Transportation Research Part C 21 (2012) 89–121

Contents lists available at SciVerse ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A multi-objective approach for Dynamic Airspace Sectorization using agent based and geometric models

Jiangjun Tang *, Sameer Alam, Chris Lokan, Hussein A. Abbass

University of New South Wales at the Australian Defence Force Academy, Canberra, Australia

ARTICLE INFO

Article history:
Received 11 November 2010
Received in revised form 23 August 2011
Accepted 26 August 2011

Keywords:
Dynamic Airspace Sectorization
ATC workload
Sector flight time
Distance between sector boundaries and traffic flow crossing points
Agent
Geometry computation
Multi-objective optimization
Genetic Algorithm

ABSTRACT

A key limitation when accommodating the continuing air traffic growth is the fixed airspace structure including sector boundaries. The geometry of sectors has stayed relatively constant despite the fact that route structures and demand have changed dramatically over the past decade. Dynamic Airspace Sectorization is a concept where the airspace is redesigned dynamically to accommodate changing traffic demands. Various methods have been proposed to dynamically partition the airspace to accommodate the traffic growth while satisfying other sector constraints and efficiency metrics. However, these approaches suffer from several operational drawbacks, and their computational complexity increases fast as the airspace size and traffic volume increase. In this paper, we evaluate and identify the gaps in existing 3D sectorization methods, and propose an improved Agent Based Model (iABM) to address these gaps. We also propose three additional models using KD-Tree, Bisection and Voronoi Diagrams in 3D, to partition the airspace to satisfy the convexity constraint and reduce computational cost. We then augment these methods with a multi-objective optimization approach that uses four objectives: minimizing the variance of controller workload across the sectors, maximizing the average sector flight time, and minimizing the distance between sector boundaries and the traffic flow crossing points. Experimental results show that iABM has the best performance on workload balancing, but it is restrictive when it comes to the convexity constraint. Bisection- and Voronoi Diagram-based models perform worse than iABM on workload balancing but better on average sector flight time, and they can satisfy the convexity constraint. The KD-tree-based model has a lower computational cost, but with a poor performance on the given objectives.

1. Introduction

Continued air traffic growth (an average rate of 4.8% for passenger traffic per year through the year 2036) (ICAO, 2010) opens up new research challenges in terms of airspace capacity management as well as air traffic safety and efficiency. Various methods are being developed to meet these challenges including air traffic flow management methods (Weigang et al., 2010), flight path planning (Sherali and Hill, 2009), etc.

Dynamic Airspace Configuration (DAC) is a new operational paradigm that proposes to migrate from the current structured, static airspace to a dynamic airspace, capable of adapting to user demand while meeting changing constraints of weather, traffic congestion and complexity, as well as a highly diverse aircraft fleet (Kopardekar et al., 2007).

* Corresponding author.
E-mail address: jiangjun.tang@student.adfa.edu.au (J. Tang).

0968-090X/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trc.2011.08.008

DAC in current operations has limited options, in terms of how sectors and airspace can be reconfigured, due to various technological and human factors issues (Mitchell et al., 2008). DAC envisions sectors to be substantially more dynamic in future, changing with the changes in traffic, weather, and resource demands (Zelinski, 2009).

Understanding the limitations of the state-of-the-art methods in DAC is a necessary initial step towards designing effective airspace reconfiguration support tools and operational concepts.

An essential element to meet the increasing demand in air traffic is to use more efficient allocation of airspace for capacity demand balancing (Ball et al., 2001). This can only be achieved by managing aircraft trajectories while the structure of the airspace as well as the Air Traffic Controller (ATC) resources are constantly adjusted to meet airspace user needs (Lee et al., 2008).

In DAC research, Dynamic Airspace Sectorization (DAS) is an initial approach for restructuring the airspace, in order to achieve capacity-demand balance while managing the workload for air traffic controllers and ensuring an orderly flow of traffic. In DAS, an airspace sector is dynamically reconfigured based on dynamic density measures (e.g. traffic flow, aircraft density, sector boundary proximity, time in sector, heading variability and speed variability), ATC workload, sector geometry constraints (e.g. convex shape, right prism constraints) and others (FAA Order 7210.3U, 2006; FAA Order 7400.2F, 2006; FAA Advisory Circular 90-99, 2003). On the other hand, it takes time to train ATC to be familiar with a sector configuration (Homola et al., 2010). Therefore, to maintain the similarity of sector shapes is another research question in DAS.

By its core nature, DAS has a number of conflicting objectives. We cannot allow a large number of small sectors to be created, because of the limited number of available air traffic controllers. Conversely, we cannot group sectors so much that the workload of an air traffic controller within a sector increases dramatically. Therefore, multi-objective approaches have been suggested recently as a method for DAS (Brinton and Pledgie, 2008; Xue, 2008). Some constraints and criteria must be considered when developing a DAS approach. These include convex sector shapes, right prism, alignment between sectors and traffic flows, minimum distance between traffic crossing points and sector boundaries, computation cost, etc. Convex sector shapes prevent flights entering the same sector more than once in order to reduce the handover workload of ATC. Right prism is required for a 3D sectorization because the ATC has only a 2D projection on their screen and facilitate their comprehension of the situation. Alignment between sectors and air traffic flows decreases the traffic flow cuts, which reduces the handover workload as well. Maintaining enough distance between traffic crossing points and sector boundaries enables ATC to have enough time to respond the potential conflicts. Minimizing the computation cost of DAS is required in an operational environment because DAS has to be adaptive to the frequent traffic changes within an acceptable time window.

Some 2D DAS optimization approaches have been developed, such as constraint programming (Trandac et al., 2003) and geometric algorithms (Basu et al., 2009), but these did not consider the multi-objective aspect of the problem. Recently there has been a focus on 3D sectorization, given the practicality of the concept when it comes to field implementation of the dynamic sectorization concept, such as Delahaye and Puechmorel (2008). In these approaches, heuristic airspace partitioning methods are used to generate sectors.

In this paper, we first experimentally evaluate and identify the gaps in the existing agent-based 3D sectorization model. We then propose an improved Agent Based Model (iABM) with simplified Movement Rules to address these gaps. Though the proposed iABM model has several advantages over existing models, such as right prism satisfaction and lower computation cost, it inherits some limitations, such as convexity constraint violation. Therefore, three other models, based on KD-Tree, Bisection and Voronoi Diagrams, are also proposed and evaluated for 3D partition of the airspace. These approaches are based on geometry computations, and focus on geometry constraints such as convexity and right prism.

We then propose optimizing the dynamic airspace sectors generated by the proposed four models, using a Genetic Algorithm. The four models are then compared on ATC workload balancing, average sector flight time, and minimum distance between sector boundaries and traffic flow crossing points. The performance and efficiency of the proposed models are then demonstrated using sample air traffic data.

2. Background

Different methods are proposed in the literature that approach the airspace sectorization problem in different ways, and thus produce different looking sectorizations. We first summarize the common methods for 2D DAS, and then look at the state-of-the-art in 3D Airspace sectorization methods.

2.1. Dynamic Airspace Sectorization methods

There are three common methods for 2D DAS widely reported in the literature.

2.1.1. Flight trajectory clustering

A common approach is to cluster the flights based on their trajectories (Brinton and Pledgie, 2008). Air traffic flow is composed of 4D aircraft trajectories; each trajectory is a temporal-spatial data set. Flight trajectory clustering groups flight track positions together to sectorize airspace. Sector boundaries are then formed around the groupings of flight route segments. However, the methods to explore, cluster, and analyze these temporal-spatial data are not straightforward.

2.1.2. Mixed integer programming algorithm

The Mixed Integer Programming (MIP) algorithm discretizes the airspace into hexagonal cells, and clusters the cells according to workload and connectivity (Mitchell et al., 2008; Verlhac and Manchon, 2001). The workload of a cell is the number of flight track counts within that cell. Connectivity from cell i to a neighboring cell j is the total number of flights that travel from cell i to cell j. Flow enters each cell from at least one of its neighbors and exits into exactly one neighboring cell. The workload of each cell is added to the flow, which is finally absorbed by a sink cell.

2.1.3. Voronoi Diagrams with Genetic Algorithm

In this method, Voronoi Diagrams are used to partition the airspace and a Genetic Algorithm is used to optimize the partitions (Delahaye et al., 2001; Xue, 2008; Songchen and Zhang, 2004). The Voronoi Diagram decomposes a space into subdivisions around given generating points. All points within a region associated with a specific generating point are closer to that generating point than any other generating point. A Genetic Algorithm is then used to find the set of Voronoi Diagram generating points that optimize given parameters.

2.2. 3D airspace sectorization

Altitude is not considered in 2D sectorization. However, it is a very important consideration in sectorization. Air traffic flows moving in opposite directions are always altitude separated. Also, depending on an aircraft's size, it may or may not have the ability to attain certain altitudes. Moreover, flight climb-out (during take off) and descent (during landing) make altitude consideration critical for sector designs, especially in the transition airspace area. Hence there is a recent focus on the third dimension — altitude — while looking into dynamic sectorization.

3D airspace sectorization is more complex than 2D airspace sectorization. 3D sectorization has specific geometry constraints, such as right prism constraint and 3D tessellation. A right prism requires a polytope to have bases aligned one directly above the other and has lateral faces that are rectangles. This is important because the ATC needs a 2D projection of the sector on their screen, which is not possible if the right prism constraint is violated. Fig. 1 shows an example of a polytope satisfying the right prism constraint and its 2D projection.

Kicinger and Yousefi (2009) extended the 2D airspace sectorization method of Mitchell et al. (2008) to 3D partitioning using a heuristic method. This approach is presented as a heuristic airspace partitioning method, combining a Genetic Algorithm (GA) and an agent-based model, GA is used to determine the initial locations of agents, and the agent-based model is used to determine cell clustering. The airspace is tiled with uniform grid cells in three dimensions. The sector boundaries are determined by grouping cells according to two objectives:

- Cumulative traffic load represents the ATC workload in a given time interval. This is calculated by accumulated traffic load (radar hits in each cell) in a given period, from historical or simulated data. In order to balance the workload among the sectors, the mean value of the traffic load is produced and an acceptable workload range is also given, based on the mean traffic load.
- Cumulative commonality measures the alignment between the redesigned sectors and the main traffic flows. The commonality in each cell is quantified by the number of aircraft traveling from this cell to its neighboring cells in a given time interval. For example, the commonality of cell i to its neighboring cell j is c if a total of c aircrafts fly from cell i to cell j in a given time interval. The total commonality of cell i is then the total number of aircrafts traveling from cell i to all of its neighboring cells.

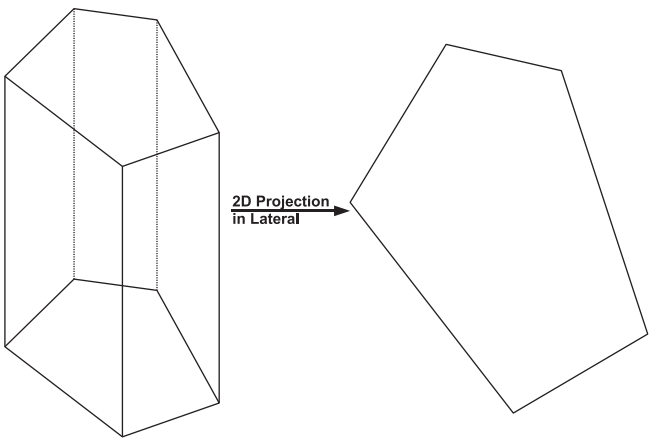


Fig. 1. An example of right prism and its 2D projection in lateral.

Their approach uses an agent-based model (ABM) for clustering cells. Each agent represents a sector, which moves in the airspace to find and group local optimal cells and layers into its sector. The local optimal means the cells or layers are selected based on giving the biggest increase in a sector's Cumulative Commonality when they are added to the sector. ABM is an iterative procedure, which keeps running until all cells in the airspace are assigned to sectors or a predefined number of iterations is reached.

This approach partitions the airspace based on the 3D grid cells grouping. It cannot guarantee to produce sector shapes that satisfy right prism requirements. Therefore, the right prism is modeled as a constraint.

Four rules work together to achieve workload balance. The Movement Rule uses a moving-ratio (from 0.0 to 1.0), which determines the fraction of agents moving at each iteration. The Layer Growth Rule limits the workload growth when new cells or layers are added to the agent: the cumulative workload is checked to ensure that the workload does not exceed an upper bound (1.2 times the average workload). The Trading Rule is executed when an agent's cumulative workload exceeds the upper bound on workload variation: by the Trading rule, an agent transfers the external boundary cells to its neighbor agent in order to reduce its own workload. The right prism constraint is also checked when grouping new cells and layers; the Repair Rule is called when cells violate the right prism constraint.

The model can partition 3D airspace according to traffic commonality and workload accumulation. However, there is no optimization of airspace sectorization, which means that this approach cannot provide optimal DAS solutions. In addition, several problems, such as right prism violation, nested sectors, and high computation cost, are unavoidable. This is investigated below in Section 4.

3. Research objectives and method overview

There are two objectives of this paper. The first is to evaluate the existing ABM and identify its operational shortcomings, and then to develop improved models to address the identified gaps. The second objective is to apply a multi-objective optimization method to generate optimal DAS solutions with these new models, and to analyze and compare their performance.

3.1. Problem definitions

For DAS, the inputs are a given airspace and a set of flight trajectories, and the output is a set of partitioned sectors that satisfy specified objectives, such as balancing ATC workload.

In this paper, two kinds of approaches are evaluated: one works on a discrete airspace (agent-based models) and the other works on a continuous airspace. Therefore, two different definitions for the airspace are used here.

The discrete airspace is denoted as $R(C(X')_{uvz})_{u=1,v=1,z=1}^{U,V,Z}$, where $C(X')_{uvz}$ are the uniform cells within R and $X' = \{x_1, x_2, x_3, \ldots, x_c\}$ is all points inside cell C_{uvz}. $x_j = \{lat_j, lon_j, alt_j\}$ is a 3-tuple defined by latitude, longitude, and altitude. u, v, and z are the indices of a cell based on its relative location in R. They are calculated by Eq. (1):

$$\begin{aligned} u &= \frac{lat - minLat}{latSize} + 1 \\ v &= \frac{lon - minLon}{lonSize} + 1 \\ z &= \frac{alt - minAlt}{altSize} + 1 \end{aligned} \tag{1}$$

where *lat*, *lon*, and *alt* represent the position of a point which has the minimum values of latitude, longitude, and altitude within the cell(C). The *latSize*, *lonSize*, and *altSize* are uniformly predefined. *minLat*, *minLon*, and *minAlt* represent the position of a point with the minimum values within the given R. Therefore, the relationship between cells (C) and R is

$$R = \bigcup_{u=1,v=1,z=1}^{U,V,Z} C_{uvz}$$

where U, V, and Z are the indices of a cell containing the point with the maximum values of latitude, longitude, and altitude inside R.

The continuous airspace is defined as $R(X)$, where $X = \{x_1, x_2, x_3, \ldots, x_n\}$ is all points included in R. $x_j = \{lat_j, lon_j, alt_j\}$ is a 3-tuple recording latitude, longitude, and altitude, with the same definition as in the discrete airspace.

The air traffic is a set of trajectories: $T = \{t_i\}_{i=1}^{N}$, where

$$t_i = (x_{ij}, spd_{ij}, time_{ij})_{j=1}^{M}$$

t_i is the trajectory of flight i, consisting of a set of ordered points $x_{ij} = \{lat_{ij}, lon_{ij}, alt_{ij}\}$ and the speed (spd_{ij}) and time ($time_{ij}$) associated with each point.

In DAS, only the trajectories located within R are considered. Therefore,

$$\forall i \text{ and } \forall j \ : \ x_{ij} \in R$$

The airspace R is partitioned into K sectors S_k,($k = 1, 2, 3, \ldots, K$). K is the number of sectors to be generated. For discrete approaches, sectors are denoted as $S(C')_k$ where C' is a set of continuous cells building up the sector and satisfies

$$\text{IF } C' \in S_k \text{ THEN } C' \notin (R \setminus S_k)$$

The boundary cells (BC_k) of $S(C')_k$ can be found by Eq. (2):

$$BC_k = \forall C_{u,v,z} \; IF \; \begin{array}{l} (C_{u+1,v,z} \notin C') \bigvee (C_{u-1,v,z} \notin C') \bigvee \\ (C_{u,v+1,z} \notin C') \bigvee (C_{u,v-1,z} \notin C') \bigvee \\ (C_{u,v,z+1} \notin C') \bigvee (C_{u,v,z-1} \notin C') \end{array} \tag{2}$$

$S(\widehat{X})_k$ is used as a sector notation for the continuous approach, where S is the superset of $\widehat{X}$ and $\widehat{X}$ is a subset of X. Additionally, for both sector definitions,

$$R = \bigcup_{k=1}^{K} S_k$$

and

$$S_i \cap S_j = \emptyset; \quad i \neq j; \quad i,j \in \{1,2,3,\ldots K\}$$

After the space is partitioned as S_k, all points x_{ij} in T must satisfy

$$\forall i \text{ and } \forall j \; : \; \text{IF } x_{ij} \in S_k \text{ THEN } x_{ij} \notin (R \setminus S_k)$$

For DAS, it is necessary to model the ATC workload mathematically. In this paper, the workload of a sector is measured by the count of traffic hits within the sector during a given period. Eq. (3) describes the workload measurement we use:

$$W_k = \sum_{i=1, x_{ij} \in S_k}^{N} |t_i(x_{ij})| \tag{3}$$

In the equation, W_k is the workload of a sector S_k, which is measured by the count of x_{ij} within S_k from all trajectories t_i.

