic DCB model for comparative evaluation.

IV. Approach

A. Integrated Conflict Management Platform

![Diagram of the integrated conflict management platform with RHDCB.]

This subsection will delve into the implementation of an Integrated Conflict Management Platform, which leverages a RHDCB strategy in concert with a learning-based Estimated Time of Arrival (ETA) predictor. The goal is to precondition air traffic flow and incorporate a tactical deconfliction method suitable for real-time operations. This approach aims to mitigate potential conflicts and elevate the efficiency of operational practices. Figure 1 depicts the architecture of this platform, whose workflow is partitioned into three distinct phases: pre-training, strategic conflict management, and real-time operation.

During the pre-training phase, the system’s initial task is to gather and preprocess simulated data. Subsequently, it focuses on training offline models and deploying estimators designed for various CCRs. Notably, this module is capable of assimilating replay data from live operations, facilitating continuous model refinement—an evolution towards an online methodology.

The strategic conflict management is responsible for acquiring in-flight aircraft data, utilizing ETA estimators to project future traffic densities at specific nodes, and accordingly updating the remaining capacities $C_{b,k}$ for each bottleneck $b$ and time interval $k$. It also processes new departure requests from vertiports and manages the departure queues, which include aircraft pending from the previous period. To optimize the process, ground delays for the current queue are calculated, and the departure schedule is updated for subsequent phases.

During real-time operations, vertiports dispatch aircraft based on the predetermined departure sequence, ensuring adherence to basic departure separation protocols. Additionally, a decentralized tactical deconfliction method is applied to each aircraft in flight to maintain operational integrity. Ultimately, the effectiveness of this integrated system is quantified through safety and efficiency metrics, providing a benchmark for system evaluation.
B. Implementation of Rolling Horizon DCB

The RHDCB, integral to strategic conflict management, segments the time horizon into discrete periods. Within these periods, the system recalibrates departure times by evaluating estimated traffic volumes over each CCR. The mechanism of this process is encapsulated in the pseudo-code provided in Algorithm 1.

Operationally, the system generates three types of departure queues. The ‘originally requested’ departure queue, denoted as $D_o$, encompasses all departure requests $\hat{t}_f$ across flights departure from every vertiport $v \in \mathcal{V}$ and for each flight $f \in \mathcal{F}_v$. The ‘actual’ departure queue, $D_a$, is a dynamic construct that is continually revised, serving as the basis for the virtual ATC’s aircraft release decisions. Lastly, the ‘period’ departure queue, $D_p$, is a provisional compilation used within each time frame to earmark departures $t_f$ designated for the current period. These are subsequently fed into the DCB solver.

**Algorithm 1: Rolling horizon demand capacity balancing.**

1. Collect original requested departure queue: $D_o = \{\hat{t}_f\}, \forall v \in \mathcal{V}, f \in \mathcal{F}_v$
2. Initialize actual departure queue: $D_a = \{t_f\} \leftarrow D_o = \{\hat{t}_f\}$
3. Sorted the actual departure queue in non-decreasing order
4. for $t \leftarrow 1$ to $T$ do
5. Run simulation step
6. if $t$ marks the beginning of period $n \in \mathcal{N}$ then
7. Select departures $t_f$ scheduled in the current time period to form the period departure queue $D_p$
8. if len($D_p$) > $L_{max}$ then
9. Postpone departures exceeding $L_{max}$ to the next time period: $t_f \leftarrow T_p \cdot (n+1)$
10. if $D_p$ not empty then
11. Initialize capacity table for all CCRs $b \in \mathcal{B}$ and all time intervals $k \in \mathcal{K}$: $C_{b,k} = \xi$
12. Get ETA($f$, $b$, $t$) for in-air aircraft $f$ at the corresponding CCRs
13. Update capacity table based on ETAs
14. Calculate optimal departure time in $D_p$ using $C_{b,k}$
15. Adjust $t_f$ to the next time period if $t_f > T_p \cdot (n+1)$
16. Update $D_a$ with the adjusted times: $D_a \leftarrow \{t_f\}$
17. Release aircraft according to the updated schedule in $D_a$

A critical operation highlighted in lines 8–9 of Algorithm 1 limits the number of aircraft processed in each period. This limitation ensures computation time remains stable, a necessity for real-time operational viability. Aircraft not accommodated within the current period are deferred to the onset of the following period, thereby granting them priority in the subsequent processing cycle. The threshold, $L_{max}$, is contingent upon available computational resources and task complexity. Empirical analysis suggests the threshold can be expressed as:

$$L_{max} = 3 \cdot \xi$$  \hspace{1cm} (1)

Here, $\xi$ represents the maximum capacity of each CCR.

