IEICE TRANS. FUNDAMENTALS, VOL.E108–A, NO.1 JANUARY 2025

PAPER

A Framework for Modeling Airspace Traffic Flow without Using Any Specific Waypoints

Kenji UEHARA[†a)], *Nonmember and* **Kunihiko HIRAISHI**[†b)], *Member*

SUMMARY In this paper, we present a framework for composing discrete-event simulation models from a large amount of airspace traffic data without using any specific waypoints. The framework consists of two parts. In the first part, abstracted route graphs that indicate representative routes in the airspace are composed. We propose two methods for extracting important routes in the form of graphs based on combination of various technologies such as space partition, trajectory clustering, and skeleton extraction. In the second part, discrete-event simulation models are composed based on statistical information on flight time along each edge of the abstracted route graph. The composed simulation models have intermediate granularity between micro models, such as multi-agent simulation, and macro models, such as queuing models, and therefore they should be classified as mesoscopic models. Finally, we show numerical results to evaluate the accuracy of the simulation model.

key words: *traffic flow modeling, airspace traffic, space partition, trajectory clustering, skeleton extraction, discrete-event simulation, mesoscopic modeling*

1. Introduction

Recently, large amounts of flight trajectory data are available and various researches using them are conducted. Modeling traffic flow in the airspace is one of such researches [1]. By extracting flight routes from flight trajectory data, we can compose simulation models and predict future traffic flow. Simulation of the models contributes to improvement of flight operation, congestion control, and reduction of CO_2 emission. For example, we can predict traffic flow in near future by simulation models and appropriate traffic control can be applied based on the prediction. Moreover, we can estimate traffic flow after changing flight routes. Models used for such purposes should be those that have a sufficient level of granularity for simulation and prediction but are not too detailed, because behavior of individual aircraft is not very important. Such models have intermediate granularity between microscopic models and macroscopic models, and therefore they should be classified as *mesoscopic models*. Our view of the three models is summarized as follows:

- Macroscopic model
  - Individual objects (aircraft, vehicles, etc.) are not identified.
  - The state of the system is usually represented by continuous variables.
  - Solutions are obtained by analytic approach.
  - Queueing network models are typical macroscopic models.
- Microscopic model
  - Individual objects possess their own information.
  - The state changes depending on interaction between object population and external environment.
  - Multi-agent simulation models are typical microscopic models.
- Mesoscopic model
  - A hybrid model of microscopic models and macroscopic models.
  - Some information about individual objects is abstracted and implemented on the simulation model as common dynamics.
  - It is possible to build a model in which individual objects are identifiable.

To build mesoscopic models for traffic flow, we first identify important points in the target space, e.g., crossings in a road network, and then composing graphs having such points as nodes and routes between nodes as edges. The abstraction level of traffic flow determines the accuracy of simulation results. Compared to road/train traffic, modeling airspace traffic flow is difficult because trajectory of each aircraft fluctuates in the 3-dimensional space due to various factors such as weather and congestion level.

In a previous work by the authors, the mesoscopic modeling approach was applied to traffic flow on an airport surface [2]. An airport surface consists of several fixed points and lines such as aprons, taxiways and runways. This situation is similar to that in road networks. We extracted a graph having nodes representing crossings of taxiways and gave the probability distribution of velocity on each edge. The inputs of the model are departure flights from one of aprons and arrival flights to one of runways. The advantages of using such mesoscopic models are in (i) less computation time keeping a certain level of accuracy, compared to microscopic models such as multi-agent simulation, and (ii) rich information such as the amount of traffic at each point, compared to macroscopic models such as queuing models.

To apply the mesoscopic modeling to traffic flow in the airspace, we propose a framework that consists of two parts.

Manuscript received October 17, 2023.
Manuscript revised May 14, 2024.
Manuscript publicized July 22, 2024.

[†]School of Information Science, Japan Advanced Institute of Science and Technology, Nomi-shi, 923-1292 Japan.
a) E-mail: jaistueh@gmail.com
b) E-mail: hira@jaist.ac.jp

DOI: 10.1587/transfun.2023EAP1132

The first part is a framework for finding important routes in the form of graphs, called abstracted route graphs, based on two approaches: (i) density-based clustering of trajectories and skeleton extraction (DC/SE), and (ii) space partition and trajectory clustering (SP/TC). In the proposed approach, information on waypoints is not used and representative flight routes are extracted from trajectory data. This is the main difference from the modeling approach in [2]. One of the objectives of the paper is to present a method for building simulation models from traffic data in situations where waypoints or similar information is not available (e.g., human flow in a large space). Under the assumption that all the aircraft always pass through waypoints, it is expected that simulation accuracy will improve if waypoints are used explicitly, but this is not always the case in reality.

The second part consists of three steps. The first step is extraction of the passage information at each node by fitting the raw data to the abstracted route graph. The outputs of this step are event logs having list of passing nodes as events. The second step is extraction of probability distribution of flight time between two adjacent nodes. The last step is construction of simulation models on a discrete-event simulation environment. The original contributions in the second part are in (i) aggregation of raw trajectory data by extracting event sequences and associated probability distribution of flight time, and (ii) modeling architecture suitable for mesoscopic discrete-event simulation models.

The paper is organized as follows. In Sect. 2, preprocessing of air traffic trajectory data is described. In Sect. 3, details of the two methods, DC/SE and SP/TC, of the first part are explained. In Sect. 4, the three steps of the second part are explained. In Sect. 5, accuracy of the obtained simulation model is evaluated. In Sect. 6, some remarks on the application to other traffic flows without information on waypoints are described. In Sect. 7, we conclude the paper with remarks on remaining problems[†].

2. Preprocessing of Flight Trajectory Data

As the flight trajectory data, we use CARATS Open data [1]. CARATS Open data consists of flight trajectory data of all regular flights in Fukuoka FIR (Flight Information Region), Japan. The sources of the data are radar data and flight plans. For each flight, the following information is recorded at every 10 seconds: (1) time, (2) flight number (unique ID of the flight), (3) latitude, (4) longitude, (5) altitude, (6) aircraft model (e.g., B772, B738, A320). The data contains flight trajectory data in one week of every odd month (2012–2016); every month (2017, 2018). In this research, we focus on the areas around an airport since trajectory endpoints are easy to identify in airport departure/arrival data and contribute to the acquisition of clean data sets. We here apply the proposed framework to the area around Haneda international airport (HND). The preprocessing procedure is described as follows:

1. Flight trajectory data in the approach control area of Haneda international airport is extracted from the CARATS data.
2. Arrival flights and departure flights are separated, and other flights that pass through this area are eliminated from the data.
3. The coordinate system is changed from WGS-84 to the plane rectangular [6].

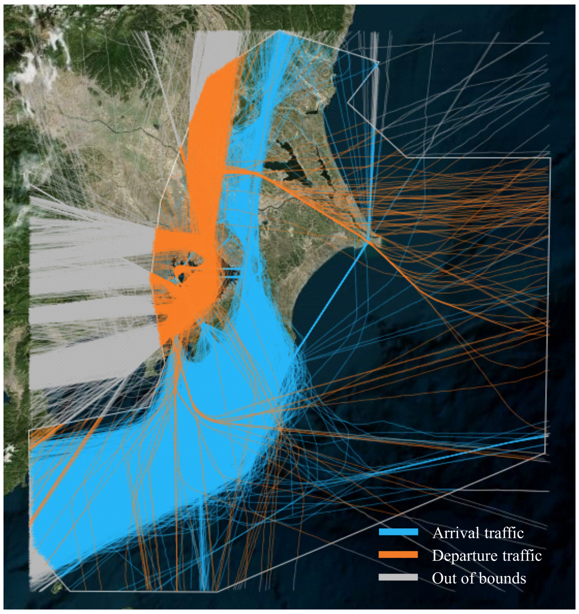


Fig. 1 Trajectories after preprocessing.