Sector flight time is also measured in this paper. It contributes to the alignment between sectors and traffic flows, which minimizes the traffic flow cut and therefore reduces the handover workload for ATCs. It is modeled as Eq. (4).

$$SFT_k = \frac{\sum_{i=1}^{N} \sum_{j=1, x_{ij} \in S_k, x_{ij+1} \in S_k}^{M} (time_{ij+1} - time_{ij})}{|T(t_i)|_{x_{ij} \in S_k}} \tag{4}$$

The sector flight time is derived from the time stamps ($time_{ij}$) of its continuous radar signatures in t_i, which contain positions (x_{ij}) at $time_{ij}$ located within the sector (S_k). $T(t_i)$ is a set of trajectories in the space R as described above. Therefore, the count of all trajectories t_i that have some parts x_{ij} located within a sector S_k is the number of flights flying within S_k. According to these, the sector flight time for a sector can be produced by Eq. (4).

The minimum distances (D) between traffic crossing points and sector boundaries is considered in this paper as well. A larger distance between crossing points and sector boundaries gives more responding time to ATC in order to resolve a potential conflict. The crossing points of air traffic flows (CP) are identified based on the 4D flight trajectories ($t_i(x_{ij})$), where the two flight footprints are satisfying:

$$\begin{array}{l} Dist(x_{pu}(lat_{pu}, lon_{pu}), x_{qv}(lat_{qv}, lon_{qv})) \leqslant 5 \text{ NM} \; \bigvee \\ |x_{pu}(alt_{pu}) - x_{qv}(alt_{qv})| \leqslant 1000 \text{ ft} \; \bigvee \\ |time_{pu} - time_{qv}| \leqslant 300 \text{ s} \end{array} \tag{5}$$

x_{pu} and x_{qv} are the locations of two flights p and q at $time_{pu}$ and $time_{qv}$. As shown in the Eq. (5), we take 5 NM lateral distance and 1000 ft vertical separation within 300 s time window as a threshold to identify the crossing points between flights. The two flight footprints are treated as the crossing points of traffic flows. Then the minimum distance between a traffic crossing point and the corresponding sector boundaries for a sector can be calculated as:

$$D_k = \min_{i=1}^{P} (Dist(CP_i, Boundaries(S_k))) \; IF \; CP_i \in S_k \tag{6}$$

where P is the total number of crossing points identified in the given airspace R and the minimum distance (D_k) is calculated for the sector S_k only when CP_i is inside it.

3.2. Experiment design

Based on the problem definition above, the problem can then be divided into the following stages:

- Evaluation of 3D DAS models:
  - Evaluation of existing ABM.

- Development and evaluation of an improved agent-based model (iABM).
- Development and evaluation of non-agent-based 3D models (using KD-tree, Bisection, and Voronoi Diagrams).

• 3D DAS multi-objective optimization for proposed models:
- Experiment on 3D DAS optimization using the proposed models.
- Analyze and compare the performance of the proposed models.

The first part evaluates the state-of-the art 3D DAS model and identifies its limitations. Four different models are then proposed, each developed to overcome its predecessor's drawbacks. A summary comparison of all DAS models is given at the end of the evaluation section.

The second part of the experiments focuses on 3D DAS multi-objective optimization, using GA, with each of the proposed models. A common air traffic data sample is used in evaluation of all the models (existing and proposed). The same objectives (workload balancing, and average sector flight time) are used with all models. The performance and efficiency of each model are analyzed, compared and discussed.

3.3. Air traffic data

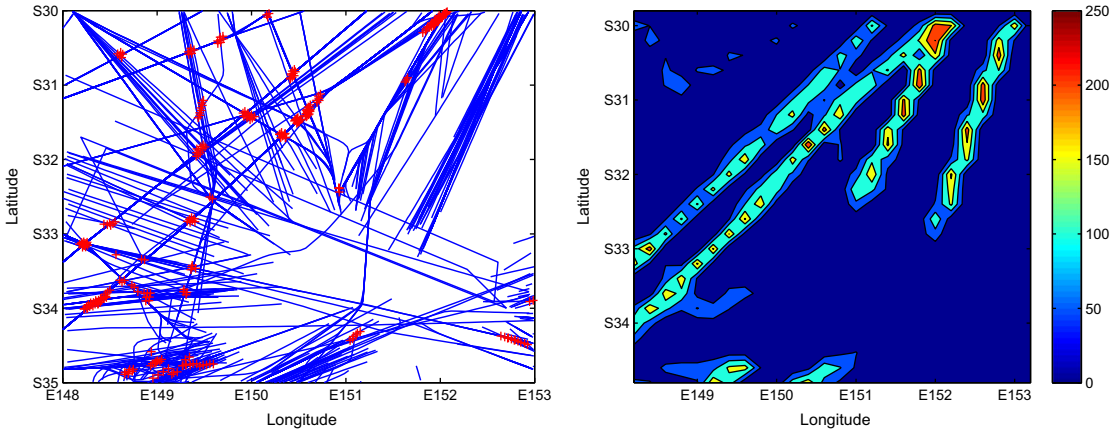
It is important to evaluate DAS approaches and algorithms with a good sample of air traffic data (high traffic feature variance like speed, heading, altitude). This may come from a historical database or a simulator.

The air traffic data used in this paper is a set of simulated flight trajectories during a fixed time window (24 h) in part of the Australian airspace. The airspace is a 5 degree by 5 degree en-route airspace (from FL200 to FL500), located in the east of Australian airspace.

The simulated flight plans are generated by a high-fidelity air traffic simulator (ATOMS) (Alam et al., 2008). They are based on public domain statistics obtained from one month of real traffic. The average and standard deviation of hourly departures for each airport pair are calculated. From these, the number of flight departures for each airport in a given period (1 h) of a day can be generated, based on a Gaussian Distribution. A Poisson Distribution for flight departure times is also built for each airport, approximated from publically available real data, and is used to generate the departure time for each simulated flight.

A total of 691 flights, including 536 domestic, 71 outgoing, 79 incoming and 5 overflying flights, was generated from the simulator. The sample contains a total of 23,619 traffic hits. The sample has 24 different aircraft, including long haul (heavy) and small (light) aircraft. A lateral view and a lateral heat-map of the flight trajectories used for evaluating the model is illustrated in Fig. 2. Fig. 2a also illustrates the crossing points of traffic flows which are identified by the 4-Dimensional simulated flight trajectories. In spatial distribution, there are 249 flights from south-west to north-east and 235 flights from north-east to south-west. The major traffic flows between the top right and bottom left are highlighted in Fig. 2b. There are also several local hot spots in the bottom and the top left corner of Fig. 2b.

The average time of a flight staying within an area is 1045.96 s (standard deviation is 787.89) and the average travel distance is 129.66 nm (standard deviation is 88.78). The distribution of the flight staying time and traveling distance are plotted in the first row of Fig. 3. The second row of the figure shows the distributions of flight directions and their maximum heading changes within 30 s. Most of them travel from north-east to south-west or vice versa which corresponds to the highlighted traffic flows shown in Fig. 2b. Most flights change their headings less than 15° and some of them change the heading around



(a) Lateral View of Flight Trajectories and Traffic Flow Crossing Points

(b) Lateral Heat-map of Flight Trajectories

Fig. 2. Lateral view and heat-map of flight trajectories and Traffic Flow Crossing Points in the given area.

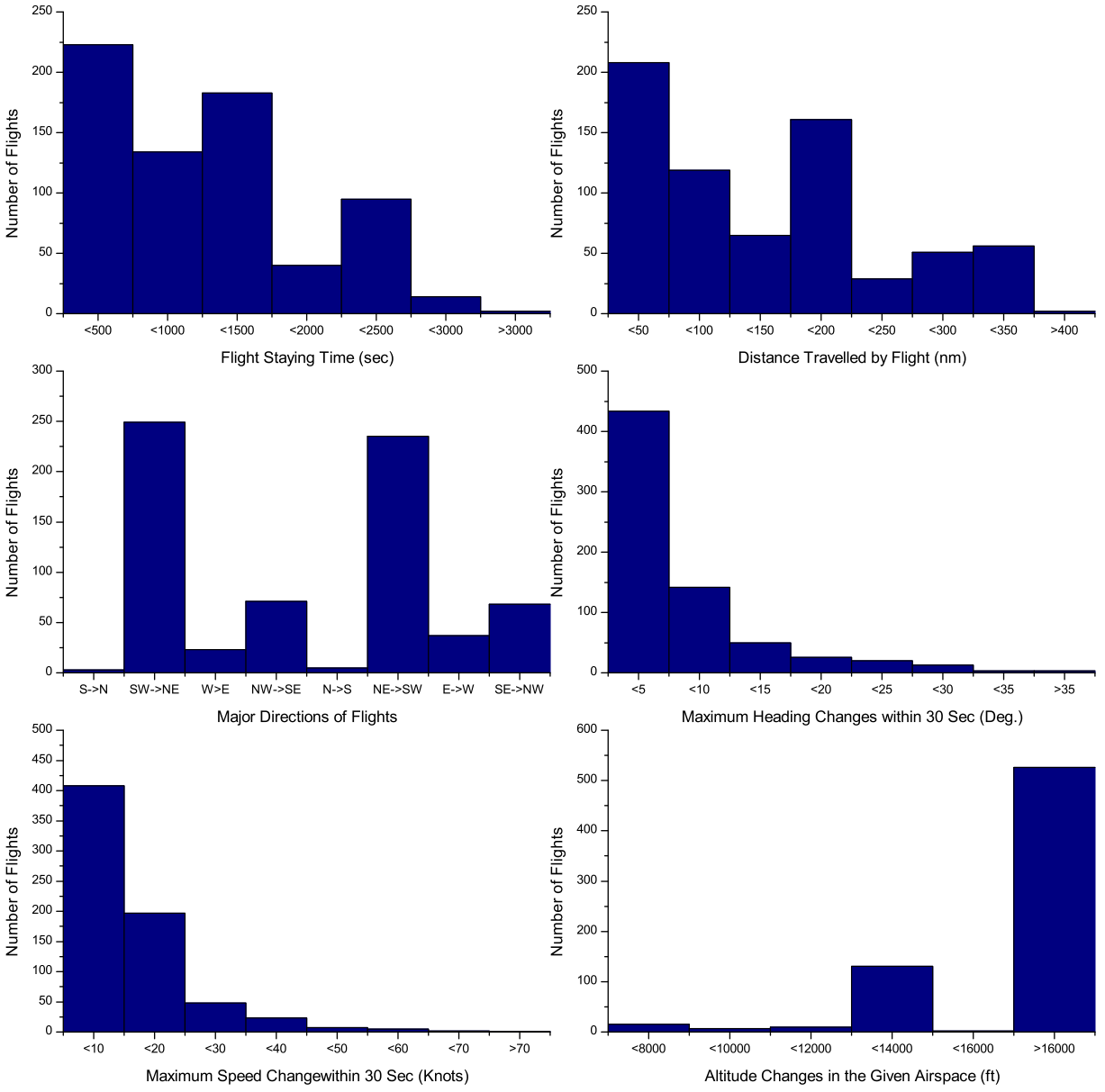


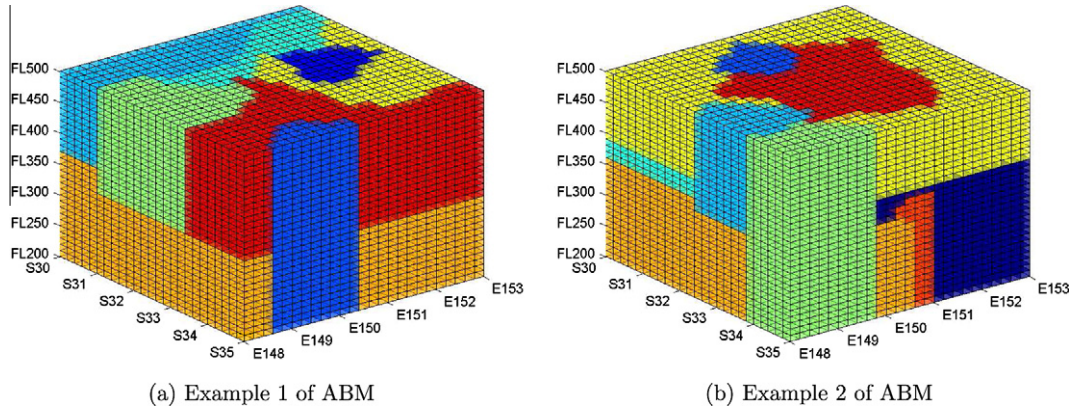
Fig. 3. Simulated flight trajectories spatial characteristics distributions.

30° within 30 s. The distributions of the flight speed changes and altitude changes are also visualized in the third row of Fig. 3. An airport is located below the given airspace, therefore, a large amount of flights change their vertical profile when flying in this area. The simulated traffic is dynamic in both horizontal and vertical dimensions, which affects the airspace sectorization methods, especially the 3D sectorization methods.

4. Evaluation of the agent based model for 3D airspace sectorization

We implemented the ABM model for 3D airspace sectorization including the four agent rules described in Section 2.2. We then evaluated it for a section in Australia airspace, using the air traffic data sample described in the previous section.

According to the Movement Rule and Layer Growth Rule, agents group the cells which increase the sector's commonality by the greatest amount without exceeding the workload upper bound (1.2 times of average workload). However, there is no clear Movement Rule definition for an agent that is not surrounded by any traffic cells. Therefore, an agent groups all empty neighboring cells of its current sector configuration.



(a) Example 1 of ABM (b) Example 2 of ABM

Fig. 4. Two examples of 3D sectorization results by ABM with random agent initial locations.

We initialized 10 agents with random locations, and then activated the agents to group cells iteratively according to a predefined movement ratio (0.2). Two examples of the 3D sectors produced by the ABM are visualized in Fig. 4.

The experimental results shows that the ABM can produce a 3D sectorization solution, with the given objectives of maximizing traffic commonality and balancing sectors workload. However, right prism violation reported in Kicinger and Yousefi (2009) is found in our experiments as well. In addition, we see other problems with this approach in terms of sector design:

- Sector nests inside another sector.
- Sector shape does not satisfy the convexity constraint.
- High computation cost.

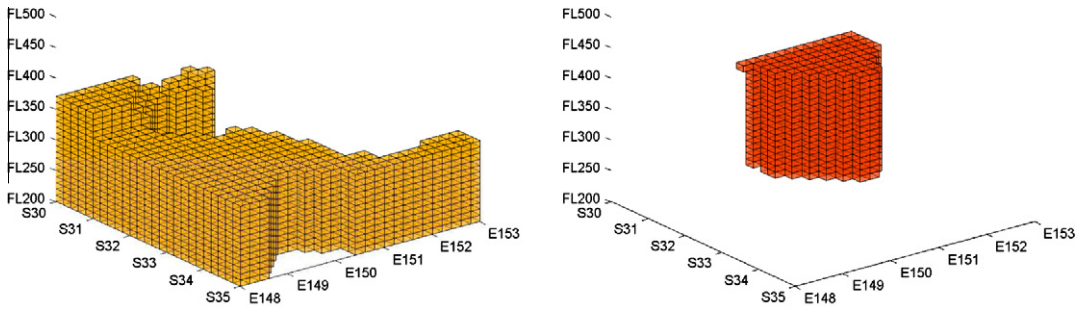
The next sections examine these limitations and attempt to identify their causes.

4.1. Right prism polygon violation

The right prism polytopes for sectors cannot be guaranteed by ABM. Fig. 5 shows two examples of the sector polytopes generated by ABM that do not meet the right prism constraint.

The right prism constraint requires that if one of the vertical neighbors and one of the horizontal neighbors of a cell belong to the same sector, the cell itself has to be in the same sector. For example, on the left of Fig. 6, the cell i is assigned to sector a while its vertical and horizontal neighbors j_1 and j_4 belong to sector b, which violates the right prism constraint.

In the shown example, the Repair Rule tries to resolve the right prism violation by reassigning either the neighbor or the cell itself to other sectors, without breaking the contiguity of the sectors involved. This is achieved in one of three manners (shown on the right of Fig. 6): to reassign the cell i to sector b, or to reassign either the cell j_1, or to reassign the cell j_4 to sector a. The selection of the reassigned cell depends on the contiguity check of its owner sector and the workload limits. After reassignment, the cell i is checked to ensure it meets the right prism constraint. If necessary, the repair rule repeats these steps until the violation is resolved.



(a) Example of Sector 3 (b) Example of Sector 4

Fig. 5. Two examples of right prism violation generated from ABM.