Upon establishing the period departure queue, ETA estimators project traffic demands for ensuing intervals. This projection is instrumental in refreshing the capacity table $C_{b,k}$. Subsequently, the DCB solver utilizes the populated $D_p$ and revised $C_{b,k}$ to derive updated departure times $t_f$, which then inform adjustments to $D_a$. A time boundary for aircraft, ensuring adherence to a first-come, first-served sequence, is instituted as indicated on line 15 of the algorithm.

In each period, the DCB solver strategically modifies a subset of departure times. This optimization is cast as a MILP problem, encapsulated by the subsequent mathematical formulations:
In the proposed model, two pivotal decision variables are delineated: the time window identifier, denoted as \( \omega \), and the actual departure times, represented by \( t \). The primary objective, as defined in equation (2), is to minimize the aggregate of departure times for all aircraft \( f \) within the fleet \( \mathcal{F}_v \), taking off from each vertiport \( v \) in the set \( \mathcal{V} \).

This minimization of aggregate departure time, especially when considered alongside constraint (4) that governs the earliest possible departure, effectively reduces the total ground delay. This approach is in line with our ultimate goal of maximizing throughput in operations.

To ensure operational safety, constraint (3) mandates a minimal temporal separation of \( \Delta s \) between successive departures at the same vertiport. Constraint (4) safeguards against any departure being scheduled prior to its originally request departure time \( \hat{t}_f \). The subsequent constraints, from (5) to (7), are operationalized to determine the precise sequence of time windows for the estimated time of arrival at each CCR \( i \) in the set \( \mathcal{I}_r \). In this context, \( \omega_{k,f,i} = 1 \) signifies that the aircraft \( f \) is slated to reach resource \( i \) within the confines of time window \( k \). The term \( t_f + d_{f,i} - s_k \) computes the arrival time relative to the onset of time window \( k \), where \( d_{f,i} \) represents the projected flight duration to CCR \( i \), \( s_k \) is the initial timestamp of window \( k \), and \( l \) is the duration of the window, which is specified as 200 seconds for this study. The identifier is set to 1 exclusively when the relative arrival time falls within the range \([0, l]\), ensuring a singular activation per flight. Constraint (8) is critical as it enforces that the aircraft count at each CCR \( b \) in the collective CCR set \( \mathcal{B} \) does not surpass the designated capacity \( C_{b,k} \). Notably, the CCR set \( \mathcal{I}_r \) is inclusive solely of those elements implicated in route \( r \), whereas \( \mathcal{B} \) encompasses all CCRs within the airspace.

### C. Arrival Time Estimator

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_pos</td>
<td>Aircraft position on the x-axis.</td>
<td>meters</td>
</tr>
<tr>
<td>y_pos</td>
<td>Aircraft position on the y-axis.</td>
<td>meters</td>
</tr>
<tr>
<td>alt</td>
<td>Altitude of the aircraft above mean sea level.</td>
<td>meters</td>
</tr>
<tr>
<td>tas</td>
<td>True airspeed relative to the air mass in which it is flying.</td>
<td>knots</td>
</tr>
<tr>
<td>hdg</td>
<td>Heading angle, indicating the aircraft’s directional orientation.</td>
<td>degrees</td>
</tr>
<tr>
<td>ahead_ac</td>
<td>Number of aircraft ahead within a defined sector.</td>
<td>count</td>
</tr>
<tr>
<td>ahead_dist</td>
<td>Minimum distance to the nearest aircraft ahead.</td>
<td>meters</td>
</tr>
</tbody>
</table>