Figure 1 shows trajectories after the preprocessing. Arrival and departure flights are indicated by different colors. The approach control area is also indicated. We use trajectories inside of this area. Judgment of arrival/departure flights is made as follows: if the initial point of a flight is close to the airport and with low altitude, then the flight is judged as a departure one; if the final point of a flight is close to the airport and with low altitude, then the flight is judged as an arrival one; other flights are eliminated from the data. Since data near the boundaries of the approach control area is necessary for accurate spatial partition and clustering, data outside the boundaries are also partly used for processing.

In this paper, differences in vertical (altitude) factors are absorbed through preliminary data processing and abstracted graph creation during the mesoscopic modeling process. Aircraft do not ascend and descend randomly; there are strict flight rules and flyable altitude are kept within a certain range. In addition, we divided the traffic data into departures and arrivals at the pre-processing stage. This separation of ascending and descending aircraft in the terminal area further minimizes altitude variation.

[†]This paper is based on two conference papers presented in IEEE SMC 2022 [3] and IFAC World Congress 2023 [4], and also a Ph.D. thesis [5] by the first author.

3. PART I: Composition of Abstracted Route Graph

3.1 Existing Technologies

The proposed framework is a combination of various technologies. We give brief introduction of them. The core technology is clustering of trajectories. It aggregates trajectories with high similarity into clusters and eliminates unimportant trajectories. As the similarity, various distance measures between two trajectories are proposed, e.g., Euclidean distance, distance between trajectories used in TRACLUS [7] and SSPD [8], distance defined on time series data used in DTW [9], and edit distance used in LCSS [10] and EDR [11]. There are variety of clustering algorithms. Due to the characteristics of flight trajectory data, we need to select an appropriate algorithm and a similarity measure. In [12], multiple distance measures are applied to various datasets but there are no significant differences in the obtained results.

Toward extraction of abstracted routes, there are several methods for simplifying traffic trajectory data and extracting abstracted routes from the data. In [13], [14], space partition and the genetic algorithm are applied to AIS (Automatic Identification System) data and patterns of ship trails are found. In [15], [16], main routes of aircrafts in the airspace are extracted using clustering of aircraft trajectories, where DBSCAN [17] based algorithm HDBSCAN is used as the clustering algorithm. In [15], superiority of trajectory clustering algorithm TRACLUS is suggested but this algorithm has not been applied to flight trajectory data.

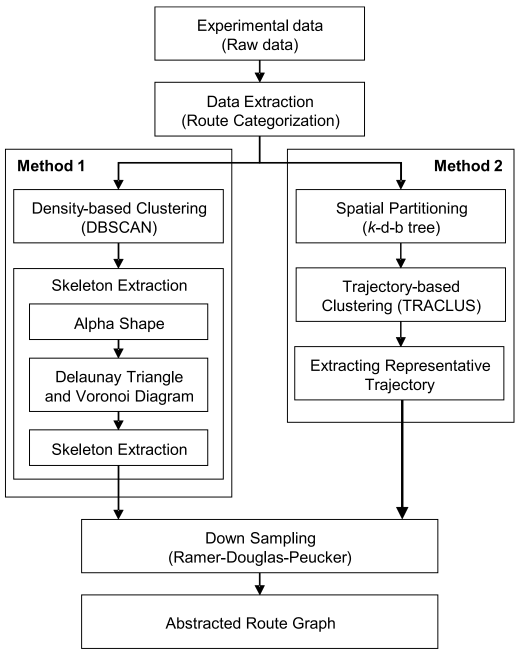
3.2 Overview of Proposed Approach

In this paper, we proposed two methods for composing abstracted route graph. In Method 1, firstly frequently used areas in the airspace is extracted by density-based clustering, and next the skeleton of routes is found. In Method 2, firstly the target airspace area is divided by the density of traffic in each region, and then trajectory clustering is applied to find representative routes. Figure 2 shows the processing flows of the two methods.

3.3 Method 1: Density-Based Clustering and Skeleton Extraction (DC/SE)

(1) Density-Based Clustering

Density-based clustering is an algorithm that begins with areas having points densely, and then expands them by merging neighbor points. DBSCAN is one of representative density-based clustering algorithms and we here use this algorithm. Since DBSCAN generates clusters according to the density of points, clusters are generated in frequently used areas and traffic in other areas is eliminated as noise. This works as filtering of data, i.e., only important areas that contain many flight trajectories remain. There are two parameters ε and



Experimental data (Raw data)
Data Extraction (Route Categorization)
Method 1
Method 2
Density-based Clustering (DBSCAN)
Skeleton Extraction
Alpha Shape
Delaunay Triangle and Voronoi Diagram
Skeleton Extraction
Spatial Partitioning (k-d-b tree)
Trajectory-based Clustering (TRACLUS)
Extracting Representative Trajectory
Down Sampling (Ramer-Douglas-Peucker)
Abstracted Route Graph

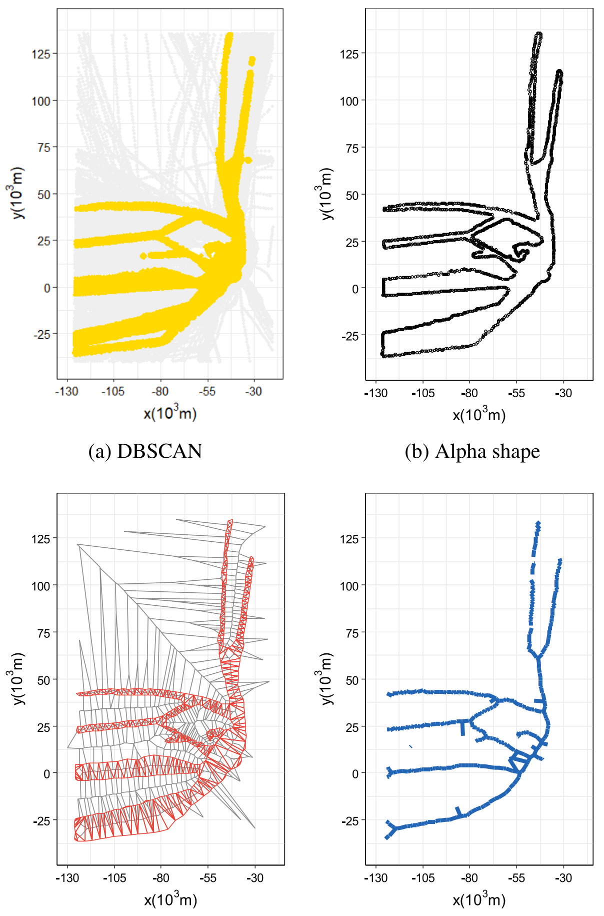
Fig. 2 Overview of the two methods.

minPts in DBSCAN. A cluster is created if there are as many or more points as *minPts* in a circle with radius ε. This procedure is applied to all points in the data and the obtained clusters are concatenated to make clusters larger. Points that do not belong to any clusters are eliminated as outliers.

Large radius ε may result in clusters having unnecessary points. The makes performance of the filtering worse. Large *minPts* works negatively for the generation of clusters. Small *minPts* enlarges the area of each cluster and this makes the post processing for finding main routes difficult. Since the value of *minPts* is sensitive to the amount of data, it is necessary to adjust it even for the same kind of data sources. Figure 3(a) shows the result of DBSCAN. We can observe that only frequently used routes remain and other routes are eliminated.