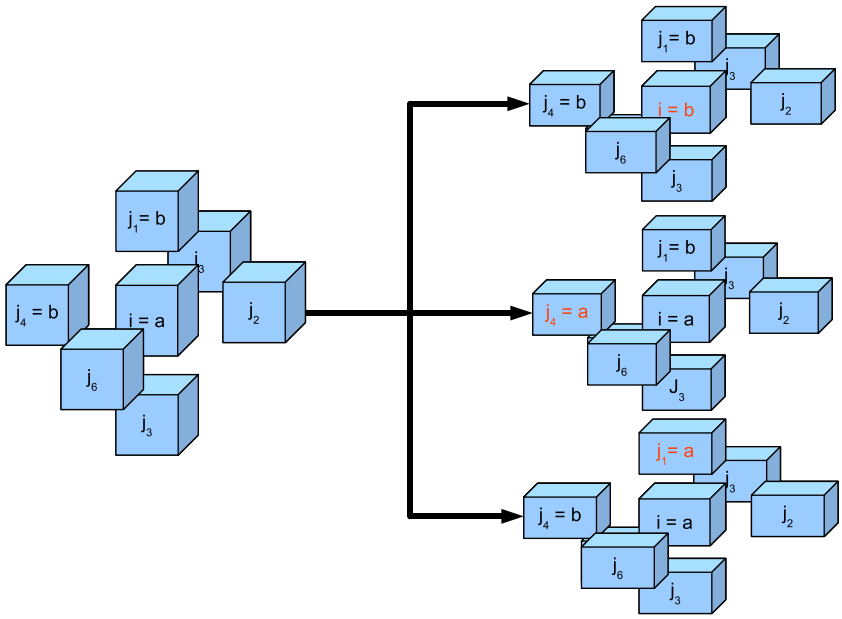


Fig. 6. Right prism violation and possible solutions by repair rule in ABM.

It is possible that no solution can be found. If the cells j_1 and j_4 belong to sector b and the cells j_2 and j_3 are assigned to sector a, the cell i cannot be assigned to either a or b. The Repair Rule bounces cell i between these two sectors, never getting a solution satisfying right prism for both sectors.

In the ABM model, the Repair Rule and right prism check are applied at the cell level, not at the geometric plane level, which implies that the sector cannot satisfy the right prism constraint. Therefore, although the maximum level of right prism constraint check and Repair Rule is applied in ABM, the violation of right prism cannot be eliminated.

4.2. Embedded sectors

We found some instances where the ABM method produced sectors embedded inside other sectors, as shown in Fig. 7d. The reason is that the movement ratio for agents causes non-uniform growth of agents. Embedded sectors is not a practical outcome.

The Movement Ratio in this model decides which agents to be activated at each iteration. The Movement Rule and Layer Growth Rule are executed by only the activated agents that have least accumulated workloads at each iteration. Therefore, the privilege of movements for each agent is not equal at each iteration. Some agents having lower accumulated workload may grow faster than others, while some agents having higher accumulated workload may not grow at all during some iterations.

If the locations of agents are close to others, and some have greater growth rate at the beginning, the slower agents may be surrounded by the faster growing agents. There is then no space left for future movements of the slower agents. In such cases, some agents stop growing before reaching their capacity limits.

An example is shown in Fig. 7. In Fig. 7a and b, the blue agent is growing much faster because it has lower accumulated workload than other agents. By iteration 12 (Fig. 7d), the red and light blue agents are surrounded by the blue agent. There are no unassigned cells around the red and light blue agents, so neither has any space to grow, even if their workloads are much lower than other agents in the future.

Randomly assigned initial positions of agents contributes to this problem. Embedded sectors are most likely to arise when the initial positions of agents are close to each other. Since there is no mechanism to limit the activities of agents whose growth rates are much higher than others in the ABM, instances of embedded sectors cannot be avoided in this implementation.

4.3. Convexity of sector shape

Requiring a sector's lateral shape to be convex can avoid the situation where flights enter the same sector more than once. The convex shape is not considered in ABM: this grid based approach does not have any constraint check on the convexity and it cannot guarantee the convexity of sectors. This is a serious limitation of the ABM approach when it comes to field testing of DAS with realistic air traffic scenarios.

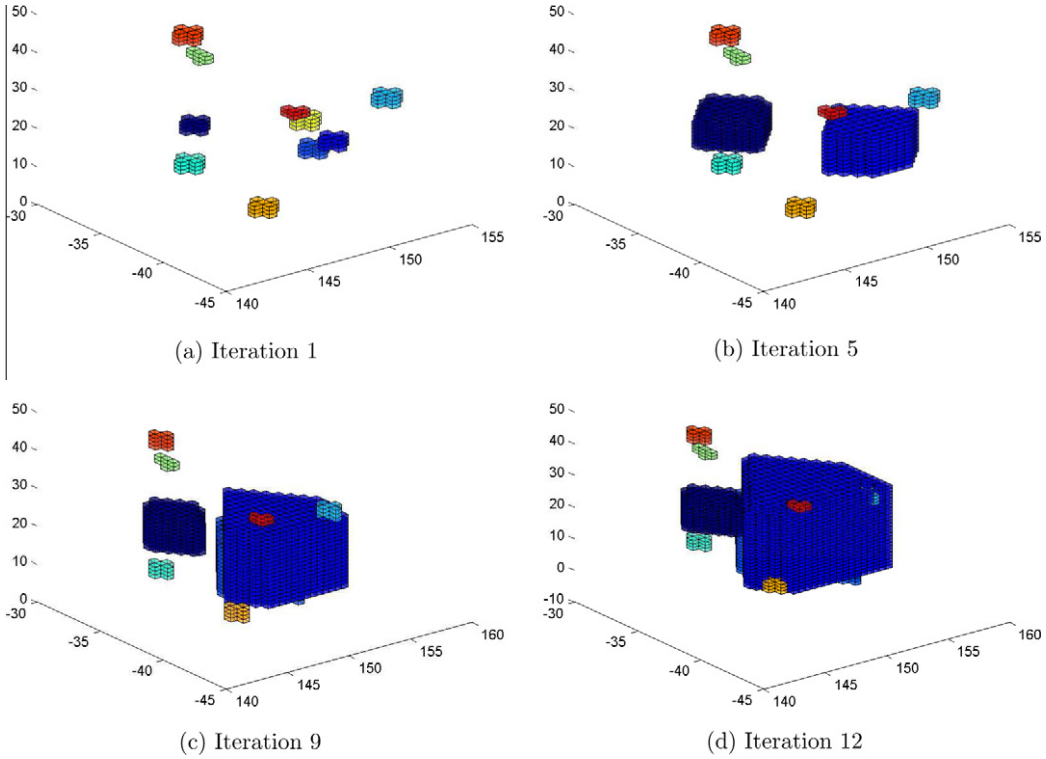


Fig. 7. An example of embedded sectors caused by unbalanced growth of agents in ABM.

4.4. High computation cost

The Movement Rule and the Layer Growth Rule have to identify the agent's boundary cells and their unassigned neighbors whenever they are executed by an agent. The worst case is to enumerate each cell of a sector to find its boundary cells. This has a computational complexity of $O(N)$, where N is the number of cells of an agent. The unassigned neighboring cells then can be found by the boundary cells; this has a complexity of $O(N_b)$ where N_b is the number of boundary cells of an activated agent. Then, the right prism check on each neighboring cell, and searching for a neighboring cell with the maximum cumulative commonality, are executed; this has a complexity of $O(N_n)$ where N_n is the number of neighboring cells of an agent. Hence, the time complexity of the Movement Rule is $O(N + N_b + N_n)$ for an activated agent.

The Layer Growth Rule has to group the neighboring cells into vertical and horizontal layers. This can be executed at the same time as the right prism checking. It also has to find a layer with the maximum cumulative commonality, which has an additional time complexity of $O(N_v + N_h)$ where N_v is the number of cells in all vertical layers and N_h is the number of cells in all horizontal layers. $N_v + N_h$ equals N_n because all vertical and horizontal layers are made from the neighboring cells. Therefore, the total time complexity of the Layer Growth Rule is $O(N + N_b + N_n)$ for an activated agent.

The Trading Rule is called by an agent when its workload exceeds the upper boundary. The Repair Rule is executed when some cells of a sector violate the right prism constraint. Both rules exchange an agent's cells with another agent without breaking sector shapes. Therefore, a "sector contiguity check" is necessary when the Trading and Repair rules are invoked. A grid-based sector can be treated as an undirected graph: the cells in a sector are the vertices (V) and the links between neighboring sectors are the edges (E) of the graph. To check the contiguity of a sector is the same as to check the contiguity of a graph. Either Breadth-First-Search (BFS) or Depth-First-Search (DFS) can be used. The cost for both BFS and DFS is $O(|V| + |E|)$. In ABM, each cell has 6 edges at most; therefore the computational complexity of the sector contiguity check can be considered as $O(N)$ where N is the number of cells in a sector.

The Trading rule requires finding an exchangeable cell of a sector without breaking its contiguity. The worst case is to check each cell in a sector: this costs $O(N^2)$. The Trading Rule is used to hand over a cell from one agent to another, so the contiguity check is applied on one agent only. Therefore, the total time complexity of the Trading Rule is $O(N + N_b + N^2)$ for an agent, including finding its boundary cells for exchanging.

The Repair Rule exchanges cells between two agents in order to satisfy the right prism constraint. It has to check the sector contiguity for both agents (k and s) involved, costing $O(N_k^2 + N_s^2)$, where N_k and N_s are the numbers of cells of agents k and s respectively. The right prism check is also enforced after the exchange, with cost $O(N_k + N_s)$. The total complexity of the Repair Rule is therefore $O(N_k + N_k^2 + N_s^2 + N_s)$ for two agents involved in the Repair Rule.

Table 1 summarizes the computational complexity for each agent rule.

As shown in Table 1, the computational complexity of the Movement Rule and Layer Growth Rules increases linearly with N as the number of cells in a sector and the number of its neighboring cells increase, but the complexity of the Trading Rule and Repair Rule increase with the square of N as the number of cells in a sector increases. The Movement Rule and Layer Growth are executed at each iteration when the agents are activated. The other two rules are called when workload balancing or right prism constraint are violated.

The above analysis applies to each agent, in each iteration. Thus the overall complexity of ABM is $O(S \times (N + N^2) \times I)$, where S is the number of agents and I is the number of iterations. The number of agents is a constant number when the expected number of sectors is decided, but the number of iterations and the number of times each rule is invoked vary depending on the agent initial locations, their movements, etc.

To investigate the cost of each rule, we ran ABM with 10 different initial agent positions to group 20,956 cells into 10 sectors for the given airspace and traffic data described in Section 3.3. Table 2 listed the total time taken by each agent rule and the total number of times each rule was invoked during these 10 runs.

As shown in the table, around 91% of the computation time was spent by the Repair Rule, and around 4% of the computation time was taken by the Trading Rule. The Layer Growth and Movement Rules, including right prism checks after grouping cells, only accounted for around 5% of computation time. The Repair Rule was invoked most often, because the right prism is frequently violated by the Movement Rules and Trading Rules. The Trading Rule is invoked after agents stop grouping more cells. At this time, each agent reaches the maximum number of cells which it can group. The Trading Rule has to check the contiguity of two sectors at the same time, therefore, time spent by the Trading Rule on each invocation is the most expensive.

All of these show that the computation in ABM is dominated by the Repair Rule. It is the most frequently invoked rule, and its cost per invocation is higher than the other rules (except for the rarely-invoked Trading Rule). Its computational complexity is $O(N^2)$, so the computational cost of ABM rises with the square of the number of cells in a given airspace. This shows that ABM is not an efficient model for airspace sectorization, especially when the size of the airspace is increasing.

4.5. Summary

The limitations of the agent based model, their causes and possible solutions are summarized in Table 3.

First of all, the right prism violation prevents the ABM from being used in practices. The sectors violating right prism constraint cannot be projected on a 2D ATC screen for air traffic management. Secondly, the embedded sectors and non-convex sector shapes generated by ABM allow flights to enter the same sector more than once which increases the ATC workload on

Table 1
Computational complexity of the agent rules in agent based model.

Agent rules	Computation complexity
Movement Rule	$O(N + N_b + N_n)$
Layer Growth Rule	$O(N + N_b + N_n)$
Trading Rule	$O(N + N_b + N^2)$
Repair Rule	$O(N_k + N_k^2 + N_s^2 + N_s)$

Table 2
Execution time and number of times invoked, of ABM agent rules in 10 runs.

Rules	Layer Growth Rule	Movement Rule	Repair Rule	Trading Rule
Time (ms)	21277.71	93.51	396107.39	15001.75
Number of times invoked	5937	143	66,063	18
Time (ms) per invocation	3.58	0.65	6.00	833.43

Table 3
Summary of limitations existing in ABM and possible solutions for 3D airspace sectorization.

Limitations	Causes	Possible solutions
Right prism polygon violation	Limitation of Repair Rule and cell level based right prism constraint check	New Rules for both agent growth and repairing violation of right prism constraint
Embedded sectors	Agent movement ratio and Growth Rules	New Rules for both agent growth
Convexity of sector shape	No convexity constraint applied	Introducing the convexity check
High computation cost	Sector contiguity check by Repair Rule and Trading Rule	Remove the contiguity check of sector without breaking sector contiguity

handover. Finally, the computation cost of ABM is high for an operational ATM environment where an efficient DAS approach is necessary to re-sectorize the airspace in order to accommodate the fluctuation of air traffic demands. Therefore, these limitations of ABM make it unusable in practice.

5. An improved Agent Based Model (iABM) for 3D airspace sectorization

The limitations of the ABM, their causes, and possible solutions provide a foundation for the development of an improved Agent Based Model (iABM), with the aim of addressing the limitations of the current model. The main objectives of the proposed model are to make the sector shapes satisfy the right prism constraint and to solve the embedded sector problem. Reducing the computation cost is another objective of iABM. Maximizing traffic commonality of sectors and balancing controller workload are also considered.

In the proposed iABM, instead of the Movement Rule and Layer Growth Rule, there is only one rule (Growth Rule) which is responsible for agents grouping neighboring cells. The Growth Rule groups cells based on traffic commonality and workload; in order to satisfy the right prism constraint, cells are only grouped if they belong to the same geometric plane horizontally or vertically.

There is no Repair Rule in iABM because all sectors satisfy the right prism constraint during their growth. However, because the sectors grow by geometric planes, some gaps (unassigned cells) may be left between sectors when agents cannot group any cells because of the right prism constraint. Therefore, a Gap Filling Rule is executed at the end of agents' movements. It reorganizes sector boundaries, and assigns unassigned cells at the geometric plane level to neighboring sectors. Since the main objective of iABM is to generate sectors meeting the right prism constraint, the Trading Rule, which can lead to violations of the right prism constraint, is not used in iABM.

5.1. Growth Rule

The Growth Rule is executed by agents to group a set of their neighboring cells, which belong to a same feasible geometric plane, in order to maximize their traffic commonality within the predefined workload range. It always applies at a plane level instead of at a cell level. The Growth Rule includes the following steps:

1. Find all neighboring and not-yet-assigned cells for sector S_k.
2. Group these cells into horizontal or vertical geometric planes according to the current sector configuration.
3. From the list of geometric planes, filter out any that violate the right prism constraint.
4. Calculate the workload for each remaining geometric plane in the list, and filter out those whose accumulated workload exceeds the upper bound of the predefined workload range.
5. From the remaining geometric planes, select the optimal geometric plane: one that most increases the sector's accumulated commonality.
6. All cells belonging to the optimal geometric plane are marked as assigned and are added into the agent's collection of cells
7. The workload associated with all cells in the optimal geometric plane are added to this agent's accumulated workload.

Algorithm 1 in the Appendix describes the Growth Rule in detail.

The feasible geometric planes generated by the Growth Rule can be classified into two catalogs:

- A horizontal layer is a plane directly above or below the current sector's top or bottom layers. The lateral (latitude and longitude) shape of a valid layer is exactly the same as the lateral configuration of a sector.
- A valid vertical block is a group of cells having the same latitude and longitude (only one pair of them) coordinates beside the current sector boundary, with all altitude levels existing in the sector.

The horizontal layers make a sector grow vertically and the vertical block make a sector grow horizontally without breaking the right prism constraint. For example, if the current sector shape is as shown in Fig. 8a, the blue[1] geometric planes in Fig. 8b are valid horizontal layers and the yellow blocks are valid vertical blocks.

The movement ratio is still applied for this Growth Rule. Embedded sectors cannot occur under the Growth Rule because the geometric plane level growth prevents an agent from surrounding others both vertically and horizontally.

5.2. Gap Filling Rule

Based on the Growth Rule, agents create sectors at the level of geometric planes. The sector shapes are guaranteed to satisfy the right prism constraint. However, some cells may be unable to be assigned to any agent under this rule. Therefore, a Gap Filling Rule is needed to reorganize the sector boundary for these unassigned cells. This rule also works at the geometric plane level. It is executed only when unassigned cells exist in the airspace after all agents finish their activities. The steps involved in this rule are:

[1] For interpretation of color in Figs. 2–13 and 16–22, the reader is referred to the web version of this article.