Predicting the arrival time of an aircraft to CCRs is a critical component in this framework. Traditionally, aircraft arrival time predictions have relied heavily on deterministic models based on radar tracking data and pilot reports. However, for high-density urban airspace, these methods have shown limitations in their accuracy and scalability. To address these challenges, machine learning offers significant advantages due to its ability to learn from large datasets and improve predictions over time. This subsection will introduce the implementation of the XGBoost regression algorithm for the arrival time estimation.
1. Feature Engineering

In this study, we examine various factors influencing aircraft arrival times, encompassing aircraft location, flight dynamics, and data on nearby traffic. Table 1 details the selected features pertinent to our analysis. Notably, we convert the aircraft’s latitude and longitude into Universal Transverse Mercator (UniTM) coordinates. This conversion is necessary as UniTM’s consistent scale and reduced distortion enhance the precision of spatial relationship representation—a critical aspect for the predictive accuracy of geographical features in our model. Moreover, the aircraft’s speed may change due to the tactical deconfliction system, which adjusts speed to mitigate potential collisions. This introduces variability in arrival times at CCRs. Therefore, incorporating data on surrounding traffic, such as the count of preceding aircraft and the minimum distance to the closest leading aircraft, is essential. These elements inform the tactical deconfliction system’s decision-making process, underpinning our model’s ability to anticipate and mitigate arrival time uncertainties.

2. Data Collection

The quality of the input dataset is paramount for training an accurate model. To ensure the model’s generalizability and robustness, we perform fast-time simulations across various scenarios by manipulating the average time intervals for requested departures and predefined resource capacities. Subsequently, we sample data points throughout these simulations. These data points are then integrated and randomized within the dataset to prime the predictive model. Table 2 delineates the specifics of the data preparation process.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range/Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. time interval for requested departure ($\Delta t$)</td>
<td>[60, 120] seconds</td>
<td></td>
</tr>
<tr>
<td>Resource capacity ($\xi$)</td>
<td>{2, 3, 4, 5, 6} units</td>
<td></td>
</tr>
<tr>
<td>Total number of routes</td>
<td>9</td>
<td>routes</td>
</tr>
<tr>
<td>Flights per route ($len(F_r)$)</td>
<td>10</td>
<td>flights</td>
</tr>
<tr>
<td>Simulation iterations per scenario</td>
<td>30</td>
<td>rounds</td>
</tr>
<tr>
<td>Data sampling frequency</td>
<td>16</td>
<td>seconds</td>
</tr>
<tr>
<td>Accumulated data points</td>
<td>139,022</td>
<td>points</td>
</tr>
</tbody>
</table>

The flight departure times are modeled using a beta distribution with shape parameters $\alpha$ and $\beta$. We selected the beta distribution due to its inherent flexibility in adapting to various data shapes and its suitability for modeling variables that are naturally bounded, such as time. Furthermore, the beta distribution’s ability to model different levels of concentration around the mode makes it useful for capturing the nuances of operation demand across diverse routes. The beta distribution, defined on the interval $[0, 1]$, has the probability density function:

$$f(x; \alpha, \beta) = \frac{x^\alpha - 1 (1 - x)^\beta - 1}{B(\alpha, \beta)},$$

(9)

where $B(\alpha, \beta)$ denotes the beta function, serving as a normalization constant to ensure that the integral of the probability density function over $[0, 1]$ is equal to 1.

During each simulation iteration, we construct the original departure queue $D_o$ by aggregating the requests from each route, with each route’s requests being independently modeled by a beta distribution. For a given route, with a total number of flights $len(F_r)$ and an average demand interval $\Delta t$, we define the time horizon $T$ as the product of these two factors:

$$T = len(F_r) \cdot \Delta t.$$  

(10)

We then calculate the departure times $\{d_i\}_{i=1}^N$ by scaling beta-distributed random variates to fit the time horizon $T$. Consequently, each departure time $d_i$ is determined as follows:

$$d_i = T \cdot X_i,$$  

(11)
where $X_i$ is a random variable following the beta distribution. To finalize the departure queue, we arrange the departure times in non-decreasing order:

$$D_o = \text{sort} \left( \bigcup_{r=1}^{R} \left\{ d_{i,r} \right\}_{i=1}^{N_r} \right).$$

(12)

This approach allows for the generation of flight schedules that adapt to different time intervals, accommodating variations in departure requests.