(2) Alpha Shape

Alpha shapes are extension of convex hulls. We use them for computing contours of the clusters obtained by DBSCAN. The alpha shape method has one parameter α that controls the accuracy of approximation for a given set of points. By the value of α, the alpha shape becomes a convex hull ($\alpha = 0$), and can have holes inside of it. Figure 3(b) shows a result by the alpha shape method. The the obtained alpha shape indicates not only external borders of the cluster but also an inner hole. Such a hole is important to acquire accurate skeletons. A single value of α to all points may not bring a desirable result. Therefore, we use a different value of α for the hole areas. In the current implementation, we manually choose an appropriate value of α according to the density of points in each region.



(a) DBSCAN (b) Alpha shape

(c) Delaunay/Voronoi diagrams. (d) Skeleton.

Fig. 3 Extraction of main routes by DC/SE.

(3) Delaunay Diagrams and Voronoi Diagrams

Next we extract the skeleton of clusters using Delaunay diagrams and Voronoi diagrams. Given a set of seed points on a plane, a Voronoi diagram is a partition of the plane into regions called Voronoi regions such that all points of each region have the same closest seed point. In the case of the Euclidian plane, each border of regions consists of bisections of the line segment between two seed points. Delaunay diagrams are the dual of Voronoi diagrams and indicate the adjacent relation between two Voronoi regions. A Delaunay diagram is obtained as follows: we first put a point in each Voronoi region and draw a line between two points if the corresponding Voronoi regions are adjacent. Usually, a Delaunay diagram consists of triangles called Delaunay triangles.

Figure 3(c) shows the Delaunay diagram and the Voronoi diagram obtained from the alpha shape. Since borders of each Voronoi region is a part of bisections of two points, we can find the center line of the alpha shape by extracting edges of Voronoi regions such that both end points are included in a Delaunay triangle. The edges are shown in Fig. 3(d). These edges indicate the skeleton of the alpha shape.

3.4 Method 2: Space Partition and Trajectory Clustering (SP/TC)

(1) Space Partition

In the space partition method, a given 2-dimensional space is firstly divided into regions by grid lines having variable intervals, and then a tree structure that represents the relation between grid regions is computed. By eliminating unimportant regions from the space, we can reduce computational cost for manipulating the partition. This technique is often used for acceleration of drawing in computer graphics.

We use the space partition technique to divide the space into regions having variable granularity that reflects the amount of traffic there. As indicated in Fig. 1, trajectories of aircrafts do no necessarily exist uniformly in the airspace. In the propose space partition, high traffic areas are divided by high-resolution grids and other areas by low-resolution grids. As the tree structure, Dobrkovic proposes quad trees [13]. However, Filipiak shows that k-d-b trees (k-dimentional binary trees) have less branches at leaf nodes than quad trees [14]. So we adopt k-d-b trees here.

Figure 4(a) shows the result of the space partition together with the density of traffic. It is observed that the size of each region becomes smaller adaptively to the density of traffic. We give the size of leaf nodes as a parameter of the k-d-b tree partition, and determine the grid interval according to the number of points contained in the region. As a result, congested areas have deeper subtrees than other areas.

(2) Trajectory-Based Clustering

In the trajectory clustering, clusters are created from fragments of flight routes. This is different from DBSCAN that considers only individual points. One of disadvantages in the point-based clustering is that we cannot distinguish routes on the same trajectory but in opposite direction since the direction vector of each track is not considered. This is possible in trajectory-based clustering.

We use here TRACLUS as the clustering algorithm. In this algorithm, firstly trajectories are divided into multiple segments, and then clusters are created within each segment. Typical trajectories around an airport begin with a common area (airport terminal area) and then branch. For such trajectories, we can obtain a cluster corresponding to the common area. Clustering in TRACLUS is similar to that in DBSCAN. If there are as many or more segments as *minLns* within radius ε of segment L, then a new cluster having segment L as its core is created. As long as the density of segments around the cluster exceeds the threshold *minLns*, the cluster is enlarged. We determine the length of each segment according to the size of leaf regions, i.e., segments becomes smaller in important regions.

The result by TRACLUS is shown in Fig. 4(b). Since the clustering result drastically changes for the values of parameters, we need to adjust them carefully. Similarly to DBSCAN, *minLns* should be determined based on the amount

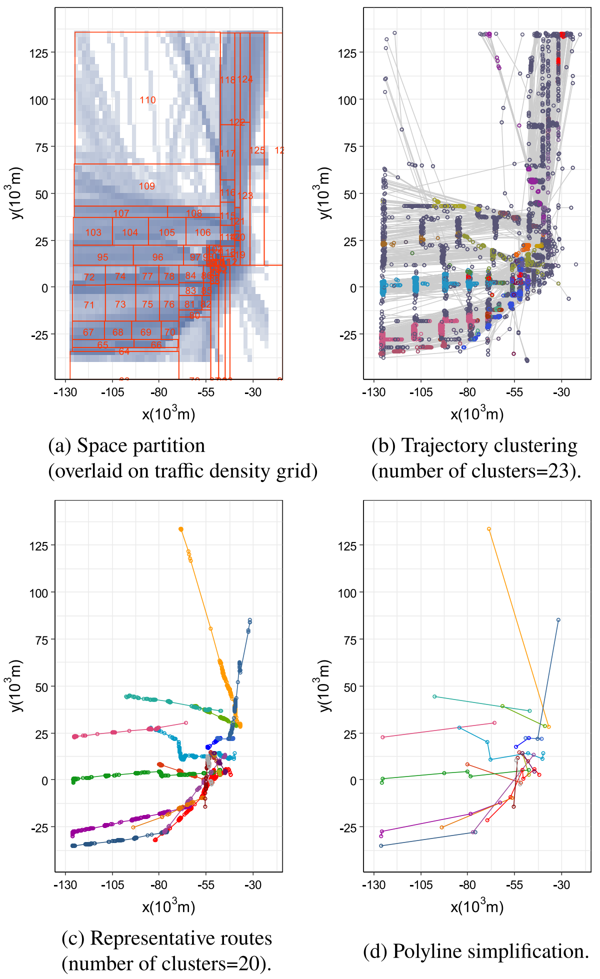

(a) Space partition (overlaid on traffic density grid)
(b) Trajectory clustering (number of clusters=23).
(c) Representative routes (number of clusters=20).
(d) Polyline simplification.

Fig. 4 Extraction of main routes by SP/TC.

of trajectory data, We here use $\varepsilon = 463\,m$ (0.25 NM[†]) derived from precision of the aircraft navigation system [††], and $minLns = 30$. End points of each segment are colored by the cluster number that contains it. It is observed that routes from the airport are clearly separated. Figure 4(c) shows the result of this step.

(3) Extraction of Representative Tracks

Main routes are obtained by connecting representative route fragments extracted from the result of trajectory clustering. We here use TRACLUS for computing main routes. In this algorithm, end points of each segments are found by scanning the plane. Next the average coordinates of the segment that crosses the scan line are obtained. To do this, we first compute the average vector $\vec{V}$ of all vectors $\vec{v_1}, \cdots, \vec{v_n}$ each of which represents a segment in the target cluster. Then we rotate the plane so that $\vec{V}$ becomes parallel to the X-axis and the scanning is done along the direction of $\vec{V}$.