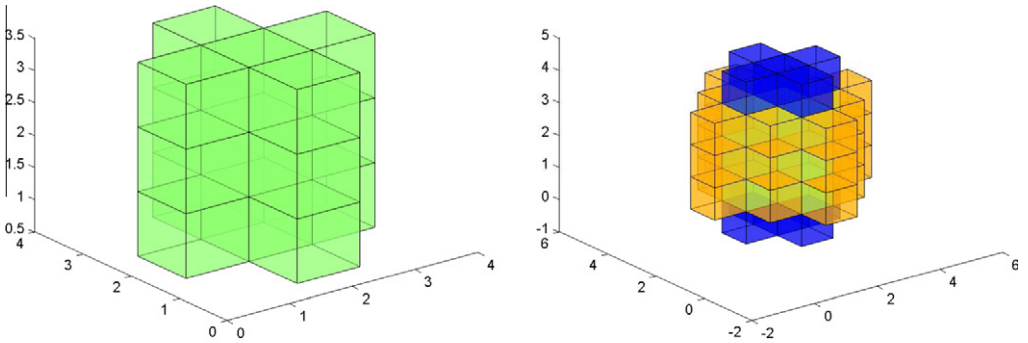


Fig. 8. Possible horizontal and vertical geometric planes for a sector generated by growth rule in iABM.

1. Find all unassigned cells (*UC*).
2. Find all agents directly above or below these cells.
3. If the number of agents above the unassigned cells is less than the number of the agents below the unassigned cells, the agents above the unassigned cells will act to group these cells; otherwise, the agents below the unassigned cells will act. This keeps the number of affected sectors as low as possible.
4. For each activated agent(S_k):
   (a) Identify a horizontal layer(L) corresponding to the agent's sector lateral configuration that contains these unassigned cells.
   (b) Find all other agents (S_j) occupying the cells (*OC*) belonging to this layer(L).
   (c) Identify the horizontal layers(L') containing the cells (*OC*) which belong to the agents (S').
   (d) Agents(S_j) release these horizontal layers (L') and recalculate their workload and commonality.
   (e) Agent(S_k) adds the horizontal layer(L) into its collection and recalculates its workload and commonality.
5. Repeat steps 1–4 until there is no unassigned cell in the airspace.

Algorithm 2 in the Appendix describes the Gap Filling rule in detail.

The Gap Filling Rule may take several iterations to assign the unassigned cells. In some cases, some sectors may end up being deleted by this rule. It is complicated to fill the gaps between sectors in 3D and to satisfy the right prism constraint at the same time. Section 8 describes the integration of iABM and an optimization algorithm to address this problem.

In summary, the Growth Rule is used for agent growth and the Gap Filling Rule is used when unassigned cells exist in the airspace. The Growth Rule groups cells to maximize a sector's commonality with consideration of workload balancing. The Gap Filling Rule fixes the problem of unassigned cells caused by the Growth Rule. Both of these rules work at a geometric plane level in order to meet the right prism constraint, which is the major objective of our agent based model. Neither rule requires a sector contiguity check, saving some computation cost compared to ABM.

5.3. Evaluation of the improved Agent Based Model

We evaluate iABM using the same input data and configurations as in our evaluation of ABM in Section 4. Ten agents are initialized with the same locations used with ABM. The airspace sectorization of two Examples is visualized in Fig. 9. Comparing Fig. 9 with Fig. 4 shows that problems such as right prism violation and embedded sectors that arise with ABM are solved by iABM.

Beside these problems, there is another problem (high computation cost) existing in ABM model. The number of cells in an airspace affects the computational cost significantly. The Growth Rule includes finding the neighboring cells, right prism check, and group cells into vertical and horizontal planes. It has time complexity of $O(N + N_b + N_n + N_v + N_h)$, where N is the number of cells in a sector, N_b is the number of boundary cells of a sector, N_n is the number of neighboring cells of a sector, N_v is the number of cells of all vertical planes, and N_h is the number of cells of all horizontal planes. All vertical planes and horizontal planes are made from neighboring cells, hence, $N_v + N_h$ equals N_n and the complexity can be considered as $O(N + N_b + N_n)$ for an activated agent.

The Gap Filling rule works on an activated agent (to group unassigned cells) and other affected agents (to release occupied cells) at the same time. The worst case is that all other agents are affected and the number of affected agents is $S - 1$ (S is the total number of agents). In this case, the complexity of checking the boundaries and releasing occupied cells from each of these sectors is $O((S - 1) \times N_j)$, where N_j is the number of cells in an affected agent. To reassign cells to the activated agent and check its boundaries is $O(N_k)$, where N_k is the number of cells in the activated agent. The complexity for the Gap Filling Rule working for one activated agent is $O(N_k + (S - 1) \times N_j)$ which can be considered as $O(N_T)$ where N_T is the total number of cells in the given airspace. The Gap Filling Rule works on the agent level not on cell level. In the worst case, the total

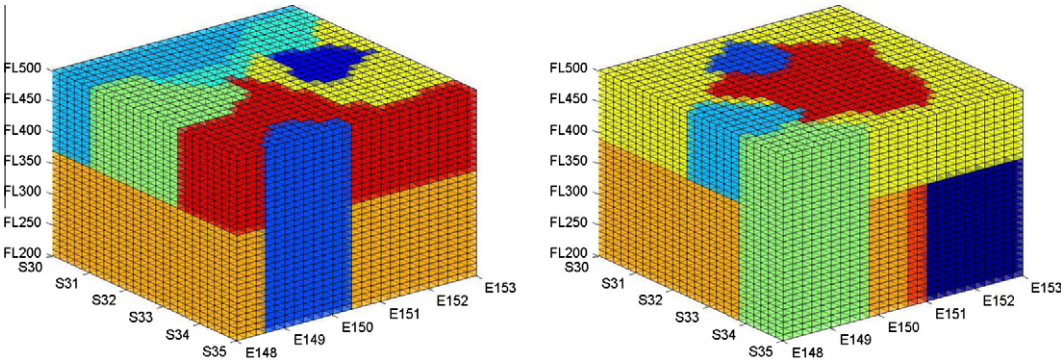


Fig. 9. Two examples of 3D sectorization results by iABM with random agent initial locations.

computational complexity of the Gap Filling Rule to fill a gap is $O(S \times N_T)$ when all agents are activated by it. The computation complexity related to each agent rule of iABM is summarized in Table 4.

The Growth Rule is executed at each iteration by the activated agent, but the Gap Filling Rule is called only after all agents stoping moving and unassigned cells exist in the airspace. Both rules show the linear relationship between the computational complexity and the number of cells, which save a lot of computational cost compared with ABM. However, the Gap Filling Rule may be executed several times until all gaps are filled. This affects the total complexity of iABM dynamically, especially for the agents with randomly initial positions. We tried ten runs of iABM with the same agents initial positions, airspace configuration, and air traffic data as being used by ABM to investigate the cost of each iABM agent rule. Table 5 shows the total time spent by each agent rule and the total times of invoked by iABM in these ten runs.

The Gap Filling Rule is the most expensive per invocation, but it is invoked fewer times than the Growth Rule. From the results of these ten runs, the time spent by the Gap Filling Rule is from 0% to 25% of total execution time, depending on whether gaps exist or not. Therefore, the relationship between total complexity of iABM and the number of cells may not be linear.

To compare the computation cost between the two agent based models, we evaluated the computation cost (processing time) of iABM and compared it against ABM, using different numbers of cells in the same airspace. We investigated 10 different cell sizes, giving 10 different numbers of cells in the airspace. For each number of cells, we initialized 10 sets of random positions for agents, giving 100 configurations in total. The same configurations were executed with both ABM and iABM. The other input data, such as flight trajectories and airspace boundary, were the same in each run. All runs were done on the same machine, equipped with an Intel Core2Duo 3.0 GHz CPU, 4 GB RAM and Windows XP.

The processing times for the 10 runs with a given number of cells was averaged, for each number of cells and for each model. The results are illustrated in Fig. 10.

It shows that the computation cost for both models increased non-linearly as the number of cells increased because ABM has the computational complexity $O(N^2)$ while iABM is affected by the Gap Filling Rules dynamically. The magnitudes of computation cost are very different. The minimum average time to finish sectorization (3751 cells) with iABM is 0.05 s while with ABM it is 3.77 s. The maximum average time to finish sectorization (34,596 cells) with iABM is 2.71 s while with ABM it is 1167.3 s. This shows that ABM lacks of scalability for large airspace. It also shows that iABM has significant computation cost savings compared to ABM.

Table 4
Computational complexity of the agent rules in the improved Agent Based Model.

Agent rules	Computation complexity
Growth Rule	$O(N + N_b + N_n)$
Gap Filling Rule	$O(S \times N_T)$

Table 5
Execution time and number of times invoked, of iABM agent rules in 10 runs.

Rules	Growth Rule	Gap Filling Rule
Time (ms)	860.58	109.53
Number of times invoked	2921	31
Time (ms) per invocation	0.29	3.53

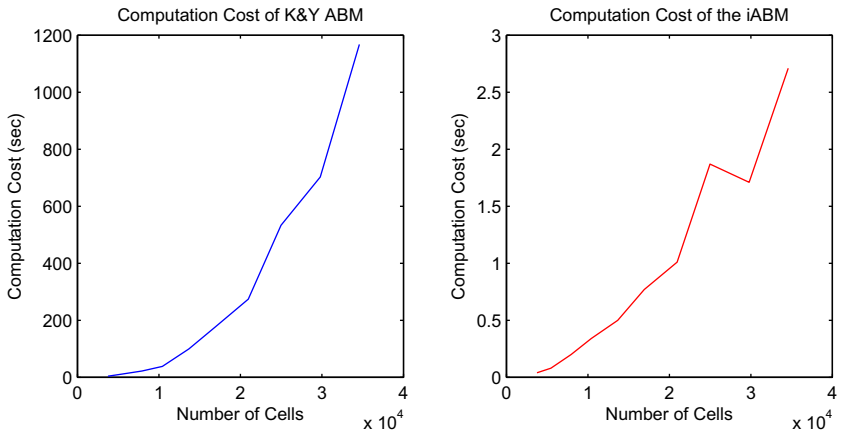


Fig. 10. Computation cost of Kicinger and Yousefi ABM and the improved ABM.

5.4. Limitations of the improved Agent Based Model

Although iABM eliminates some problems of ABM, such as right prism violation, and the computation cost is greatly reduced, two limitations remain. First, the convexity constraint still cannot be guaranteed (examples can be seen in Fig. 9). Second, the computation cost is still high for processing a large airspace, especially when optimization is required.

6. Geometry-based 3D airspace sectorization methods

Both iABM and ABM are grid based models, thus the sector shapes are all cuboid when the convexity constraint is applied. All cuboid sectors are not practical in reality. Therefore, it is not worth introducing a new complex agent rule for convexity check. Hence, to address the convexity constraint we develop three more models, for 3D Airspace Partitioning based on geometric space partition methods, which can meet necessary requirements including the convex constraint with fast processing time.

The agent based model works on the discrete airspace which is divided into uniform cells. If the convexity constraint is a prerequisite for the agent based model, it must only produce cuboid sectors. To produce cuboid sectors, agents need more rules to constrain their growth. However, there are several other efficient approaches existing in the literature, such as Octree, KD-tree, etc., to produce cuboid partitions in a space.

We investigated three more approaches, apart from agent based approaches to sectorize 3D airspace for satisfying the requirements of convex constraint and efficiency.

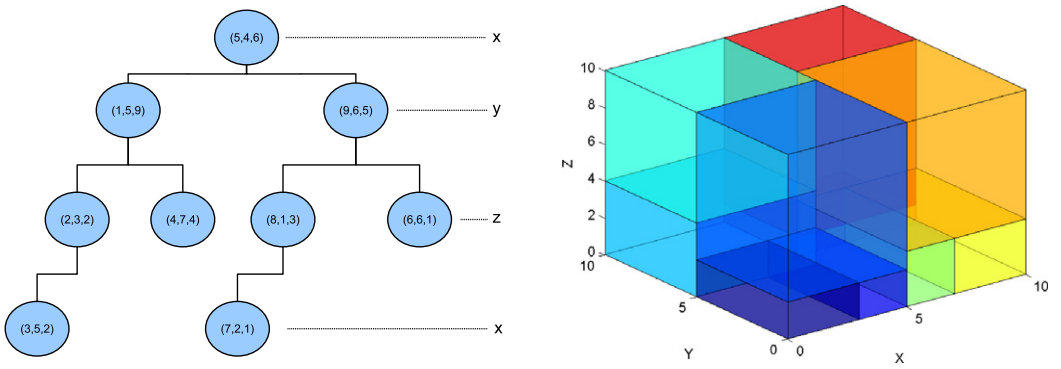
6.1. Airspace sectorization by KD-Tree based model

A KD-tree (k-dimensional tree) (Bentley, 1980) is a space-partitioning data structure. It is a binary tree where each node is a k-dimensional point. Every non-leaf node implicitly generates a splitting hyperplane, which partitions the space into two parts. The left sub-tree of that node represents points on one side of this hyperplane, and the right sub-tree represents points on the other side of the hyperplane. The hyperplane direction is chosen by the depth of every node in the tree associated with one of the k-dimensions. The hyperplane is perpendicular to the chosen dimension's axis. The computation complexity to construct a KD-tree is $O(N \log N)$ where N is the number of the given points.

We used the KD-tree based model for 3D airspace partitioning. The model partitions the airspace into cuboid sectors, depending on a set of initial points. It is easy to apply and satisfies the convex shape requirement. The detail of the model is described in Algorithm 3 in the Appendix.

An example of KD-tree based sectorization is shown in Fig. 11. In this example, the initial nine points are sorted on their x-axis value first. Then the middle point (5,4,6) is selected; it divides the space along the x-axis at its x-axis value (5). All points with x less than 5 are in the left sub-tree of point (5,4,6), and the others are the right sub-tree. At the next depth of the tree, the points are sorted by their y-values, and the middle points are selected (point (1,5,9) in the left sub-tree, and point (9,6,5) in the right sub-tree). Therefore, we get 4 sub-spaces. Three points have been selected as splitting points; the other six are grouped into the 4 sub-spaces based on their x and y values. At the next step, the same process is executed based on the z-values. The process is repeated until no further splitting is possible. The binary tree constructed in this example is shown in Fig. 11a. The corresponding 10 sub-spaces are illustrated in Fig. 11b.

As illustrated in the example, the KD-tree based model guarantees a convex shape for each sub-space, but the polytopes are limited to cuboids. This limitation may limit the KD-tree based approach's capacity to align sectors with traffic flows and



(a) Tree Structure Constructed by KD-Tree based model

(b) Space Sectorization by KD-tree based model

Fig. 11. An example of 3D space sectorization by KD-tree based model.

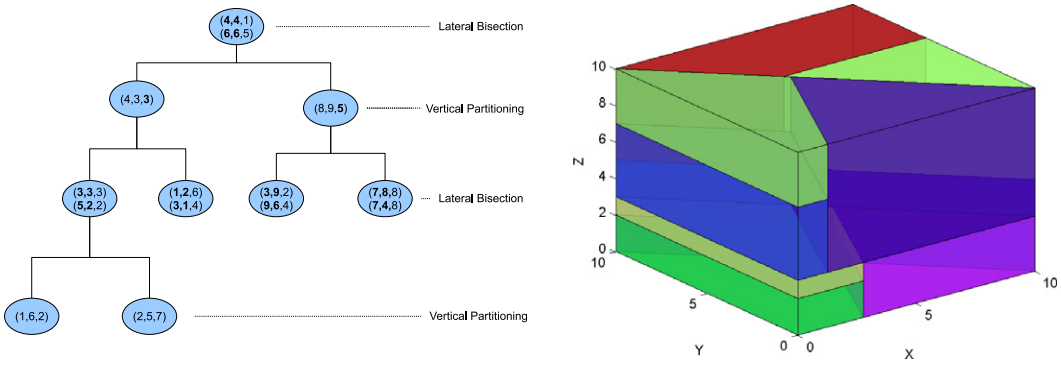
to distribute workload evenly when it is used in DAS. In order to overcome these potential problems, two more models, based on Bisection and Voronoi Diagram, which can import variety of polytope types, have been developed and implemented.

6.2. 3D airspace sectorization by Bisection

To increase the variety of sector shapes, we developed a method called Bisection-based model. It is an extension of the KD-Tree based model, still using a tree structure to store the data. Instead of using only one value along one axis to partition the space at each step, it alternates between using two values along two axes and one value on one axis. First it uses a line between two points to divide a space laterally, grouping the remaining points into the sub-spaces based on their locations. Then, it splits the sub-spaces vertically at the middle altitude value of the points in each sub-space. These two steps alternate until all points are processed. If it is a lateral partition's turn but the points in the sub-space only differ vertically, a vertical division is undertaken instead. The details of bisection are described in Algorithm 4 in the Appendix.