3. Model Selection

Given the input data and the corresponding labels (arrival time on CCRs), the next step is to select an appropriate model. Several methods exist, such as rule-based, linear regression, and advanced machine learning techniques. It’s important to note that methods based on neural networks are typically more suited to classification tasks and may not be ideal for this regression-focused problem. Therefore, our attention is directed towards regression methods.

XGBoost (eXtreme Gradient Boosting) [20], known for its efficiency and performance, is a standout in regression tasks. This algorithm is an implementation of gradient-boosted decision trees, optimized for speed and performance. In XGBoost, the objective function is designed to optimize the model’s performance. It combines a loss function and a regularization term. The loss function, denoted as $l(y_i, \hat{y}_i)$, measures the difference between the predicted values $\hat{y}_i$ and the actual values $y_i$. This component ensures that the model’s predictions are as accurate as possible. On the other hand, the regularization term, represented as $\Omega(f_k)$, helps to minimize overfitting by penalizing the complexity of the model. The regularization term is a function of $f_k$, where $k$ ranges from 1 to $K$, representing the number of trees in the model.

In summary, the final loss function, given by the equation:

$$\text{loss} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

(13)

is a summation of all $n$ data points in the dataset for the loss function and overall $K$ trees for the regularization term.

A baseline method, calculating the distance to the goal divided by current speed, and linear regression were also considered for comparison. As shown in Fig. 2 and Table. 3, the baseline method exhibited a significant dispersion in predictions, suggesting a poor model fit. In contrast, the linear regression model displayed tighter clustering of predictions along the red line of perfect prediction, evidenced by a higher, positive R-squared value. The XGBoost model outperformed both, with the tightest clustering, lowest mean square error and mean absolute error, and highest R-squared value, indicating the best fit among the three models.

![Fig. 2 Comparative Performance of Estimation Models](image)

Beyond accuracy, the decision tree basis of XGBoost offers clear insights into feature importance, aiding in factor analysis and model improvement. Fig. 3 depicts a feature importance plot, derived from the XGBoost algorithm, which quantifies the relative importance of each feature in our model for estimating aircraft arrival times. The ‘F score’ on the x-axis represents the number of times a feature is used to split the data across all trees within the model. A higher value indicates a greater importance of the feature in the prediction process. Notably, ‘ahead_dist’ emerges as the most pivotal feature. This aligns with operational logic where tactical deconfliction critically depends on the distance to the nearest aircraft ahead; changes in speed to maintain safe separation distances can profoundly affect estimated arrival
Table 3  Experimental Results of Different Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Squared Error</th>
<th>Mean Absolute Error</th>
<th>R-squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>18716.8</td>
<td>28.1</td>
<td>0.3593</td>
</tr>
<tr>
<td>Linear regression</td>
<td>622.2</td>
<td>16.5</td>
<td>0.9787</td>
</tr>
<tr>
<td>Xgboosting</td>
<td>58.93</td>
<td>4.0</td>
<td>0.9979</td>
</tr>
</tbody>
</table>

Fig. 3   Feature importance.

Likewise, 'x_pos' and 'y_pos', representing the aircraft’s current coordinates, are also significant predictors, underscoring the model’s sensitivity to spatial positioning. The 'ac_ahead' feature, denoting the number of aircraft ahead, further influences the model, reflecting how traffic density can affect the flow and timing of arrivals.

Conversely, the aircraft’s current state variables such as 'tas' (true airspeed), 'alt' (altitude), and 'hdg' (heading), are assigned lower importance scores. This suggests a dynamic operational environment where the current state offers limited predictive power for arrival times, likely due to the rapidly changing conditions and the need for continual adjustments in response to traffic and air traffic control directives. The minimal role of these features indicates the complexity of estimating arrival times based on the aircraft’s instantaneous status alone. Additionally, its ability to leverage multi-core CPUs during training enhances computational efficiency. Considering these advantages, XGBoost is selected as our model for estimating arrival times in aviation.