3.5 Down Sampling

The obtained tracks may have too many points. To simplify them, we apply the polyline simplification technique. In this technique, a point is removed when its deviation from the route is less than a given threshold value. We here use Ramer-Douglas-Peucker (RDP) algorithm [18], [19]. RDP is originally a technique for cartographic generalization of drawing maps and is used for simplifying the map according to its scale. RDP algorithm has one parameter ε. If the deviation of a point is smaller than ε, then the point is removed. We use $\varepsilon = 926m$ (0.5 NM) here. The result of simplification is shown in Fig. 4(d). Note that RDP algorithm is also applied to outputs of TRACLUS, skeleton graphs, and k-d-b trees.

3.6 Composing Abstracted Route Graph

We obtain abstracted route graphs from results by DC/SE and SP/TC, respectively. For the output of DC/SE, clusters are concatenated to obtain graphs. For the output of SP/TC, we first remove unnecessary edges that remain after the process using Delaunay/Voronoi diagrams, and apply RDP algorithm. The obtained abstracted route graphs are shown in Fig. 5. It is observed that the two methods give similar graph structures but the details of the graphs such as the number of branches at each node are slightly different. Detailed comparison of the abstracted route graphs obtained by SP/TC and DC/SE is described in [5]. The result of the analysis shows that the abstracted route graph by SP/TC represents frequently used routes more accurately than that by DC/SE.

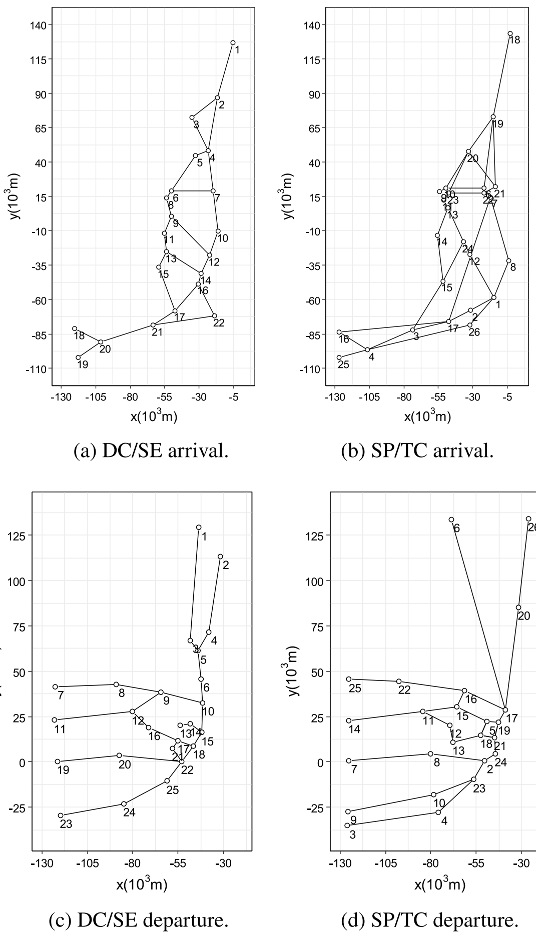
4. PART II: Modeling Airspace Traffic Flow

4.1 Related Work

There are mainly two approaches to the modeling of airspace traffic. The first approach is based on detailed modelling of individual particle (aircraft), and future trajectory of each particle is estimated by the model. This type of microscopic modeling is called Lagrangian approach (e.g., [20], [21]). Multi-agent simulation is classified as this approach (e.g., [22], [23]), and is used for air-traffic management tools (e.g., [24]). The trajectory-based models predict adequately for short intervals of up to 20 minutes, but the accuracy decreases with the increasing prediction interval [25]. In the second approach, space and time are divided into control regions, and each region has properties such as size, density, and flow rate. This type of macroscopic modeling is called Eulerian approach and is used for estimation of traffic flow between adjacent regions. The cell transmission model is a typical Eulerian model for land road traffic. This modeling approach is extended and applied to airspace traffic flow

[†]Nautical mile.

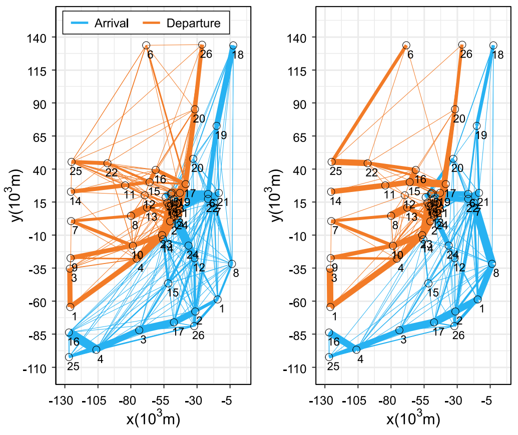
[††]Aircraft arriving at and departing from Tokyo International Airport are almost all large aircraft, and are flown using GNSS-based satellite navigation. The instrument flight procedure set for Tokyo International Airport requires an accuracy of 0.3 NM for the highest requirement, therefore we set 0.25 NM in consideration of actual navigational accuracy.



(a) DC/SE arrival. (b) SP/TC arrival.

(c) DC/SE departure. (d) SP/TC departure.

Fig. 5 Obtained abstracted route graphs.



(a) Overall graph (b) 1-day graph

Fig. 6 Departure/Arrival graphs reflecting traffic volumes. Line thickness of each edge represents the proportion of traffic volume.

[26]–[28]. As stated in Introduction, we here propose the third approach, mesoscopic modeling.

4.2 Traffic Volume Analysis

In the actual aircraft operation, there are transitions between nodes other than representative routes in the abstracted route graph, due to shortcuts and detours. To analyze such irregular routes, we investigate transition information between the nodes by using a process mining tool [29].

First, we extract passage information of each node by fitting the raw data to the abstracted route graph, and then record them as event logs, where we define an event occurs at time instant when the aircraft reaches the point closest to one of the nodes. Next, we obtain transition information between two nodes as bigram by using the process mining tool. Figure 6(a) shows the results of extracting transitions between two nodes for all data under analysis. Figure 6(b) shows the results of the same analysis for one sample day. Although the major traffic flows are consistent with edges of the abstracted route graph, transitions not on edges are observed. In particular, the arrival traffic graphs are complicated in approach routes from the south.

In Fig. 6, major traffic flows can be easily identified because line thickness of each edge denotes the frequency of occurrence. For arrival routes, the pairs of the adjacent nodes $u_i - 1$ and u_i with a high probability of occurrence $P(u_i|u_i - 1)$ are $(18^a, 19^a)$, $(19^a, 6^a)$, $(19^a, 21^a)$, $(16^a, 4^a)$, $(4^a, 3^a)$, $(3^a, 17^a)$, $(17^a, 2^a)$, $(2^a, 1^a)$ from the top, where the superscript indicates a(arrival) or d(departure). On the other hand, divergences from the representative routes are also frequent. This is due to intervention of control instructions frequently causes arrival aircrafts to take shortcuts. Arrival traffic traces with a high probability of occurrence are

$$\begin{aligned}
&\langle 18^a, 19^a, 6^a, 13^a, 11^a\rangle^{0.103},\\
&\langle 18^a, 19^a, 21^a, 22^a, 23^a\rangle^{0.030},\\
&\langle 18^a, 20^a, 21^a, 22^a, 23^a\rangle^{0.022}, \langle 15^a, 24^a, 13^a, 11^a\rangle^{0.018},\\
&\langle 18^a, 21^a, 22^a, 23^a\rangle^{0.017}, \langle 12^a, 24^a, 13^a, 11^a\rangle^{0.013},\\
&\langle 14^a, 13^a, 11^a\rangle^{0.011}, \langle 15^a, 12^a, 24^a, 13^a, 11^a\rangle^{0.010},
\end{aligned}$$

where the superscript indicates the probability, and traces lower than these eight traces have a probability of occurrence less than 0.01. Also in the departure traffic trace, the top high-probability traces are

$$\begin{aligned}
&\langle 5^d, 19^d, 17^d, 20^d, 26^d\rangle^{0.049},\\
&\langle 18^d, 2^d, 23^d, 4^d, 1^d, 3^d\rangle^{0.033},\\
&\langle 18^d, 13^d, 8^d\rangle^{0.029}, \langle 18^d, 2^d, 13^d, 8^d\rangle^{0.028},\\
&\langle 18^d, 2^d, 23^d, 10^d, 9^d\rangle^{0.024},
\end{aligned}$$

and the top 24 traces have probability 0.01 or greater.