A simple example of the Bisection based model is illustrated in Fig. 12. In this example, the space is partitioned into 10 sub-spaces by 14 points. First the points $(4,4,1)$ and $(6,6,5)$, which are closest to the middle of the given space, are selected. The space is bisected by the (x,y) values $(4,4)$ and $(6,6)$ of these two points. The remaining points are grouped into these two sub-spaces. The second step divides the sub-spaces by the points having the middle z-values. These two steps are repeated alternately until all points are processed.

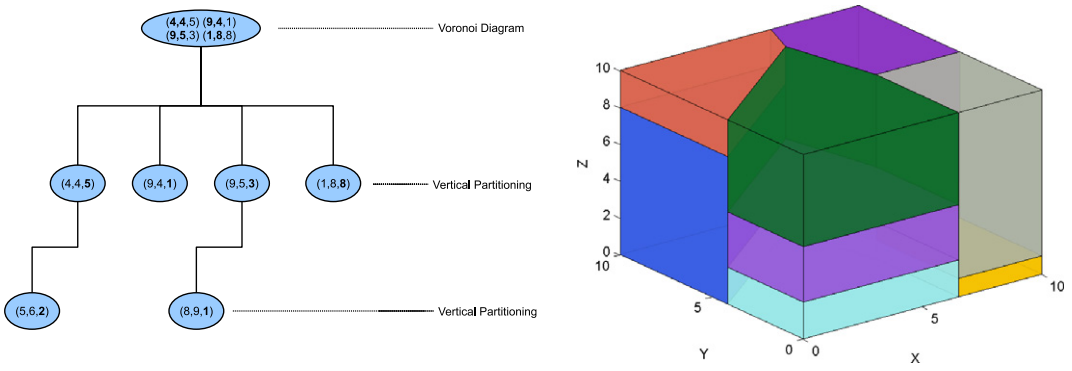
Bisection of a geometric plane only generates convex shapes, so the convexity constraint is always satisfied. Although the computation cost of the Bisection-based model is more than for the KD-tree based model, because of the additional geometry calculations, it is still faster than agent based models. Furthermore, the Bisection-based model introduces more polytopes for



(a) Tree Structure Constructed by Bisection based Model

(b) Space Sectorization by Bisection based Model

Fig. 12. An example of 3D space sectorization by Bisection based model.



(a) Tree Structure Constructed by Voronoi Diagram based Model

(b) Space Partitioned by Voronoi Diagram based Model

Fig. 13. An example of the 3D space partitioning by Voronoi diagram based model.

the sector shapes to capture air traffic flows, which is a big advantage over the KD-tree based model. But the computation complexity is higher than KD-tree. The computation complexity to get two points closest to the centroid is $O(N)$, N is the number of given points in a space. The whole construction of the Bisection based model, it is $O(S \times N)$ where S is the total number of the expected sectors and N is the number of points inside space.

6.3. 3D airspace sectorization by Voronoi Diagram-based model

A Voronoi Diagram-based model for 3D airspace sectorization is also developed to introduce more diversity of sector shapes. A Voronoi Diagram is a special kind of decomposition of a metric space, determined by distances to a specified discrete set of objects (sites) in the space. A Voronoi Diagram guarantees convexity for each Voronoi cell. The optimal Voronoi Diagram algorithm (Fortune's Sweep Line Algorithm) (Fortune, 1987) is applied in this model.

In this model, a tree structure is again used during the processing. The inputs are a list of points (P) and a predefined number of sites N which is larger than the list size. The space is firstly decomposed into N Voronoi Cells laterally by a sub-list of points (of size N) chosen from the points list (P). Then each Voronoi cell is divided vertically into two sub-spaces (lower and upper) by its site's altitude. Therefore, N points can produce $2 \times N$ sub-spaces. The remaining points are grouped into these sub-spaces according to their locations. The sub-spaces are further decomposed similarly. If there is only one point in the point list, only the vertical split is executed. The detail of this model is described in Algorithm 5 in the Appendix.

Fig. 13 shows an example of a given space divided into 10 subspaces, using 6 input points and with the sites number set to 4, and the corresponding tree structure.

By the nature of Voronoi Diagram, the convexity of sector shape is guaranteed. It has advantages over agent based model, such as satisfying right prism constraint, non-embedded sectors, and lower computation cost. Like the Bisection model, this model can generate different sector shapes compared to the KD-tree model. The computation complexity of Fortune's Sweep Line Algorithm is $O(N \log N)$ where N is the number of sites. Because our Voronoi based model works in a 3D space, and it divided space into sub-spaces iteratively until all given points are processed. Therefore, the number of expected sub-space affects the computational complexity. As a whole, the complexity of Voronoi based model is $O(S \times N \log N)$, where S is the number of expected sectors.

7. Summary of 3D DAS models

Table 6 lists the summary of all five models in five aspects: right prism, convexity, non-embedded sectors, shape diversity and algorithm efficiency. The shape diversity of sectors is a potential factors offering more flexibility to sectors for air traffic flow alignment and keep necessary minimum distance between sector boundaries and traffic flow crossing points.

As shown in the table, the ABM has more drawbacks than other models. iABM overcomes some limitations, such as right prism violation and embedded sectors, and takes into account ATC workload and sector alignment with traffic flow. However, it still lacks a mechanism to satisfy the convexity requirement. The computation cost of iABM is improved a lot compared to ABM, but is still not efficient enough.

The grid-based approach of ABM and iABM limits the sector shape to cuboid if convexity is required. The KD-tree based model is more efficient, and also produces cuboid sectors which satisfy the convexity constraint and avoid problems of right prism violation and embedded sectors. On the other hand, it does not directly consider ATC workload, the alignment of sector shape with air traffic flow, or the minimum distance between sector boundaries and traffic flow crossing points, and so may produce sectorizations that are not realistic in practice.

Table 6
Summary of 5 DAS models on the sector geometric design and algorithm efficiency.

Models	Right prism	Convexity	Non-embedded sector	Shape diversity	Algorithm efficiency
ABM				✓	
iABM	✓		✓	✓	
KD-Tree	✓	✓	✓		✓
Bisection	✓	✓	✓	✓	✓
Voronoi	✓	✓	✓	✓	✓

The models based on Bisection and 3D Voronoi Diagrams address the issue of cuboid sectors, which cause problems in alignment between traffic flow and sectors, minimum distance between sector boundaries and traffic flow crossing points, and workload distributions, while maintaining all other capabilities. Bisection is a special case of Voronoi Diagrams, as it uses two points to bisect the geometric plane instead of a set of points as in Voronoi Diagrams. Hence, the Voronoi Diagram based model may produce more variety of sector shapes.

Since there are no agent rules associated with the geometric space partitioning models, they are more efficient than both ABM and iABM. However they cannot go into finer details of sector boundaries according to traffic flows when it comes to partitioning the airspace, which can be handled effectively by agent based models. But this problem can be solved by the optimization method to satisfy the objectives of DAS.

8. Optimizing 3D airspace sectorization by Genetic Algorithm

In the above sections, we introduced four different approaches for 3D airspace sectorization. They focus on geometric requirements and computational efficiency. We now propose a multi-objective optimization approach, using Genetic Algorithms to optimize the airspace sectors in terms of three objectives: balancing the ATC workload, maximizing the average sector flight time, and maximizing the minimum distance between sector boundaries and traffic flow crossing points. We used the state-of-the-art multi-objective optimization algorithm NSGA-II (Deb et al., 2002).

ABM is excluded in this section, because of its inability to generate feasible solutions for sectors and its high computation cost as shown in Section 4.

8.1. NSGA-II

NSGA-II is an efficient multi-objective evolutionary algorithm based on the Pareto optimal concept. It applies non-dominated sorting to rank individuals, and uses crowding distance to maintain diversity without specifying any additional parameters. The improved sorting approach reduces the computational complexity. It also uses an elitist strategy to expand the sample space. NSGA-II is widely used in many applications to solve multi-objective optimization problems.

NSGA-II is adopted as the optimization algorithm here, as follows:

1. Randomly initialize the first population.
2. Continue the following steps until the termination condition (maximum generation reached) is satisfied:
   (a) 3D airspace sectorization by proposed models.
   (b) Evaluate the fitness functions for each individual.
   (c) Sort the population and produce offspring by crossover and mutation.
3. Represent the optimal solutions.

Fig. 14 illustrates the architecture of the proposed system, which integrates our proposed 3D DAS models and NSGA-II to optimize airspace sectorization.

8.2. Chromosome representation

The inputs to the four 3D airspace partitioning approaches are the same: a list of points specifying latitude, longitude, and altitude in an airspace. Therefore, all four proposed models use the same basic chromosome representation in NSGA-II. However, the chromosome lengths differ.

Each chromosome contains a list of triplets. Each triplet has values of latitude, longitude, and altitude for a point, as shown in Fig. 15. These triplets represent different information for different models: agent locations for iABM; partitioning points for KD-tree and Bisection based models; and Voronoi sites or vertical partitioning points for Voronoi Diagram based model.

8.3. GA operators

8.3.1. SBX crossover

For recombination, we use the standard SBX operator (Deb et al., 2007) with crossover probability $p_c = 0.9$ and a distribution index of $\eta_c = 15$. The SBX operator stands for simulated binary crossover, whose search power is similar to that of

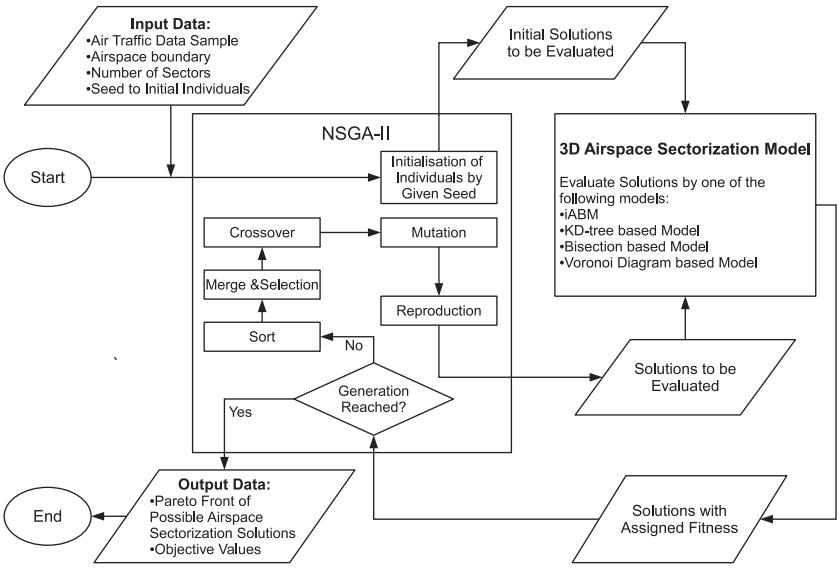


Fig. 14. Architecture of the integrated system of 3D DAS models and NSGA-II.

the single-point crossover used in binary-coded GAs. SBX is found to be particularly useful in problems like the one at hand, in which multiple optimal solutions exist with a narrow global basin, or in problems where the lower and upper bounds of the global optimum are not known a priori.

8.3.2. Mutation

Eq. (7) details the mutation operator employed by the Genetic Algorithm (see Deb and Goyal (1996)).

$$y_i = x_i + (x_i^U - x_i^L)\delta_i \tag{7}$$

$$\delta_i = \begin{cases} (2r_i)^{1/(\eta_m+1)} - 1 & \text{if } r_i < 0.5 \\ 1 - |2r_i|^{1/(\eta_m+1)} & \text{otherwise} \end{cases} \tag{8}$$

where x_i is the value of the ith parameter selected for mutation; y_i is the result of the mutation; x_i^L and x_i^U are the lower bound and the upper bound of x_i respectively, and r_i is a random number in $[0,1]$; η_m is a control parameter ($\eta_m = 20$ in our study).

8.4. Fitness function

In this DAS optimization, we specify three objectives as the fitness functions to be evolved by NSGA-II:

- Workload balancing, aiming to minimize the workload variance between sectors with a given sector number.
- Maximum average sector flight time, aiming to maximize the time for which flights stay within a sector and to align the traffic flow with sectors in order to reduce the flow cuts.
- Maximum distance between traffic flow crossing points and sector boundaries, aiming to maximize the minimum distance between the flow crossing points and sector boundaries to ensure enough responding time for ATC to resolve a potential conflict.

The first objective is modeled as the standard deviation of the sectors' workloads, as shown in Eq. (9).

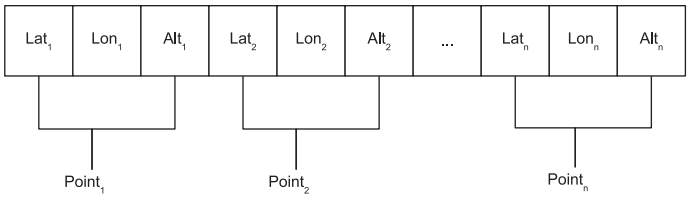


Fig. 15. Chromosome representation in NSGA-II for 3D airspace sectorization.

$$F(W) = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(W_k - W_{avg})^2} \tag{9}$$

W_k is the workload of sector S_k, which is derived from Eq. (3). W_{avg} is the average workload of all sectors in the given airspace. K is the number of desired sectors, which is predefined as an input to the optimization. W_{avg} is calculated by Eq. (10):

$$W_{avg} = \frac{\sum_{i=1, x_{ij}\in R}^{N} |t_i(x_{ij})|}{K} \tag{10}$$

The second objective is modeled as the average sector flight time of all sectors, as shown in Eq. (11). SFT_k is the sector flight time of sector S_k, produced by Eq. (4).

$$F(SFT) = \frac{\sum_{k=1}^{K} SFT_k}{K} \tag{11}$$

The third objective is modeled as the minimum distance between traffic flow crossing points and sector boundaries among all sectors, as shown in Eq. (12).

$$F(D) = \min_{i=0}^{K} D_i \tag{12}$$

where D_i is calculated by Eq. (6). $F(D)$ is the minimum distance among all sectors which excludes the distances between the crossing points and the airspace boundaries.

The first two objectives do not conflict directly, but there is implicit conflict between them. Eq. (11) shows that to maximize the flight time within a sector, the sector should have more traffic hits from the same flights. That means a sector can achieve longer sector flight time by aligning its shape with the major traffic flows, to minimize the flow cut. This may cause sectors to encapsulate as much traffic flow as possible, but this will violate the objective of balancing ATC workload. The third objective ensures that the distances between crossing points of traffic flows and sector boundaries are as far as possible. Therefore it is a multi-objective problem, and there is no straightforward way apart from multi-objective optimization to generate a trade-off set for the given objectives.

As mentioned in Section 5, the Gap Filling rule can eliminate some sectors in some cases. This problem is solved by the optimization algorithm as it attempts to minimize the workload standard deviation, which is calculated with a fixed K. If a sector is eliminated, the workload standard deviation increases; therefore, in subsequent generations such solutions are eliminated by the genetic algorithm.

8.5. Experiment design and results

We evaluate the four 3D airspace sectorization models in a given airspace with the same traffic sample which was used to evaluate the agent based model. As mentioned in Section 3.3, the airspace is a 5 degree by 5 degree en-route airspace with a 30,000 ft vertical range (from 200FL to 500FL).

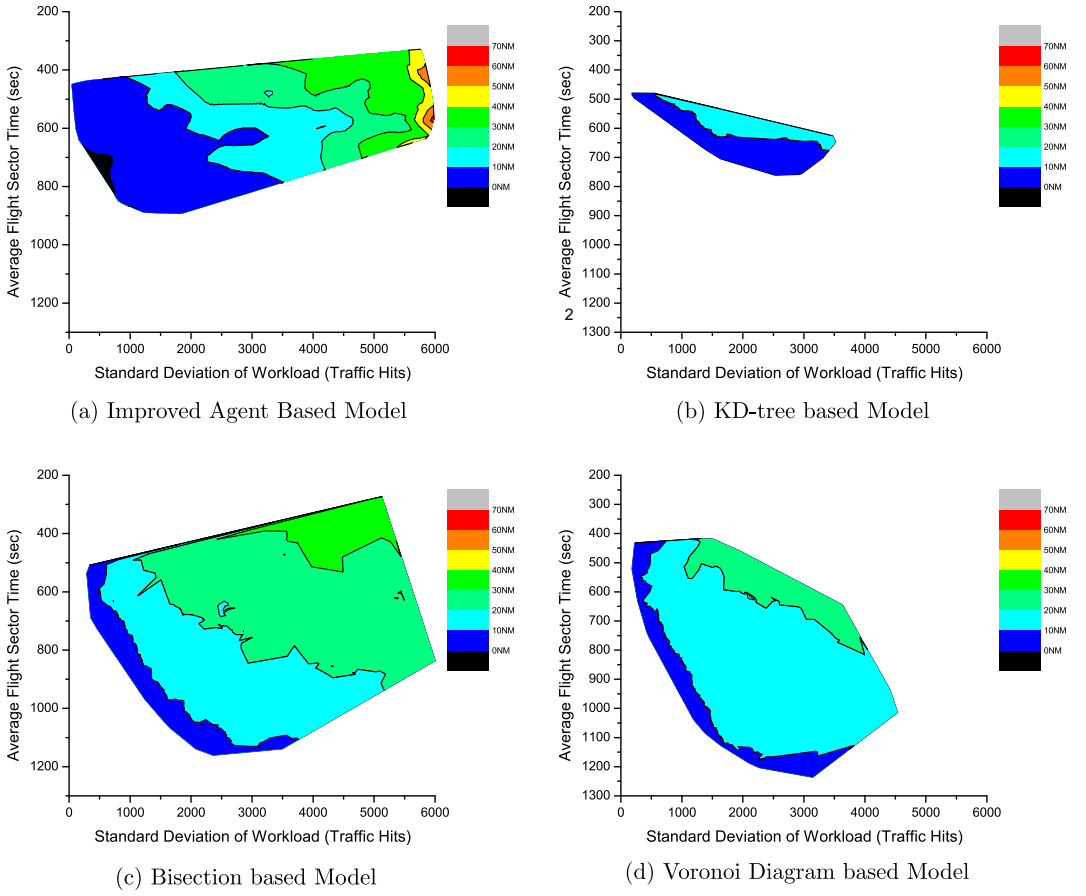
The objective is to divide the airspace into 10 sectors optimally according to the 3 objectives: workload balancing, average sector flight time, and minimum distance between traffic flow crossing points and sector boundaries. The input traffic contains a total of 23,619 traffic hits, so the average workload is 2361.9 hits for 10 sectors.