V. Numerical Experiments

A. Simulation Environment

To evaluate and validate the proposed method, we utilized structured airspace in the Dallas-Fort Worth area, as depicted in Fig. 4. This airspace layout is designed based on NASA’s X4 simulation for UAM operations [19]. To better adapt to our experimental settings, the airspace network consists of three origins (marked in green), three destinations (marked in red), and corridors (marked in blue lines). The origins, destinations, and waypoints are connected by single-direction edges, with each edge representing a corridor for aircraft. In this setup, we assume all aircraft in a corridor fly in the same direction at the same altitude, with reverse-direction corridors at different altitudes.

Within the structured airspace network, two key waypoints where the flow merges are highlighted in orange and labeled as 'M1' and 'M4'. In the DCB framework, these merging points are considered CCRs, and the demand at these points should be managed by DCB programming. In the simulation, flights are programmed to randomly select origin-destination (OD) pairs, with each flight choosing the shortest route based on the A* algorithm. This setup results
in a total of nine distinct routes, ensuring a comprehensive evaluation of various flight paths and their impact on the predictive models.

For this study, UAM operations are simulated using BlueSky [21], an open-source tool in aviation research. BlueSky is capable of running numerous aircraft simulations in parallel and provides rich interfaces to capture aircraft status and control each flight during the simulation. Thus, through BlueSky, we can combine strategic and tactical deconfliction methods and efficiently measure the performance of the proposed policies.

B. Demand Analysis on Capacity-Constrained Resources

In this experiment, we analyzed the demand curve on the CCRs where we implemented the DCB algorithms. Fig. 5 and Fig. 6 compare the flow of demands at the merging points M1 and M4 over the time horizon. In each subplot, the x-axis represents the arrival time at the merging point of a specific flight, and the y-axis represents the number of aircraft nearby (within 100 seconds). Thus, the area enclosed by the lines connecting each point clearly illustrates the performance of different algorithms in managing the demand at each merging point.

We compared four scenarios:
1) Basic departure separation (without DCB).
2) Single-round DCB, which computes the departure time for each flight before the start of operations (DCB).
3) RHDCB with a rule-based ETA estimator (RHDCB + RB estimator).
4) RHDCB with a machine-learning-based estimator (RHDCB + ML estimator).

All scenarios employ rule-based tactical deconfliction, which introduces speed changes to ensure safe separation.

Given a fixed set of original requests generated by a beta distribution, we compared the demand on the two CCRs by applying four different ground delay rules. Here, DCB algorithms choose capacity limit $\xi = 3$ to manage the demands, which is marked in a red dashed line in each plot. The chosen value is from our previous work [18] that when demands are within this level, the rule-based tactical deconfliction method can handle the safe separation task well. Our observations are as follows:

- **Periods exceeding the capacity limit.**
Fig. 5  Demand at Merging Point M1 for Different Algorithms.

Fig. 6  Demand at Merging Point M4 for Different Algorithms.
The periods above the capacity limit indicate dangerous situations where too many aircraft are merging simultaneously, potentially overwhelming tactical deconfliction measures and leading to safety concerns such as loss of well clear (LoWC) and near mid-air collisions (NMACs). Comparing the areas above the capacity limit reveals that without any DCB algorithms (Scenario 1), the regions around the two CCRs would be overly congested at certain times, posing significant safety risks. And because more flights are crossing the merging point M1, this point suffers a much higher peak than the other one.

For the single-round DCB (Scenario 2), although demands are flattened, several periods still exceed the capacity limit due to the arrival uncertainty introduced by tactical deconfliction not being accounted for.

The RHDCB with the rule-based estimator (Scenario 3) shows several demand peaks, even worse than the single-round DCB, indicating that a poor ETA estimator, which does not consider tactical deconfliction, adversely affects capacity estimation and results in incorrect departure times for some aircraft, leading to overcrowded periods.

The RHDCB with the ML estimator (Scenario 4) has the fewest periods exceeding the capacity limit, demonstrating a strong ability to manage demand by accounting for arrival uncertainties.