The results are well characterized by the arrival and departure routes. The arrival traffic shows characteristics such that the main paths exist at the north entry and south entry around adjacent node pairs $(18^a, 19^a)$, $(19^a, 6^a)$, $(19^a, 21^a)$, $(16^a, 4^a)$, $(4^a, 3^a)$, $(3^a, 17^a)$, $(17^a, 2^a)$, $(2^a, 1^a)$ from which the traffic branches off in detail. Thus, there are a small number of high-frequency traces and many low-frequency traces with similar occurrence probabilities. The departure routes,

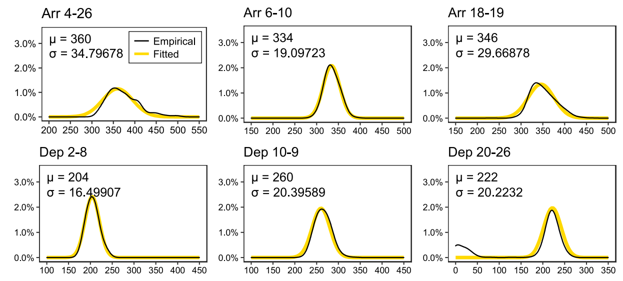


Fig. 7 Typical probability density distribution of flight time.

on the other hand, have many traces with medium probability of occurrence, and are structured in such a way that they branch in a well-balanced manner. This is because the departure routes at Tokyo International Airport are dispersed according to the direction of the destination.

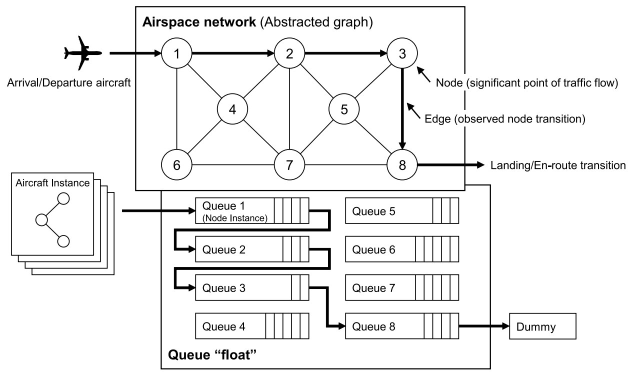
The total number of pairs of adjacent nodes $u_i - 1$ and u_i for the sampled one day was 239 (136 pairs for arrival and 103 pairs for departure). In the airport surface model, traces with low frequency of occurrence were removed as noise, but in the airspace model, probability density distribution parameters were obtained for all patterns to model more rigorously.

4.3 Probability Density Distribution of Flight Time

In the airspace model, we extract the probability density distribution of flight time by the curve fitting technique. From the two-node transition data obtained in 4.1 shows that flight time on each edge varies little and is symmetrically distributed around the mean value on many edges. Therefore, Gaussian family is suitable for airspace data. Fig. 7 shows probability density distributions of flight time and the results of curve fitting on typical edges. The data fits Gaussian distribution well. We conducted curve fitting on all two-node transition data obtained from the trace analysis and extract Gaussian distribution parameters μ (mean) and σ (variance). The obtained parameters allow us to reproduce aircraft movement on the model without having real data.

4.4 Modeling by Object Petri Net

Based on the abstracted route graph and the probability density distribution parameters of inter-node flight time, we compose airspace models using a modeling tool RENEW [30]. It is a tool based on object Petri nets and runs on JAVA VM. Object Petri nets are extension of Petri nets in such a way that each token can be a Petri net having data and methods, i.e., object Petri nets can have hierarchical structure. RENEW is suitable for modeling discrete-event systems with real time feature. Since RENEW supports dynamic generation of net instances, it is also suitable for implementing multi-agent simulation models. Moreover, communication channel between multiple instances is supported.



Airspace network (Abstracted graph)
Arrival/Departure aircraft
1
2
3
4
5
6
7
8
Node (significant point of traffic flow)
Edge (observed node transition)
Landing/En-route transition
Aircraft Instance
Queue 1 (Node Instance)
Queue 2
Queue 3
Queue 4
Queue 5
Queue 6
Queue 7
Queue 8
Dummy
Queue "float"

Fig. 8 Overview of model architecture. An aircraft in the order $1 \rightarrow 2 \rightarrow 3 \rightarrow 8$ (upper figure). In the proposed model architecture shown in the lower figure, all node instances are connected to the common place 'float', and the aircraft instance swims among node instances in the order $1 \rightarrow 2 \rightarrow 3 \rightarrow 8$. Therefore, we do not have to prepare the network structure in the model.

(1) Architecture of the Model

In the airspace model, we adopt hierarchical architecture. The top layer is the airspace Petri net, on which aircraft Petri nets move around. The airspace model has a total of 52 nodes (26 nodes for departure and 26 nodes for arrival). Since the number of patterns of two-node transition are 239 even in one day of log data, directly reproducing the abstracted route graph as a network would require an enormous amount of work. This is a problem faced not only in airspace models, but also in modeling traffic flows that have a vast area of coverage, such as ship data, or traffic flows that are highly random and do not have fixed paths, such as human or animal walking paths.

Here, we take advantage of the characteristics of object Petri nets. We prepare a common place 'float' that connects all instances of node Petri nets. Aircraft Petri nets are assembled in the float place, and by letting the aircraft itself determine which node to go next, there is no need to construct the abstracted route graph as the complex network diagram entirely in Petri nets. In other words, each aircraft Petri net has information of its flight trajectory on the abstracted route graph.

Figure 8 shows conceptual design of the airspace model. It has a hierarchical structure, with the upper layer being the airspace network created based on the abstracted route graph. However, the actual Petri net do not have a concrete network form, but a set of node instances on 'float' place instead. Each node instance is assigned a node ID when it is created, and the aircraft instance swims around in the 'float' in search of this node ID, which results in the same movement as transitioning through the airspace network. The delay time (flight time) cannot be generated by a single node alone because the probability density distribution parameters depend on the last node the aircraft has passed. Therefore, the delay time is generated by the aircraft instance itself. Since the aircraft instance has routing information, it knows the previous node $u_i - 1$ and the current node u_i. The aircraft instance is designed to impose delay time by Gaussian distribution parameters based on the routing information. The airspace Petri net model generates node instances and aircraft instances at the start of the simulation, and then connects all node instances to place 'float'. All aircraft instances are also placed in place 'float'. After the simulation starts, the aircraft repeatedly moves between 'float' and node instances, and when there are no more nodes to move next, it transitions to a dummy node and disappears. The simulation model halts when the number of aircraft instances on 'float' reaches zero.