NSGA-II was used to optimize the resulting airspace sectorizations, with all four models. For each model, we create 500 random individuals as the initial population and evolve them for 500 generations.

Each model was run 10 times with the same 10 seeds. Because the three objectives specified in this paper conflict, a Pareto Front (a set of non-dominated solutions) is generated instead of a single solution. All populations from the 10 runs generated by the same model were combined; the overall Pareto Front of the standard deviation of workload and average flight sector time for each model is shown in Fig. 16. The color is mapped by the values of the third objective (minimum distance between traffic flow crossing points and sector boundaries).

The plotted figures are the best airspace sectorization solutions, which are not dominated by other solutions in terms of the three objectives, found with each model. The values of the first two objectives (workload balance and sector flight time) are plotted along with the X and Y axes respectively, and the values of the third objective (distance between crossing points and sector boundary) are mapped into color codes. As illustrated in the figure, a solution is better in terms of the first two objectives when it is closer to the bottom-left corner. The red color of a solution indicates the farther distance between crossing points and sector boundaries.

Fig. 17 shows the examples with the minimum standard deviation of workload among the possible solutions from each model. Although the air traffic data and airspace are the same for these four models, they result in strikingly different sectorizations. The next section discusses the results for each model in detail.



(a) Improved Agent Based Model

(b) KD-tree based Model

(c) Bisection based Model

(d) Voronoi Diagram based Model

Fig. 16. Populations and Pareto Fronts of 4 airspace sectorization models.

9. Analysis of results

As illustrated above, all models are able to generate airspace sectors in order to optimize three objectives, but their outputs and performance are quite different from each other. In this section, we analyze the results for each model in terms of performance and computation cost.

9.1. Performance analysis

To compare these four methods, only the solutions having an acceptable workload standard deviation of sectors are investigated. The acceptable workload standard deviation is calculated from the average workload and a coefficient ε_w, as shown in Eq. (13):

$$W_e = W_{avg} \times \varepsilon_w \tag{13}$$

In our experiments, ε_w is set as 0.2. As the average workload is 2361.9, the acceptable workload standard deviation is 472.38. Therefore, the possible solutions from all models are plotted in Fig. 18.

As shown in Fig. 18a, the iABM can achieve the best workload balancing but has no good results for average sector flight time when being compared with the Bisection and Voronoi models. Both the Bisection and Voronoi models have better performance on the sector flight time, although the Voronoi model is better than the Bisection model in terms of workload balancing. The KD-tree has some solutions with lower workload standard deviation but all solutions have a shorter average sector flight time, because the cuboid sectors are unable to tune the sectors' shapes according to the traffic flows; this resulted in the lower average sector flight time.

On the other hand, neither iABM nor KD-tree model can keep sector boundaries far away from the traffic flow crossing points. The minimum distances between crossing points and sector boundaries of their solutions are all below 8 NM. The grid based approach adopted by iABM generates the fine boundaries of sectors limiting the capability to keep boundaries away from the traffic flow crossing points. The cuboid sectors produced by KD-tree model have less flexibility to tune the

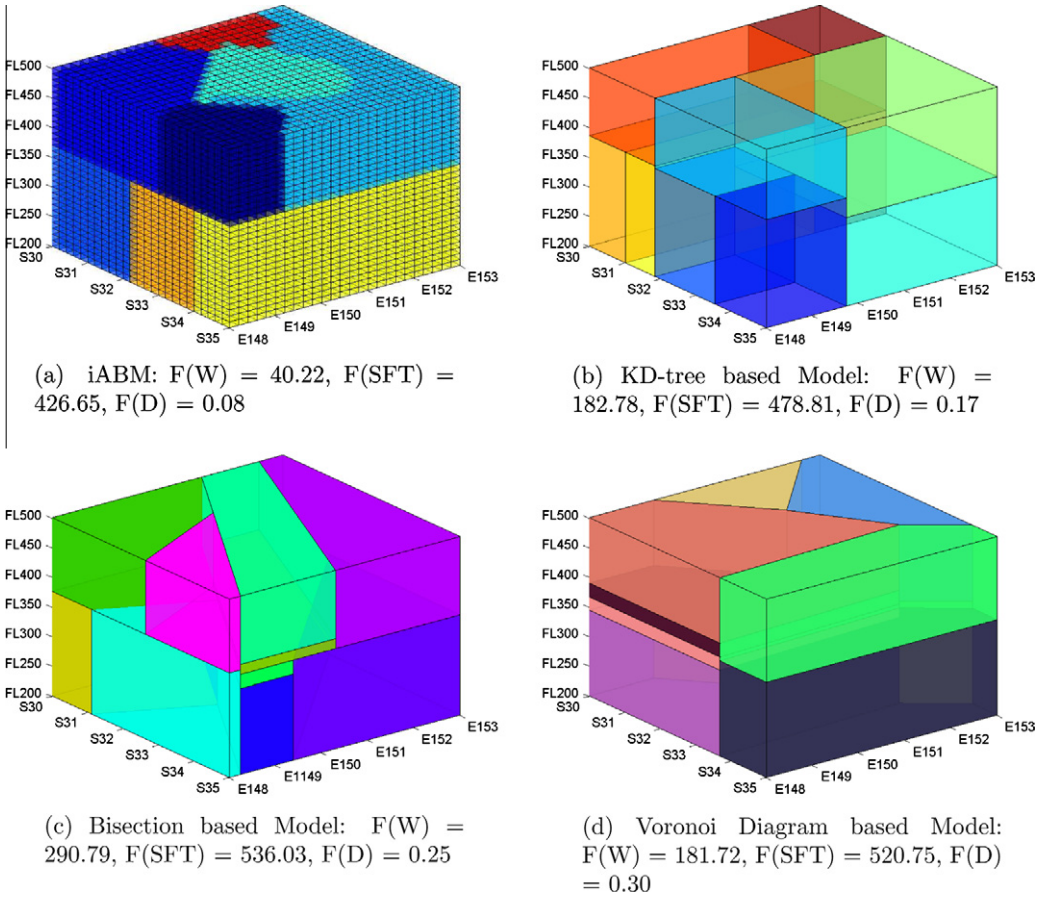


Fig. 17. Examples of airspace sectors (Minimum Standard Deviation of workload) generated by 4 airspace sectorization models.

boundaries in order to satisfy all three objectives at the same time. Both the Bisection and Voronoi models are able to produce solutions where the distances between sector boundaries and traffic flow crossing points are more than 10 NM, although the Voronoi model has the largest minimum distance (13.50 NM) between the sector boundaries and traffic flow crossing points among all proposed models.

The lowest workload standard deviation (40.22) was achieved by iABM. There are two reasons for this. The first is that iABM only allows agents to group the cells within a predefined workload range, unless the Gap Filling Rule takes actions. The second is that agents are working on a gridded airspace, and small grids give the agent more flexibility to partition airspace. However, the average sector flight time of this model is lower than Bisection and Voronoi Diagram based models because the Growth Rule prevents the agent from grouping more traffic flows when the workload limits are exceeded.

Table 7 lists the percentage of acceptable solutions with each model. iABM has the highest percentage of acceptable solutions, because the Growth Rule limits the workload variance; the other models have no specific guidance to limit the workload variance when sectorizing airspace, and so produce fewer acceptable solutions than iABM. Table 7 also shows the objective values of the best-performed examples, in terms of minimum workload variance, maximum average sector flight time, or maximum smallest distance between sector boundaries and traffic flow crossing points, generated by each model. iABM has the best solution for workload balancing, the Voronoi model has the longest sector flight time and the longest minimum distance between sector boundaries and traffic flow crossing points. The Bisection model has better results on the maximum average sector flight time and larger minimum distance between sector boundaries and traffic flow crossing points than both iABM and the KD-tree model, however the minimum workload standard deviation achieved by it is worse than them.

Fig. 19 shows the distributions of the acceptable solutions with each model. The left column shows the distributions according to workload balancing, the middle column shows distributions according to the average sector flight time, and the right column shows the distributions according to the minimum distance between sector boundaries and traffic flow crossing points.

Higher bars in the figures mean more solutions falling in the nominated objective range. Again it is seen that iABM performs best on workload balancing (larger percentage of solutions with lower workload variance) among the models. Both

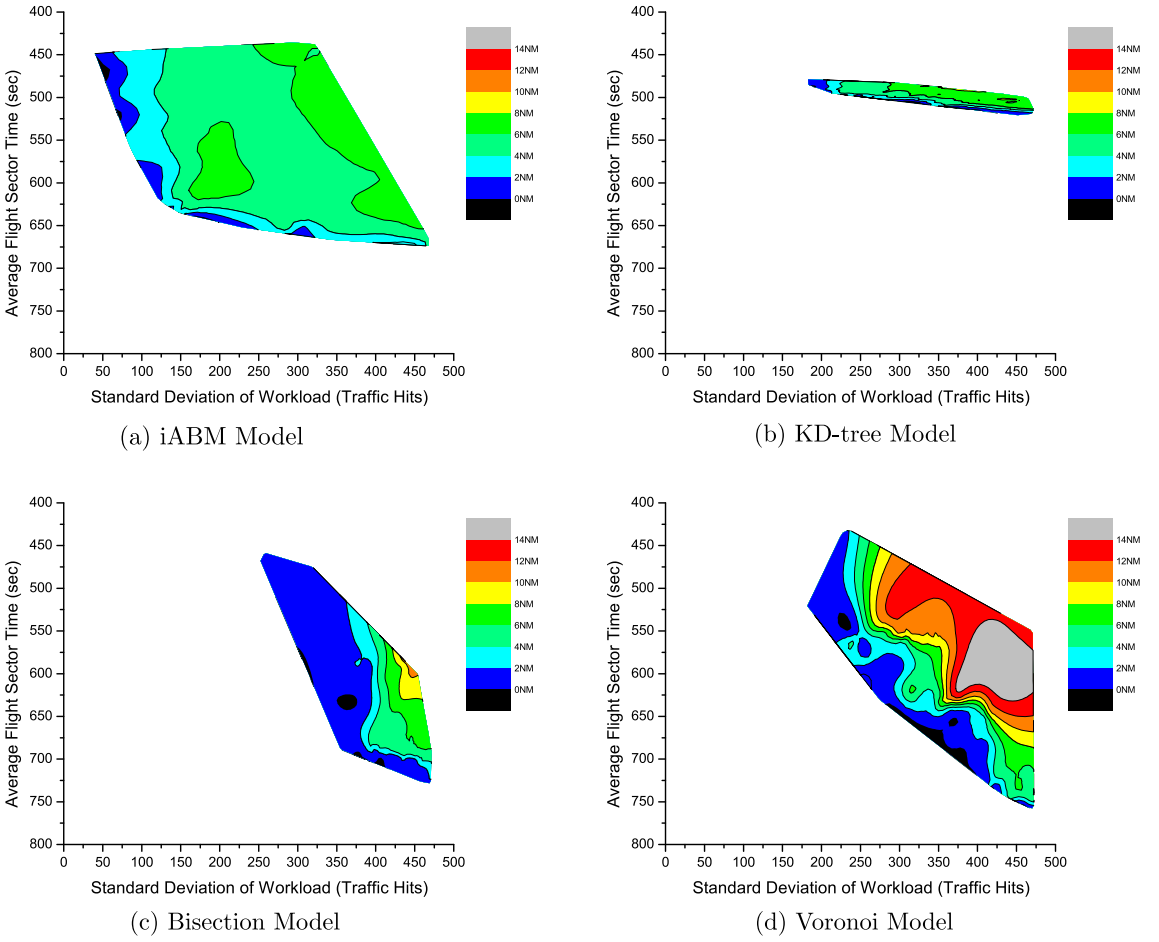


Fig. 18. Comparison of Pareto Fronts of feasible solutions by 4 3D airspace partitioning methods.

Table 7
Percentages of acceptable solutions based on workload standard deviation of 4 airspace sectorization models, and their best solutions for three objectives.

Models		iABM	KD-Tree	Bisection	Voronoi
% of acceptable solutions		**25.00**	1.21	3.16	1.29
Example of min ($F(W)$)	**F(W)**	**40.22**	182.78	290.79	181.72
	$F(SFT)$	448.65	478.81	536.03	520.75
	$F(D)$	0.08	0.17	0.25	0.30
Example of max ($F(SFT)$)	$F(W)$	469.99	450.97	469.10	472.27
	F(SFT)	673.98	520.73	728.34	**757.81**
	$F(D)$	4.95	0.94	0.78	0.07
Example of max ($F(D)$)	$F(W)$	386.70	414.23	454.14	453.12
	$F(SFT)$	557.53	494.73	600.66	544.55
	F(D)	7.35	7.96	10.02	**13.50**

Bisection and Voronoi Diagram based models have better performance when maximizing average sector flight time. They also both have some solutions falling into portion of the larger minimum distance between sector boundaries and crossing points compared to the other models. The KD-tree based model performs poorly on all objectives.

Table 8 shows the performance of the examples achieving the best workload balance in each sector, with each model. The minimum distance between sector boundaries and traffic flow crossing points does not exist for some sectors when there are no crossing points inside the sectors, therefore the value of the minimum distance is marked as N/A in the table. The workload variances of the sectors produced by iABM are smaller than with other models, and therefore the standard deviation of the workload with this model is the lowest. The other models are capable of achieving acceptable workload balance, as

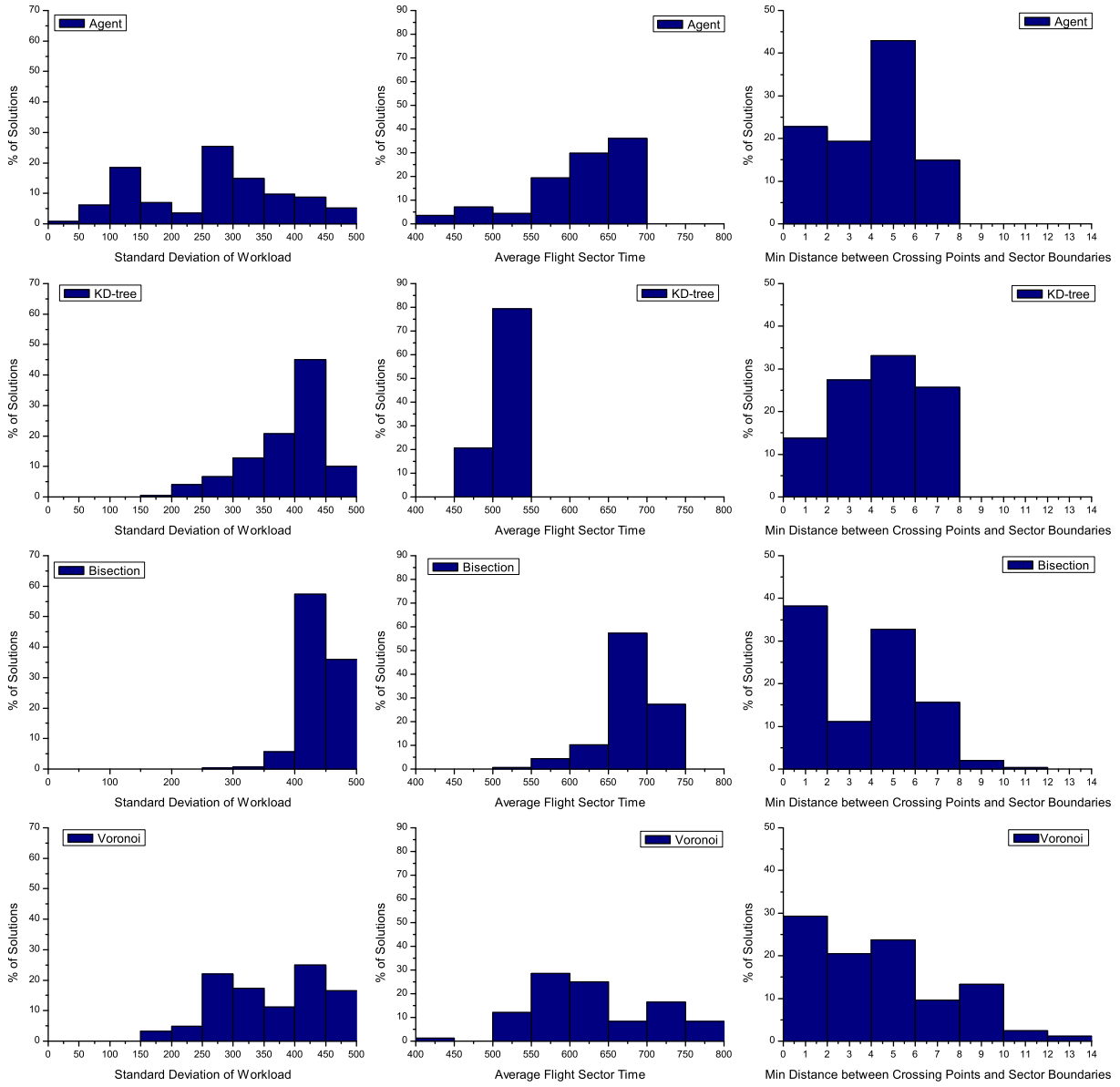


Fig. 19. Distributions of all acceptable solutions based on workload standard deviation, average sector flight time, and the minimum distance between crossing points and sector boundaries for 4 airspace sectorization models.

shown in the examples, but the variation is greater. However, the average sector flight time and the minimum distance between sector boundaries and traffic flow crossing points are not that high for all models.