- **Airspace utilization efficiency.**

  The length of the demand curve indicates the total time the airspace around the CCRs is utilized, implying airspace utilization efficiency. Using the scenario without DCB as a baseline, we observe that the single-round DCB has the longest time horizon, indicating poor airspace utilization efficiency with many aircraft experiencing unnecessary ground delays.

  Conversely, the RHDCB with the rule-based estimator shows the highest efficiency in airspace utilization, though this compromises safety due to too many aircraft exceeding capacity limits.

  The RHDCB with the ML estimator achieves a balance between safety and airspace utilization efficiency, issuing ground delays only when necessary to specific aircraft.

C. Safety and Efficiency Analysis

Based on the demand analysis on CCRs, we observed that the proposed RHDCB with a learning-based ETA estimator has the potential to improve both the safety level of UAM operations and airspace utilization efficiency. In this experiment, we implemented a Monte Carlo simulation, running each scenario 100 times, and provided numerical results to support these conclusions.

We selected various metrics to measure the performance of the proposed algorithms. LoWC is defined as the event where the distance between two aircraft is within 500 meters. LoWC per flight hour measures the average LoWC time in seconds per flight hour among all operations. NMAC, another safety metric, measures the event where the distance between two aircraft is within 150 meters, significantly increasing the risk of an in-air collision. NMAC per flight hour also measures the average NMAC time in seconds. Additionally, the average number of alerts counts the number of times tactical deconfliction is triggered, requiring an aircraft to change its speed to maintain separation. A lower number of alerts indicates that strategic deconfliction methods effectively reduce the workload on tactical deconfliction. Lastly, ground delay, a time efficiency metric, measures the average ground delay per flight, indicating how strategic deconfliction methods alter departure times.

From previous work on DCB [18], we concluded that DCB can set predefined capacity limits for each CCR, depending on the power of tactical deconfliction and the topology of the airspace network. In this experiment, we compared two different capacity limits, defining DCB strength as high when the capacity is set to 3, and low when the capacity is set to 4. The detailed numerical results are shown in Table 4.

<table>
<thead>
<tr>
<th>DCB strongness</th>
<th>LoWC/ flight hr</th>
<th>NMAC/ flight hr</th>
<th>Avg. number of alerts</th>
<th>Avg. ground delay (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Without DCB</td>
<td>-</td>
<td>29.99</td>
<td>-</td>
<td>4.16</td>
</tr>
<tr>
<td>DCB</td>
<td>6.40</td>
<td>10.78</td>
<td>0.60</td>
<td>1.31</td>
</tr>
<tr>
<td>RHDCB + RB estimator</td>
<td>5.78</td>
<td>12.73</td>
<td>0.54</td>
<td>1.33</td>
</tr>
<tr>
<td>RHDCB + ML estimator</td>
<td>4.26</td>
<td>8.31</td>
<td>0.27</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Table 4  Numerical results without pop-up aircraft
The results in Table 4 support the conclusions from the demand analysis. Without DCB, ground delay is only introduced by basic departure separation, but aircraft are frequently in dangerous situations as both LoWC and NMAC metrics are significantly high, overwhelming tactical deconfliction. For the single-round DCB, the safety metrics are similar to the RHDCB with a rule-based estimator, but the average ground delay is much higher, indicating poor airspace utilization. The RHDCB with the rule-based estimator dynamically measures the remaining capacity of the airspace and adjusts aircraft ground delays efficiently, resulting in low ground delays. However, the poor ETA estimator leads to inaccurate airspace demand estimation, resulting in high safety event occurrences. Compared with the benchmark algorithms, our proposed RHDCB with a machine learning algorithm addresses these shortcomings and demonstrates an improved ability to enhance safety and operational efficiency.

D. Computational Time Analysis

In addition to enhancing safety and operational efficiency, the RHDCB offers benefits in terms of computational time. The single-round DCB processes all aircraft requests and solves the optimization problem over the entire time horizon. As the number of aircraft and the length of the time horizon increase, the computational time rises significantly. To address this issue, the RHDCB decomposes the larger problem, solving for a limited number of aircraft over a fixed length of the time horizon in each computation round. Consequently, the computational time increases linearly with the problem scale.