(2) Airspace Petri Net Model

The airspace Petri net consists of a set of node instances, all of which are connected to a common place 'float'. When an aircraft instance enters place 'float', it searches for the next node instance to enter, and if it can enter, then the node instance moves immediately. Once the aircraft instance enters a node, it obtains the probability density distribution parameters determined by the previous node $u_i - 1$ and the current node u_i, and calculates the delay time. The time is added to the timer of the aircraft instance, and the timer counts down as the simulation proceeds. The node instances act as FIFO queues. As the aircraft leaves the node, it checks its timer value and if the timer is greater than or equal to 1, then it waits; if it is 0, then the aircraft leaves the node and returns to 'float' and search again for the next node to enter.

Air traffic control is the process of keeping aircraft in line and evenly spaced, and all aircraft must follow the rules. Air traffic service to aircraft is provided on a "first come, first served" basis, except for emergency and presidential aircraft [31]. Therefore, we can say the aircraft between nodes behave as FIFO. If there are special airspace conditions that are frequently disrupted by anormal traffic, then the effects between aircraft must be taken into account, but such cases do not usually occur. If they frequently occur, the trajectory will be clustered as shorter part by the clustering process, with variations occurring according to their effects. In other words, the output is a queueing network with a finer mesh according to the complexity of the airspace.

Figures 2–5 shows node instances of the actual airspace model for the arrival traffic. Although there are 26 independent node instances, they are all connected to a common place 'float'.

Each aircraft instance moves among the node instances according to its routing information. If the next node to go cannot be identified in the node set, then the aircraft stays at the 'float'. The simulation completes when the number of aircraft instances on 'float' reaches zero. As the aircraft instances swim around the float and nodes, they record transition information as an event log. This event log is used for conformance checking and model evaluation by process mining.

We should remark that the proposing Petri net model is not a multi-agent simulation model because of the following reason. Although each aircraft model has information on the flight plan, but it is nothing but an alternative representation

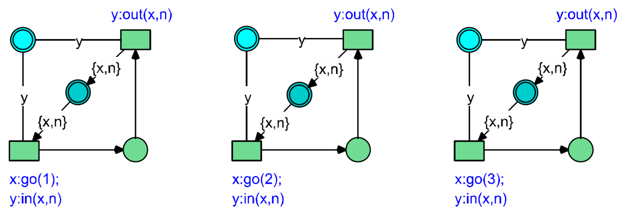


Fig. 9 Floating queues (node 1-3). In the model, 26 node instances and a dummy node are located, and all connected to the common place 'float' (dark cyan place), where the same place can be put on different net instances. Each aircraft instance in float can enter a node if the transition having method ':go(n)' of the aircraft instance fires synchronously with the transition having methods 'x:in(x,n)' and ':in(x,n)' of a node instance, where these methods compose a communication channel through a shared variable n. Note that n denotes the node ID and x denotes the aircraft instance. When the aircraft exits the current node, the transition having ':out(x,n)' fires synchronously with the transition having 'y:out(x,n)' of the node instance and the aircraft instance returns to 'float'.

of the graph structure. Moreover, the flight speed of each aircraft is determined by the common probabilistic distribution. Therefore, each aircraft is not capable of deciding its behavior by itself.

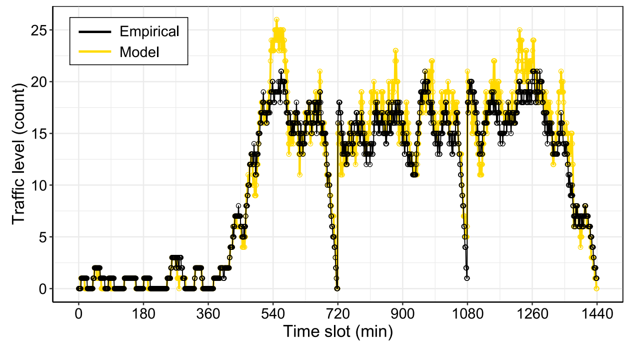
5. Evaluation of the Model

Whether the simulation model shows desired performance or not depends on its objective. Mesoscopic models have the advantage of downsizing large amounts of raw data while still preserving the information needed for analysis. The important information in airspace models is accuracy of flight time for each aircraft and the throughput in the airspace. Therefore, we evaluate the traffic congestion level in the target airspace area and the flight time of each aircraft. Note that the evaluation is to be conducted after completing the conformance checking by process mining, confirming that the model is working as designed, and assuring that the event logs do not contain any incorrect information [32].

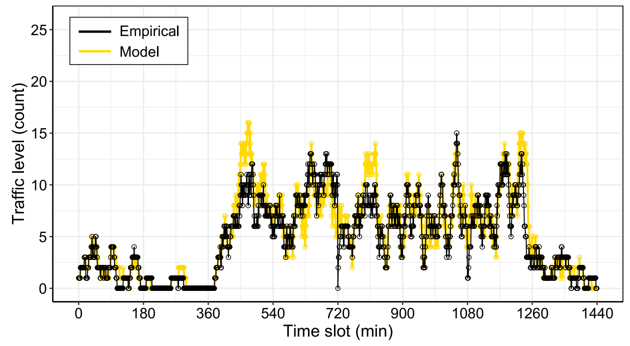
(1) Validation of Airspace Traffic Level

We count the number of aircrafts flying in the target airspace area at a given moment. We will refer this number as *the traffic level*. Here, the time axis is divided into slots of a fixed time interval (60 seconds), and the number of in-flight aircrafts in the airspace is counted for each slot.

Figure 10 shows the results of traffic level comparison for arrival and departure aircrafts. For both arrival and departure, the simulation results show good tracking compared to the raw data. In arrival traffic, there is a time period when the traffic level drops around time slot 720 and 1080 (correspond to 12:00 and 18:00, respectively), and the model is able to reproduce almost the same drop points. On the other hand, there are some areas where the traffic level in the model exceed the actual for both arrival and departure. For example, in the vicinity of time slot 540 (9:00) for arrival, the model shows around 24 aircrafts while the actual data is around 20 aircrafts. Similarly for departures, there are time slots where the traffic level by the simulation is higher than



(a) Arrival traffic level



(b) Departure traffic level

Fig. 10 Airspace traffic level. The size of the queue assigned to each node is set to 5. The x-axis is represented in time slots of 60 seconds.

the actual level. There are two possible reasons for this:

Queue capacity overflow: In the model, all node instances have the same queue capacity $l_q = 5$. This value is set by considering the length of edges in the abstracted route graph and the horizontal separation in the terminal area. If more inputs than the queue capacity arrive at once, then the queue overflows. The number of aircrafts that can fly simultaneously on an edge is equal to the capacity of the queue in the node instance. Therefore, if the queue overflows, then the aircraft instances will be stuck in 'float'. However, we assume that separation is well arranged in actual airspace, and air traffic control operations are well performed without overflows even during busy times. The queue capacity of each node is also related to the throughput of the airspace to some extent. Therefore, how much the queue capacity can be expanded determines the throughput of the airspace. Actually, unlimited expansion of queue capacity does not necessarily improve the tracking of raw data. Table 1 shows the maximum error, the mean error, and the standard deviation between the raw data and the model for each queue size. For the initial value of $l_q = 5$, the mean error is 1.175 for arrivals and 0.973 for departures, which is generally within the range of about one aircraft for each time slot. As l_q increases, the error becomes smaller for the arrival aircraft, but conversely, the error becomes larger for the departure aircraft. Moreover, although the accuracy can be improved by adjusting queue capacity, the error at maximum congestion is not due solely to queue overflow.