Conversely, Table 9 shows the performance of the examples achieving the best average sector flight time, among the acceptable solutions produced with each model. All models can achieve longer flight times, but with higher workload variance and smaller minimum between sector boundaries and traffic flow crossing points.

Finally, the examples with the largest minimum distance between sector boundaries and traffic flow crossing points, among the acceptable solutions are listed in the Table 10. The largest minimum distance (13.5 nm) is achieved by the Voronoi Diagram based model. iABM and KD-tree models cannot generate sectorizations which have the minimum distance larger than 8 nm.

The longer sector flight time is achieved by alignment between sectors and traffic flows and minimizing the traffic flow cuts by sectors. Fig. 20 shows an example of some sectors produced by the Voronoi Diagram based model (workload standard deviation = 472.27 and average sector flight time = 757.81, and Minimum Distance between Sector Boundaries and Traffic flow Crossing Points = 0.07). It demonstrates that sectors encapsulate and align with the major air traffic flows in both the low and high airspace. Only one major air traffic flow is segmented by sectors in order to balance the workload. Therefore, the number of traffic flow cuts is minimized.

Table 8
Workload variance ($Var.W = W_k - W_{avg}$ and $W_{avg} = 2361.9$), average sector flight time (SFT_k) and minimum distance between sector boundaries and traffic flow crossing points (D_k) of each sector of the examples with best workload balance (bold values) produced by four airspace sectorization models.

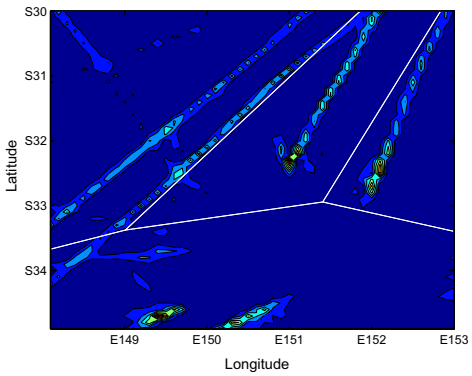
Sectors	3D DAS models					
	iABM			KD-tree		
	Var. W	SFT_k	D_k	*Var. W*	SFT_k	D_k
S_1	−53.90	238.76	3.36	−205.90	333.40	7.82
S_2	−50.90	592.56	11.33	57.10	549.77	1.43
S_3	−39.90	627.57	38.79	195.10	525.41	48.78
S_4	−13.90	437.52	1.83	−26.90	188.31	31.12
S_5	−5.90	234.04	7.42	296.10	275.92	N/A
S_6	8.10	430.91	31.57	43.10	385.83	0.17
S_7	12.10	494.58	3.26	28.10	539.10	23.43
S_8	17.10	499.09	1.89	−284.90	759.88	9.22
S_9	52.10	332.20	8.42	−241.90	594.39	10.51
S_{10}	75.10	599.26	0.08	140.10	636.10	27.49
Objectives	**40.22**	448.65	0.08	**182.78**	478.81	0.17
	F(W)	**F(SFT)**	**F(D)**	**F(W)**	**F(SFT)**	**F(D)**
	Bisection			Voronoi		
S_1	163.10	224.78	1.12	−9.90	235.99	6.77
S_2	300.10	273.49	3.65	−117.90	288.93	1.92
S_3	−290.90	267.80	30.74	−249.90	581.28	0.30
S_4	−311.90	272.12	0.65	275.10	627.86	0.73
S_5	33.10	565.75	0.78	−157.90	418.48	16.26
S_6	−236.90	827.92	N/A	240.10	804.74	N/A
S_7	−270.90	825.39	0.25	74.10	395.03	0.92
S_8	557.10	557.77	0.47	−34.90	450.39	0.48
S_9	257.10	618.66	0.88	−219.90	531.07	1.37
S_{10}	−199.90	926.57	10.89	201.10	873.75	0.56
Objectives	**290.79**	536.03	0.25	**181.72**	520.75	0.30
	F(W)	**F(SFT)**	**F(D)**	**F(W)**	**F(SFT)**	**F(D)**

Table 9
Workload variance ($Var.W = W_k - W_{avg}$ and $W_{avg} = 2361.9$), average sector flight time (SFT_k) and minimum distance between sector boundaries and traffic flow crossing points (D_k) of each sector of the examples with longest average sector flight time (bold values) produced by 4 airspace sectorization models.

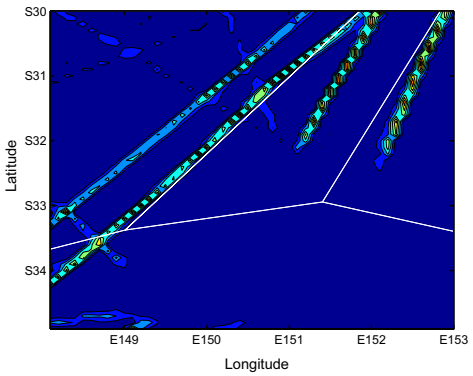
Sectors	3D DAS models					
	iABM			KD-tree		
	Var. W	SFT_k	D_k	*Var. W*	SFT_k	D_k
S_1	−631.90	320.37	31.89	25.10	356.27	8.81
S_2	−522.90	227.98	7.42	892.10	682.66	0.94
S_3	−281.90	1006.45	6.00	−136.90	606.82	75.64
S_4	−179.90	770.12	10.80	299.10	220.52	35.45
S_5	−127.90	736.48	18.61	−343.90	254.37	N/A
S_6	−93.90	756.00	13.16	−515.90	407.21	4.81
S_7	36.10	1057.94	4.95	−180.90	491.95	22.28
S_8	370.10	1037.47	N/A	−637.90	783.64	4.88
S_9	373.10	416.50	6.65	49.10	669.72	23.78
S_{10}	1059.10	410.52	6.74	550.10	734.12	40.75
Objectives	469.99	**673.98**	4.95	450.97	**520.73**	0.94
	F(W)	**F(SFT)**	**F(D)**	**F(W)**	**F(SFT)**	**F(D)**
	Bisection			Voronoi		
S_1	223.10	994.23	9.29	452.10	456.32	1.04
S_2	−447.90	941.31	13.62	163.10	714.62	0.11
S_3	−342.90	484.56	4.18	310.10	229.68	3.78
S_4	162.10	1009.60	13.65	355.10	335.43	0.24
S_5	−626.90	1269.51	14.46	−401.90	753.85	N/A
S_6	−412.90	512.89	2.77	−1044.90	1067.84	N/A
S_7	734.10	488.84	12.06	−491.90	935.00	0.28
S_8	518.10	230.40	1.74	33.10	970.95	0.64
S_9	−359.90	984.59	N/A	106.10	1073.04	0.07
S_{10}	553.10	367.44	0.78	519.10	1041.33	0.32
Objectives	469.10	**728.34**	0.78	472.27	**757.81**	0.07
	F(W)	**F(SFT)**	**F(D)**	**F(W)**	**F(SFT)**	**F(D)**

Table 10
Workload variance ($Var.W = W_k - W_{avg}$ and $W_{avg} = 2361.9$), average sector flight time (SFT_k) and minimum distance between sector boundaries and traffic flow crossing points (D_k) of each sector of the examples with largest minimum distance between sector boundaries and traffic flow crossing points (bold values) produced by 4 airspace sectorization models.

Sectors	3D DAS models					
	iABM			KD-tree		
	Var. W	SFT_k	D_k	*Var. W*	SFT_k	D_k
S_1	−605.90	675.38	16.40	−332.90	320.37	7.96
S_2	−402.90	489.75	15.55	−402.90	511.04	7.96
S_3	−371.90	320.97	18.25	325.10	541.01	49.36
S_4	−185.90	615.85	9.82	−332.90	166.77	14.49
S_5	−48.90	913.03	300.00	−520.90	210.00	134.12
S_6	66.10	543.58	18.26	497.10	422.51	8.85
S_7	230.10	283.80	13.20	−22.90	527.59	25.65
S_8	256.10	347.52	7.35	−238.90	786.30	25.85
S_9	305.10	571.50	12.26	269.10	674.62	23.20
S_{10}	758.10	813.91	11.11	760.10	787.06	40.18
Objectives	386.70	557.53	**7.35**	414.23	494.73	**7.96**
	F(W)	**F(SFT)**	**F(D)**	**F(W)**	**F(SFT)**	**F(D)**
	Bisection			Voronoi		
S_1	−449.90	807.89	10.02	−294.90	543.95	62.72
S_2	−260.90	851.76	12.97	268.10	830.53	17.45
S_3	117.10	336.52	11.08	−495.90	227.56	19.49
S_4	−377.90	901.82	16.70	−441.90	411.43	N/A
S_5	−72.90	995.22	10.08	−130.90	527.01	22.83
S_6	610.10	422.56	28.99	256.10	561.00	56.02
S_7	883.10	479.56	22.47	541.10	268.80	13.50
S_8	−654.90	226.59	65.28	933.10	349.29	62.80
S_9	−14.90	442.83	N/A	−264.90	806.54	23.30
S_{10}	221.10	541.89	36.95	−369.90	919.38	28.52
Objectives	454.14	600.66	**10.02**	453.12	544.55	**13.50**
	F(W)	**F(SFT)**	**F(D)**	**F(W)**	**F(SFT)**	**F(D)**



(a) Low Level Sectors and Major Traffic Flows (FL200-FL370)

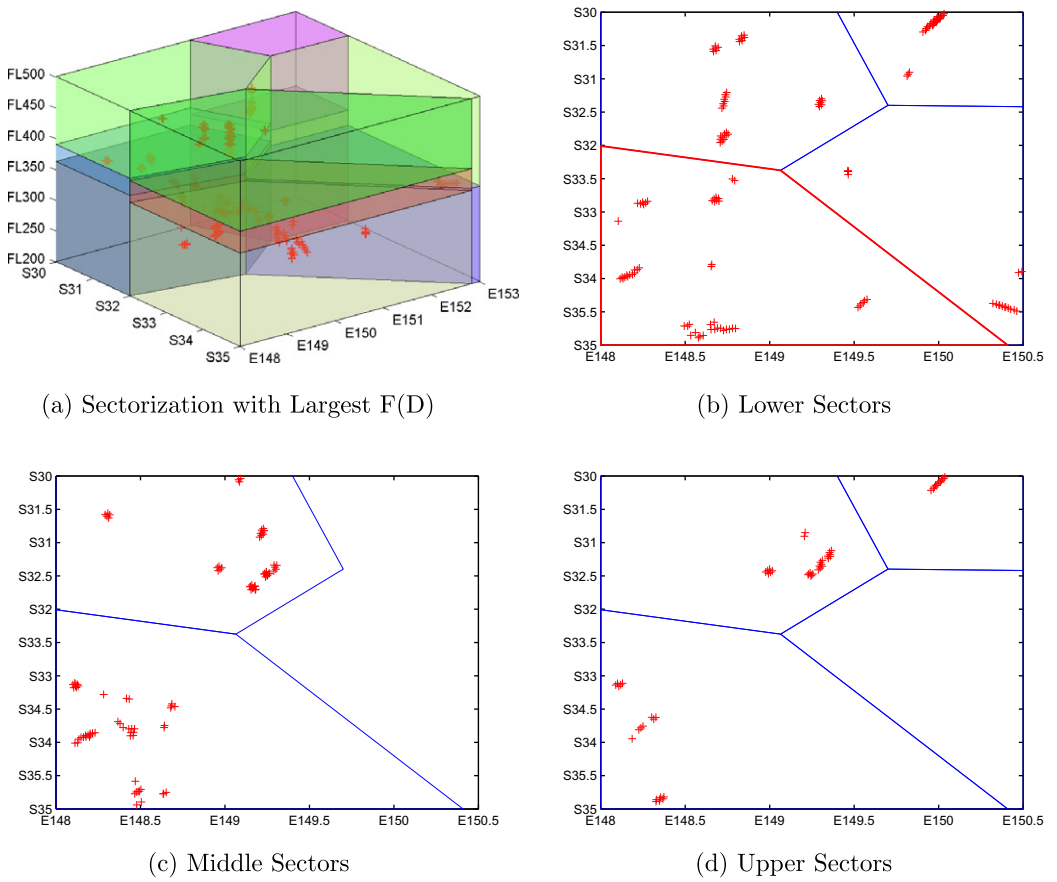


(b) High Level Sectors and Major Traffic Flows (FL370-FL500)

Fig. 20. Alignment between sectors and traffic flows: an example with the longest average sector flight time among the acceptable solutions produced by Voronoi Diagram based model.

Fig. 21 shows an example with the largest minimum distance between sector boundaries and traffic flow crossing points generated by the Voronoi Diagram based model. In this case, the minimum distance is 13.50 nm among all sectors, which happens in a lower sector as highlighted in Fig. 21b. The workload standard deviation of the shown example is 453.12. However, its average sector flight time (544.55) is not as high as the previous example. For this example, the traffic flows are segmented by sectors in order to keep their boundaries away from the crossing points.

In summary, iABM performs best on workload balancing, but cannot satisfy the convexity constraint on sector shape. Both Bisection and Voronoi Diagram based models have better performance on average sector flight time, but the Voronoi Diagram based model has better performance on maximizing the minimum distance between sector boundaries and traffic flow crossing points. Although they cannot achieve the lower workload variance of iABM, both can achieve acceptable workload



(a) Sectorization with Largest F(D)

(b) Lower Sectors

(c) Middle Sectors

(d) Upper Sectors

Fig. 21. Distance between sector boundaries and traffic flow crossing points: an example with the largest minimum distance between sector boundaries and traffic flow crossing points ($F(D) = 13.50$ nm) produced by Voronoi Diagram based model.

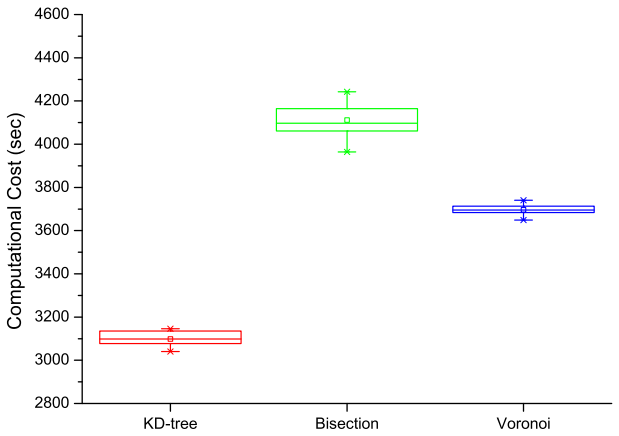


Fig. 22. Box chart of computational costs incurred by the KD-tree, Bisection, and Voronoi Diagram based models. The figure depicts the averages, the medians, the interquartile range (between 25th and 75th percentiles), the 5th and 95th percentiles, and the minimum and maximum costs of 10 runs for each model.

balance. The KD-tree based model performed poorly, because of its geometry limitation which also limits its value in a real operational environment. The results also show that the three objectives adopted in this 3D DAS optimization conflict with each other. The trade-off between the three objectives needs to be considered by decision makers depending on the operational conditions.

9.2. Computation cost comparison

DAS is intended to be used in a real time operational environment, to adjust the airspace sectorization quickly in order to accommodate fluctuating air traffic demand. An efficient DAS method is necessary to satisfy this requirement. In this section, we compare the computation cost of the four proposed models based on the execution time.