Based on this analysis, we conducted computational time experiments on different problem scales. The tests were performed on a Linux workstation with a 32-core AMD Ryzen Threadripper PRO 3975WX CPU and 512 GB of RAM, using the Gurobi Optimizer to solve the problems. Table 5 details the results.

<table>
<thead>
<tr>
<th>Number of AC</th>
<th>DCB Time</th>
<th>DCB Rounds</th>
<th>RHDCB + RB estimator Time</th>
<th>RHDCB + RB estimator Rounds</th>
<th>RHDCB + ML estimator Time</th>
<th>RHDCB + ML estimator Rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>0.16</td>
<td>1</td>
<td>3.42</td>
<td>39</td>
<td>3.01</td>
<td>39</td>
</tr>
<tr>
<td>90</td>
<td>1.77</td>
<td>1</td>
<td>7.21</td>
<td>85</td>
<td>9.19</td>
<td>119</td>
</tr>
<tr>
<td>180</td>
<td>489.64</td>
<td>1</td>
<td>17.92</td>
<td>197</td>
<td>21.57</td>
<td>255</td>
</tr>
<tr>
<td>270</td>
<td>1115.75</td>
<td>1</td>
<td>24.13</td>
<td>266</td>
<td>30.81</td>
<td>383</td>
</tr>
</tbody>
</table>

The table shows that the computational time for single-round DCB increases almost exponentially, while for RHDCB, the computational time for each round remains nearly constant, around 0.08-0.09 seconds. Only the number of computation rounds increases linearly as the problem scale grows. Additionally, the method with the rule-based estimator consistently requires fewer computation rounds than the method with the learning-based estimator. This is consistent with our previous conclusion that the rule-based estimator is more aggressive in utilizing the airspace and issues fewer ground delays compared to the learning-based estimator. However, this kind of airspace utilization may lead to the demand overwhelm the capacity limit, and introduce more safety concerns.

VI. Conclusion

In this study, we proposed a UAM conflict mitigation platform that includes a rolling-horizon demand capacity balancing (RHDCB) algorithm, and a learning-based ETA estimator, combined with a tactical deconfliction method. The RHDCB decomposes the strategic conflict management problem into multiple segments, solving for the optimal departure time for each UAM flight to avoid conflicts and enhance operational efficiency. The learning-based ETA estimator, using the XGBoost algorithm, accurately estimates the arrival time of each flight to each CCR, considering the influence of tactical deconfliction. This enhances the ability to evaluate demand at CCRs in real time and supports the RHDCB in making informed decisions. Additionally, the tactical deconfliction method ensures safe separation by adjusting aircraft speeds.

We analyzed the performance of various DCB algorithms through Monte Carlo simulations, focusing on the safety and efficiency impacts of single-round DCB versus RHDCB, particularly when enhanced by a learning-based ETA estimator. Through extensive simulations, we demonstrated that the RHDCB with a learning-based ETA estimator
significantly improves both safety and operational efficiency. The results showed a notable reduction in dangerous situations such as LoWC and NMAC events, alongside more efficient airspace utilization compared to other methods. This approach balances the need for strategic deconfliction with the dynamics of tactical adjustments, ensuring a safer and more efficient use of airspace. Moreover, our computational time analysis revealed that the RHDCB method offers computational advantages. Unlike the single-round DCB, whose computational time increases exponentially with the problem scale, the RHDCB maintains a nearly constant computational time per round, increasing linearly with the problem size. This efficiency is crucial for real-time applications in UAM, where quick and reliable decision-making is essential. The study also highlighted the importance of accurate ETA estimates. The learning-based estimator outperformed the rule-based estimator by providing more reliable arrival time predictions, which are critical for effective capacity management and safety assurance.

In summary, the RHDCB with a learning-based ETA estimator offers a robust solution for managing the complexities of UAM operations. It enhances safety, improves airspace utilization efficiency, and provides practical computational benefits. Future work will focus on addressing weather uncertainty, integrating pop-up aircraft, and further refining tactical deconfliction methods, such as using reinforcement learning models, to enhance airspace utilization and ensure even greater safety in UAM traffic management.

References