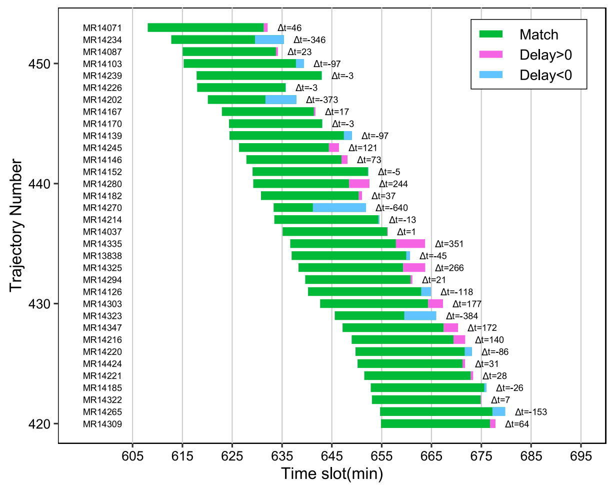
Table 1 Traffic level comparison between raw data and model output.

Route	Queue size	Max. error	Mean error	Std. dev.
Arrival	5	7	1.175	1.380
	6	6	1.147	1.337
	10	6	1.106	1.258
Departure	5	9	0.974	1.302
	6	9	0.980	1.290
	10	10	0.994	1.313

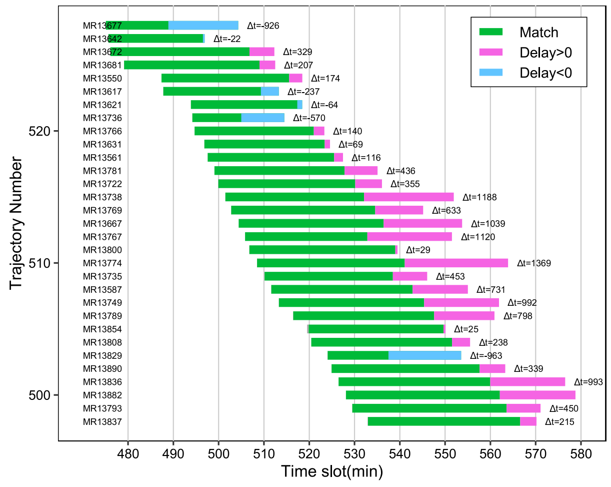
Runway alternation: In the normal airport operation, the used runway is changed depending on the wind direction or other factors. At large airports with multiple runways, such as Tokyo International Airport, the change of runway has a significant impact on arrival and departure routes, and therefore it causes discrepancy between simulation and actual operations. The results of trace analysis for the arrival traffic shows that the runway direction changed around time slot 360 (6:00). The final approach phase of the traces for the southbound approaching aircraft is $\langle 14^a, 13^a, 11^a \rangle$ before time slot 360, but after that the phase changes to $\langle 22^a, 20^a, 9^a \rangle$. As a result, both the aircrafts approaching from the north and south use common segments, which affected the following aircraft in the simulation and caused delays. In actual operations, however, congestion was mitigated by the air traffic control (ATC) instructions, which probably caused the discrepancy. In actual ATC operations, runways are not suddenly switched without prior arrangement. Since the wind direction is predictable, arrangement should be made in advance to minimize the impact on operations before the runway is changed. Therefore, the actual data shows less congestion than the simulation. Although there are differences between actual operations and the model outputs for traffic peaks during maximum congestion, they converge quickly, and the model traffic volume follows the actual data in a short time.

(2) Validation of Flight Time

The flight time is defined as the time between the appearance and disappearance of an aircraft in the real data. In the simulation by the airspace model, only the appearance time is given to the aircraft instance as an individual property. Therefore, if place 'float' is congested at the original appearance time and the aircraft cannot enter the first node, then the appearance time is delayed. If the preceding aircraft is stuck in the node and it takes time to transition to the node, then the accumulated delay will be the delay of the disappearance time. The comparison results of flight times between the raw data and the simulation of arriving aircrafts are shown in Fig. 11. We here show results for arrival flight, but similar results were obtained for departure flight. The results for departure flight is available in [5]. Figure 11(a) shows the results when good accuracy was obtained in the traffic level comparison. The simulation reproduces the actual operation generally well, with minimal errors. On the other hand, Fig. 11(b) shows the results when the model gives bad



(a) Time periods showing good consistency (arrival)



(b) Time period showing bad consistency (arrival)

Fig. 11 Flight time comparison between model and actual data. Green bars represent the model matches the real data. If the model took longer flight time relative to the real data, these are indicated by magenta bars; and if the model took less time relative to the real data, these are indicated by blue bars. The strings on the right side of the y-axis indicate the flight ID.

performance. Increasing the operating time affected the following aircraft. Delays begin to appear around time slot 510 (8:30), and about 10 more aircrafts are affected by the delays that follow, with the delays gradually increasing.

Unlike taxiing aircraft on the ground, aircraft in flight cannot stop in mid-air, so if the preceding aircraft is stuck and cannot proceed, then the following aircraft has to either enter a holding pattern and wait for the congestion to ease, or make a detour. Holding aircraft creates additional delay time and has an additional impact on the following aircrafts. In practice, however, few aircrafts are forced to hold in mid-air. Even in congested airspace, the situation is well coordinated through the ATC intervention, and efforts are made to prevent the deterioration of punctuality. Let us focus on the green bars in Fig. 11(b), although slightly longer than in Fig. 11(a), is the same length for almost all aircrafts, with no significant bias. It is thought that ATC interventions are made to avoid negative effects on congestion by adjusting the separations by radar vectors, adjusting the speed of following aircraft, and applying narrower separation with tight margin. The airspace model does not reproduce the flexible spacing adjustments made by ATC intervention, making subsequent aircraft more susceptible to congestion and changes in runways in use. However, the duration of the impact is slight, and the model shows considerable accuracy at other times of the day. Thus, as with the traffic level assessment, the airspace model reproduces aircraft behaviour to some extent in the assessment of individual aircraft operating times, and the necessary information is retained even after downsizing by mesoscopic modeling.

6. Application to Other Traffic Flows

The methods presented in this paper can be applied to traffic flows without fixed points other than air traffic flows. The traffic flow data for vessels and vehicles have been widely used for a long time, and there are a variety of methods for their analysis. As mentioned in 3.1, vessel AIS data are used in [13], [14] to perform space partition and also used the genetic algorithm to extract key traffic flow points of the traffic flow. In [8], trajectory clustering are also performed with SSPD on cab trajectory data. These are applicable to the modeling framework presented in this paper. The methods could also be applied to pedestrian and animal tracking data. Pedestrian data in station premises and plazas are different from those with fixed routes, such as roads, where the randomness of movement is high, and it is difficult to extract the major traffic flow simply by plotting it directly. In the method we proposed, the space is divided on a density basis and the trajectories of each pedestrian are clustered. Thus, it is possible to extract the major traffic flows hidden in the data. Animal Tracking Data can also be applied to extract frequently traversed paths from animal activity histories in forests, etc., where there are no fixed paths. Deer track data are used in the evaluation of the TRACLUS trajectory clustering [7].

7. Conclusion

In this paper, we have proposes a framework for modelling airspace traffic flow. The framework consists of two parts.

In the first part, we have shown two methods, DC/SE and SP/TC, for extracting route graphs that represent characteristics of airspace traffic flow. Since fluctuation exists in flight trajectories, representative routes are extracted. In DC/SE, first density-based clustering is applied, and next a geometrically-defined skeleton is extracted from each cluster. In SP/TC, first the airspace area is divided into grid regions according to importance of each region and next trajectory clustering is applied. The abstracted route graphs are obtained by both methods but they have different characteristics. Separate routes in the graph by SP/TC tend to be merged in the graph by DC/SE, i.e., the SP/TC gives more detailed graphs. We need to choose the method according to usage of the model. Through the evaluation on geometrical and operational points of view, we observe SP/TC gives a more appropriate graph for the modeling but the result may change by parameter setting.