The computation cost of the optimizations was measured 10 times with all four models, and the average cost as well as the minimum and maximum cost were calculated for each model. The average time consumed by iABM is 209,321 s, which is the highest among all models. The KD-tree based model had the lowest average computation cost of 3097.40 s. The Bisection based model (average computation cost of 4096.78 s) took around 10% longer time than the Voronoi Diagram based model (average computation cost is 3696.60 s) because it needs to find the two points closest to the middle of a space when bisecting the space, which requires some extra computation cost. The computation cost of iABM is not at the same magnitude as other models (almost 50 times of the Bisection model) because of the complex agent rules associated with the model. Therefore, the computation costs of the proposed models excluding the iABM are shown in Fig. 22 as box charts. The difference between the minimum and maximum computation cost of Bisection model is also larger than the other two models.

10. Discussion

In this paper we first evaluated the agent based approach for 3D airspace sectorization. Although ABM can produce a 3D sectorization solution with the consideration of traffic commonality and balancing sectors workload, experiments show that this approach suffers from right prism violation due to limitations of the Repair Rule and cell level based Movement Rule. We then proposed the iABM approach, where new rules are proposed for agent growth and gap filling. Another limitation of ABM was that some sector nests inside sectors, which was also addressed by an improved Agent geometric plane-based Growth Rule.

The high computation cost of the Repair Rule and Trading Rule, caused by sector contiguity check, was handled in iABM by removing the contiguity check of sectors without violation. This leads to significant computation cost savings compared to the ABM model. It was also found that both iABM and ABM are grid based approaches, thus the sector shapes are all cuboid when the convexity constraint is applied. Further, an additional agent rule on convexity check will increase the computation cost significantly.

Therefore, a geometric space partition method based on KD-tree was used to addresses the convex shape constraint for each sub space. However the polytope is limited to cuboid, which makes the KD-tree based model less effective to align sectors with traffic flows and to maintain the minimum distance between sector boundaries and traffic flow crossing points.

All cuboid sectors may not be practical in reality, and hence to address the convexity constraint we further developed 3D sectorization models based on Bisection and Voronoi Diagrams. Models based on Bisection and Voronoi Diagrams generate convex shapes although their computation cost is more than the KD-tree based model because of geometric calculations. However, they are still faster than agent based models. Furthermore, Bisection and Voronoi based models can introduce more polytopes for the sector shapes to capture air traffic flows, which is a big advantage over KD-tree based model.

We then applied multi-objective optimization using NSGA-II to optimize the airspace sectorization in terms of ATC workload balancing, the maximum average sector flight time, and the minimum distance between sector boundaries and traffic flow crossing points; and used the Pareto concept of optimality for performance analysis. iABM has the best performance on workload balancing, but it cannot satisfy the convexity constraint on sector shape. Both Bisection and Voronoi Diagram based models have better performance on the average sector flight time. The Voronoi Diagram based model is better on the minimum distance between sector boundaries and traffic flow crossing points than the Bisection model. Although they cannot achieve the as low a workload variance as iABM, both of them achieved expected workload balance. KD-tree based model shows poor performance because of its geometric limitation, which also limits its use in an operational environment.

This paper shows that there might not be a silver bullet for the Dynamic Airspace Sectorization problem. Different approaches each have advantages and limitations. What is required is to adapt them to specific needs based on airspace constraints and user requirements. In future work we will be extending our work by conducting sensitivity analysis with different traffic distribution, different airspace constraints (weather, special use airspace, etc.) and different airspace requirements (en-route, transition, etc.) as well as conducting a similarity analysis on the sector shape changed by DAS.

Acknowledgements

The authors wish to acknowledge the useful discussions with AirServices Australia and EUROCONTROL CND, Brétigny.

Appendix A. Algorithms

Algorithm 1. Agent Growth Rule of iABM

$C''_{uvz} \Leftarrow BC_k$ of S_k
$P_T \Leftarrow C''_{u,v,T}$ where $T \Leftarrow \max(z) + 1$ and $C''_{u,v,T}$ is unassigned {Top horizontal layer}
$P_B \Leftarrow C''_{u,v,B}$ where $B \Leftarrow \min(z) - 1$ and $C''_{u,v,B}$ is unassigned {Bottom horizontal layer}
{Vertical blocks}
$P_{v1} \Leftarrow \{C''_{u\pm1,v,z}\}$ where $C''_{u\pm1,v,z}$ is unassigned
$P_{v2} \Leftarrow \{C''_{u,v\pm1,z}\}$ where $C''_{u,v\pm1,z}$ is unassigned
$P_V \Leftarrow P_{v1} \cup (P_{v2} \backslash (P_{v1} \cap P_{v2}))$
$P'_V \Leftarrow P_V$
for each P_{Vi} **in** P'_V **do**
  if P_{Vi} *violates Right Prism* **then**
    $P_V \Leftarrow P_V \backslash P_{Vi}$
  end if
end for
if *workload*$(P_T) \equiv 0$ **and** *workload*$(P_B) \equiv 0$ **and** *workload*$(P_{Vi}) = 0$ for each P_{Vi} **then**
  $S_k \Leftarrow S_k \bigcup P_T \bigcup P_B \bigcup P_V$
else
  $P' \Leftarrow (P_T \cup P_B \cup P_V)$
  for each p in $(P_T \cup P_B \cup P_V)$ **do**
    if *workload*$(p) >$ *MAX WORKLOAD* **then**
      $P' \Leftarrow P' \backslash p$
    end if
  end for
  Select *OP* if *Commonality*$(OP) \equiv \max($*Commonality*$(P'))$
  $S_k \Leftarrow S_k \bigcup OP$
end if
Update workload of S_k

Algorithm 2. Agent Gap Filling Rule of iABM

$UC \Leftarrow$ all unassigned cells in R
while $|UC| > 0$ **do**
  Create two empty list of agents *SU* and *SL*
  for each S_k **in** S **do**
    if $C_{u,v,z} \in UC$ **and** $C_{u,v,z+1} \in S_k$ **then**
      put S_k in *SU*
    end if
    if $C_{u,v,z} \in UC$ **and** $C_{u,v,z-1} \in S_k$ **then**
      put S_k in *SL*
    end if
  end for
  if $(|SU| < |SL|)$ **then**
    $AS \Leftarrow SU$
  else
    $AS \Leftarrow SL$
  end if
  for each S_k **in** *AS* **do**
    $C''_{uvz} \Leftarrow BC_k$ of S_k
    if $L \equiv SU$ **then**
      $l = min(z) - 1$
      $l' = min(z)$
    else

(*continued on next page*)

```
23:            l = max(z) + 1
24:            l′ = max(z)
25:          end if
26:          L ⇐ C″_{u,v,l}
27:          for each agent S_j in (S\S_k) do
28:            if OC ∈ L and OC ∉ S_k and OC ∈ S_j then
29:              OC″_{u′v′z′} ⇐ BC_j of S_j
30:              L′ ⇐ OC″_{u′,v′,l′}
31:              S_j = S_j\L′
32:              update workload of S_j
33:            end if
34:          end for
35:          S_k = S_k ⋃ L
36:          update workload of S_k
37:        end for
38:        UC⇐ all unassigned cells in R
39:      end while
```

Algorithm 3. 3D Airspace Sectorization by KD-tree based model

```
1:   {Input: a list of points (P{x_1,x_2,...,x_p}), a given space (R), a dimension value (K), and a flag (D and
     initially D ⇐ 1) to determine partitioning axis}
2:   Function KDTreeAirspace (P, R, D) {
3:   if |P| ≡ 0 then
4:     RETURN
5:   else
6:     axis ⇐ D mod K
7:     Sort P by the value of axis
8:     Select the median x_m from P
9:     Divide R by axis value of x_m into S_1, S_2 where R ≡ S_1 ∪ S_2 and S_1 ∩ S_2 ≡ ∅
10:    P′ ⇐ P\x_m
11:    Create two empty list P_1 and P_2
12:    for each x_i in P′ do
13:      if x_i ∈ S_1 then
14:        P_1 ⇐ P_1 ⋃ x_i
15:      else
16:        P_2 ⇐ P_2 ⋃ x_i
17:      end if
18:    end for
19:    d ⇐ D + 1
20:    KDTreeAirspace(P_1, S_1, d)
21:    KDTreeAirspace(P_2, S_2, d)
22:  end if
23:  }
```

Algorithm 4. 3D Airspace Sectorization by Bisection based model

```
1:   {Input: a list of points (P{x_1,x_2,...,x_p}), a given space (R), and a flag (IsLateral) to determine
     partitioning directions}
2:   Function BisectAirspace (P, R, IsLateral) {
3:   if |P| ≡ 0 then
4:     RETURN
5:   end if
6:   if IsLateral ≡ FALSE or |P| ≡ 1 then
```

```
Sort P by the altitude value
Get the middle point x_m from P
Divide R vertically by altitude value of x_m into S_1, S_2 where R ≡ S_1 ∪ S_2 and S_1 ∩ S_2 ≡ ∅
P' ⇐ P\x_m
Create two empty list P_1 and P_2
for each x_i in P' do
  if x_i ∈ S_1 then
    P_1 ⇐ P_1 ⋃ x_i
  else
    P_2 ⇐ P_2 ⋃ x_i
  end if
end for
BisectAirspace(P_1, S_1, TRUE)
BisectAirspace(P_2, S_2, TRUE)
else
  Get two points x_j, x_k that are closest to the centroid of all points in P
  Bisect the R laterally into two sub-spaceR_1, R_2 by the latitude and longitude of x_j, x_k where
    R ≡ S_1 ∪ S_2 and S_1 ∩ S_2 ≡ ∅
  P' ⇐ (P\x_j)\x_k
  Create two empty list P_1 and P_2
  for each x_i in P' do
    if x_i ∈ S_1 then
      P_1 ⇐ P_1 ⋃ x_i
    else
      P_2 ⇐ P_2 ⋃ x_i
    end if
  end for
  BisectAirspace(P_1, S_1, FALSE)
  BisectAirspace(P_2, S_1, FALSE)
end if
}
```

Algorithm 5. 3D Airspace Sectorization by Voronoi Diagram based model

```
{Input: a list of points (P{x_1,x_2,...,x_p}), a given space (R), and N is the number of Voronoi Sites}
Function VoronoiDiagramAirspace(P, R) {
if |P| ≡ 0 then
  RETURN
end if
if |P| ≡ 1 then
  Divide R vertically by altitude value of x_m in P into S_1, S_2 where R ≡ S_1 ∪ S_2 and S_1 ∩ S_2 ≡ ∅
  RETURN
else
  if (N > |P|) then
    n ⇐ |P|
  else
    n ⇐ N
  end if
  Get n points from P as Voronoi sites site_i
  Discompose R laterally into Voronoi Cell V by the latitude and longitude of all site_i where
  R ≡ ∪_{i=1}^{n} V_i and V_i ∩ V_j ≡ ∅ (i ≠ j)
  for all V_i ∈ V do
    Divide the each Voronoi Cell V_i into two sup-spaces (S_i1,S_i2) by the altitude of its site site_i
    where V_i ≡ S_i1 ∪ S_i2 and S_i1 ∩ S_i2 ≡ ∅
    S ⇐ S ⋃ S_i1 ⋃ S_i2
  end for
```

(*continued on next page*)

```
21:        P' ⇐ P\sites
22:        Create (n × 2) empty lists PL
23:        for each x_k in P' do
24:          for each s_ij in S do
25:            if x_k ∈ s_ij do
26:              PL_ij ⇐ PL_ij ⋃ x_k
27:            end if
28:          end for
27:        end for
30:        for each S_ij in S do
31:          VoronoiDiagramAirspace(P_ij, S_ij)
32:        end for
33:      end if
34:      }
```

Appendix B. Acronyms

2D	2-Dimensional
3D	3-Dimensional
4D	4-Dimensional
ABM	Agent Based Model
ATC	Air Traffic Controller
ATOMS	Air Traffic Operations and Management Simulator
DAC	Dynamic Airspace Configuration
DAS	Dynamic Airspace Sectorization
GA	Genetic Algorithm
KD-tree	k-dimensional tree
iABM	improved Agent Based Model
MIP	Mixed Integer Programming
NSGA-II	Non-dominated Sorting Genetic Algorithm-II
SBX	Simulated Binary Crossover

References

Alam, S., Abbass, H., Barlow, M., 2008. Air traffic operations and management simulator ATOMS. IEEE Transactions on Intelligent Transportation System 9 (2), 209–225.

Ball, M., Hoffman, R., Chen, C., Vossen, T., 2001. Collaborative decision making in air traffic management: current and future research directions. New Concepts and Methods in Air Traffic Management, 17–30.

Basu, A., Mitchell, J.S.B., Sabhnani, G.K., 2009. Geometric algorithms for optimal airspace design and air traffic controller workload balancing. Journal of Experimental Algorithmics 14, 2.3–2.28.

Bentley, J., 1980. Multidimensional divide-and-conquer. Communications of the ACM 23 (4), 214–229.

Brinton, C., Pledgie, S., 2008. Airspace partitioning using flight clustering and computational geometry. In: IEEE/AIAA 27th Digital Avionics Systems Conference, 2008 (DASC 2008), p. 3.

Deb, K., Goyal, M., 1996. A combined genetic adaptive search (geneas) for engineering design. Computer Science and Informatics 26, 30–45.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

Deb, K., Sindhya, K., Okabe, T., 2007. Self-adaptive simulated binary crossover for real-parameter optimization. In: GECCO '07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. ACM, New York, NY, USA, pp. 1187–1194.

Delahaye, D., Puechmorel, S., 2008. 3D airspace design by evolutionary computation. In: IEEE/AIAA 27th Digital Avionics Systems Conference, 2008 (DASC 2008). IEEE, p. 3.

Delahaye, D., Schoenauer, M., Alliot, J., 2001. Airspace sectoring by evolutionary computation. Spatial evolutionary modeling, p. 203.

FAA Advisory Circular 90–99, August 2003. High altitude airspace design phase i. Tech. Rep. FAA Advisory Circular 90–99, ATA-301, FAA.

FAA Order 7210.3U, February 2006. Facility operations and administration. Tech. Rep. FAA Order 7210.3U, FAA.

FAA Order 7400.2F, February 2006. Procedure for handling airspace matters. Tech. Rep. FAA Order 7400.2F, FAA.

Fortune, S., 1987. A sweepline algorithm for Voronoi diagrams. Algorithmica 2 (1), 153–174.

Homola, J., Lee, P., Prevôt, T., Lee, H., Kessell, A., Brasil, C., Smith, N., 2010. A Human-in-the loop exploration of the dynamic airspace configuration concept. In: AIAA Guidance, Navigation, and Control (GNC) Conference and Exhibit.

ICAO, 2010. Environmental report 2010. Tech. Rep., International Civil Aviation Organization.

Kicinger, R., Yousefi, A., September 2009. Heuristic method for 3d airspace partitioning: Genetic algorithm and agent-based approach. In: 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO) and Aircraft Noise and Emissions Reduction Symposium (ANERS).

Kopardekar, P., Bilimoria, K., Sridhar, B., September 2007. Initial concepts for dynamic airspace configuration. In: 7th AIAA Aviation Technology, Integration and Operations Conference (ATIO), Belfast, Northern Ireland.

Lee, P., Mercer, J., Gore, B., Smith, N., Lee, K., Hoffman, R., Aug 2008. Examining airspace structural components and configuration practices for dynamic airspace configuration. In: Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit, Honolulu, HI, pp. 18–21.

Mitchell, J.S.B., Sabhnani, G., Krozel, J.A., Hoffman, R.L., Yousefi, A., August 2008. Dynamic airspace configuration management based on computational geometry techniques. In: AIAA Guidance, Navigation and Control Conference and Exhibit. No. AIAA 2008-7225, Honolulu, HI.

Sherali, H.D., Hill, J.M., 2009. Reverse time-restricted shortest paths: application to air traffic management. Transportation Research Part C: Emerging Technologies 17 (6), 631–641.

Songchen, H., Zhang, M., 2004. The optimization method of the sector partition based on metamorphic Voronoi polygon. Chinese Journal of Aeronautics 17 (1), 7–12.

Trandac, H., Baptiste, P., Duong, V., 2003. Airspace sectorization by constraint programming. In: RIVF, pp. 49–58.

Verlhac, C., Manchon, S., 2001. Optimization of opening schemes. In: Proceedings of the fourth USA/Europe Air Traffic Management R&D Seminar.

Weigang, L., Dib, M.V.P., Alves, D.P., Crespo, A.M.F., 2010. Intelligent computing methods in air traffic flow management. Transportation Research Part C: Emerging Technologies 18 (5), 781–793. Applications of Advanced Technologies in Transportation: Selected papers from the 10th AATT Conference.

Xue, M., 2008. Airspace sector redesign based on Voronoi diagrams. In: Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit, pp. 18–21.

Zelinski, S., July 2009. A comparison of algorithm generated sectorizations. In: Proceedings of Eighth USA/Europe Air Traffic Management Research and Development Seminar, Napa, CA.