In the second part, we have proposed a method for composing mesoscopic discrete-event simulation models based on statistical information on flight time along each edge of the abstracted route graphs. Since the airspace models are complex in their composition, we takes advantage of the characteristics of object Petri nets and allow the aircraft instance itself to swim among nodes instances in place 'float'. This contributes to reducing the amount of work required to create the airspace network and improving the efficiency of model implementation. In the evaluation of the airspace model, good reproducibility was demonstrated for traffic level and flight time. This means that the proposed mesoscopic modeling framework can reproduce airspace traffic flow with some accuracy while significantly downsizing large volume data. The raw traffic flow data used in the airspace modeling is several gigabytes in size, but was downsized to a few megabytes by replacing all of the node-to-node movement information with statistical data in the model. Depending on the scale of traffic flow data and the granularity required by designers for mesoscopic models, the required modeling effort can be enormous. Mesoscopic models originally had the advantage of providing accurate results to some extent with lightweight processing instead of directly handling raw data.

The proposed framework requires appropriate values for parameters, especially in the extraction of abstracted route graphs. Searching good values in a somewhat automatic way remains as future work.

Acknowledgments

This research is partly supported by Grant-in-Aid for Scientific Research of MEXT under Grant No. 20K04544 and No. 23K10996. The authors also thank to Japan Civil Aviation Bureau, MLIT for providing CARATS Open data.

References

[1] http://www.mlit.go.jp/report/press/kouku13_hh_000087.html
[2] K. Uehara and K. Hiraishi, "An efficient mesoscopic modeling method for large volume traffic flow using process mining techniques," Proc. 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), 2021.
[3] K. Uehara and K. Hiraishi, "A framework for extracting abstracted route graphs toward air traffic flow modelling," Proc. IEEE SMC2022 (online), pp.1564–1569, 2022.
[4] K. Uehara and K. Hiraishi, "Mesoscopic modeling of airspace traffic flow," Proc. 22nd IFAC World Congress, pp.5000–5005, 2023.
[5] K. Uehara, "A study on mesoscopic modeling methodologies for large volume traffic flow data," Ph.D. Thesis, Japan Advanced Institete of Science and Technology, 2022.
[6] B.R. Bowring, "Total inverse solutions for the geodesic and great elliptic," Survey Review, vol.33, no.261, pp.461–476, 1996.
[7] J.G. Lee, J. Han, and K.Y. Whang, "Trajectory clustering: A partition-and-group framework," Proc. 2007 ACM SIGMOD International Conference on Management of Sata, Association for Computing Machinery, pp.593–604, 2007.
[8] P. Besse, B. Guillouet, J.M. Loubes, and F. Royer, "Review and perspective for distance based trajectory clustering," IEEE Trans. Intell. Transp. Syst., vol.17, no.11, pp.3306–3317, 2016.
[9] D.J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," AAAI-94 Workshop on Knowledge Discovery in Databases, pp.359–370, 1994.
[10] M. Vlachos, G. Kollios, and D. Gun, "Discovering similar multidimensional trajectories," Proc. 18th International Conference on Data Engineering, pp.673–684, 2002.
[11] L. Chen, M.T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," Proc. 2005 ACM SIGMOD International Conference on Management of Data, pp.491–502, 2005.
[12] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," Proc. 34th Very Large Data Bases Endowment, vol.1, no.2, pp.1542–1552, 2008.
[13] A. Dobrkovic, M.E. Iacob, and J. van Hillegersberg, "Maritime pattern extraction and route reconstruction from incomplete AIS data," Int. J. Data Sci. Anal., vol.5, pp.111–136, 2018.
[14] D. Filipiak, K. Węcel, M. Stróżyna, M. Michalak, and W. Abramowicz, "Extracting maritime traffic networks from AIS data using evolutionary algorithm," Bus. Inf. Syst. Eng., vol.62, pp.435–450, 2020.
[15] L. Basora, J. Morio, and C. Mailhot, "A trajectory clustering framework to analyse air traffic flows," 7th SESAR Innovation Days, 2017.
[16] X. Olive and M. Jérôme, "Trajectory clustering of air traffic flows around airports," Aerospace Science and Technology, vol.84, pp.776–781, 2019.
[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," Proc. Second International Conference on Knowledge Discovery and Data Mining, pp.226–231, 1996.
[18] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," Computer Graphics and Image Processing, vol.1, no.3, pp.244–256, 1972.
[19] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," The Canadian Cartographer, vol.10, no.2, pp.112–122, 1973.
[20] C.R. Kaplan, E.S. Oran, N. Alexandrov, and J.P. Boris, "The monotonic Lagrangian grid particle grid: A fast tracking methodology for air-traffic modeling," American Institute of Aeronautics and Astronautics Paper 2009-1635, 2009.
[21] A. Bayen, P. Grieder, G. Meyer, and C.J. Tomlin, "Lagrangian delay predictive model for sector-based air traffic flow," Journal of Guidance, Control, and Dynamics, vol.28, no.5, pp.1015–1026, 2005.
[22] Á. Cámera, D. Castro, E. Oliveria, and P.H. Abreu, "Comparing a centralized and decentralized multi-agent approaches to air traffic control," Proc. 28th European Simulation and Modelling Conference, pp.189–193, 2014.
[23] R. Breil, D. Delahaye, L.. Lapasset, and É. Féron, "Multi-agent systems for air traffic conflicts resolution by local speed regulation and departure delay," Proc. IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 2016.
[24] ETMS: https://www.fly.faa.gov/Products/Information/ETMS/etms.html
[25] B. Srighar, N.Y. Chen, and H.K. Ng, "An aggregate sector flow model for air traffic demand forecasting," Proc. 9th AIAA Aviation Technology, Integration, and Operations Conference, AIAA 2009-7129, 2009.
[26] H.M. Arneson and C. Langbort, "Distributed control design for a class of compartmental systems and application to Eulerian models of air traffic flows," Proc. 46th IEEE Conference on Decision and Control, pp.2876–2881, 2007.
[27] P.K. Menon, G.D. Sweriduk, and K.D. Bilimoria, "New approach for modeling, analysis, and control of air traffic flow," Journal of Guidance, Control, and Dynamics, vol.27, no.5, pp.737–744, 2004.

[28] D. Sun and A.M. Bayen, "Multicommodity Eulerian-Lagrangian large-capacity cell transmission model for en route traffic," Journal of Guidance, Control, and Dynamics, vol.31, no.3, pp.616–628, 2008.
[29] ProM: https://www.promtools.org/
[30] RENEW: http://www.renew.de/
[31] FAA Joint Order 7110.65W: https://www.faa.gov/documentlibrary/media/order/atc.pdf
[32] K. Uehara and K. Hiraishi, "Process mining approach for the conformance checking of discrete-event simulation model," Proc. 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp.615–620, 2019.

Kenji Uehara received from the Tokyo University of Foreign Studies the B.A. in Language and Area Studies in 2010, and from the Japan Advanced Institute of Science and Technology (JAIST) the M.S. in Information Science in 2014 and D.S. in Information Science in 2022. His research interests include discrete event systems, traffic flow modeling and simulation.

Kunihiko Hiraishi received from the Tokyo Institute of Technology the B.E. degree in 1983, the M.E. degree in 1985, and D.E. degree in 1990. He is currently a professor at School of Information Science, Japan Advanced Institute of Science and Technology. His research interests include discrete event systems and formal verification. He is a member of the IEEE, IPSJ, and SICE.