JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 4, July–August 2010

Graph-Based Algorithm for Dynamic Airspace Configuration

Jinhua Li,[*] Tong Wang,[†] Mehernaz Savai,[‡] and Inseok Hwang[§]
Purdue University, West Lafayette, Indiana 47907

DOI: 10.2514/1.47720

In this paper, a new algorithm for dynamic airspace configuration is developed based on graph theory. A graph model is first constructed that accurately represents the air-route structure and air traffic in the National Airspace System. The airspace configuration problem is then formulated as a graph-partitioning problem to balance the subgraph (sector) workload while satisfying the capacity constraint, which is efficiently solved by a spectral clustering method. Since the original spectral clustering method shows some undesirable properties, such as disconnected subgraphs and unbalanced partitions, an algorithm is proposed to refine the partitions. Lastly, using the partitioned graph as an input, a novel airspace sectorization algorithm is developed, based on the graph search method. The new sectors computed by the sectorization algorithm have smooth boundaries and good geometrical shapes, which satisfy the minimum distance requirement (i.e., the sector boundaries are at least a minimum distance away from the airports, waypoints, and main air routes). The performance of the proposed dynamic airspace configuration algorithm is validated through various air-traffic scenarios with real air-traffic data.

Nomenclature

cut	=	cut cost of a graph with respect to a partition
D	=	graph degree matrix
deg	=	degree of a graph
$\mathbb{G}$	=	graph
H	=	graph indicator matrix
L	=	graph Laplacian matrix
$\mathcal{V}$	=	set of graph vertices
v_i	=	vertex i
W	=	graph weight matrix
WL	=	air-traffic-control sector (graph) total workload
ω	=	graph weight
$\omega_{i,j}$	=	weight of link (i, j)

I. Introduction

IN THE current National Airspace System (NAS), the airspace of the continental U.S. is divided into 20 air-route traffic-control centers (ARTCCs) (henceforth referred to as centers), each of which is then divided into several air-traffic-control (ATC) sectors (henceforth referred to as sectors), as shown in Fig. 1. An airspace sector is the fundamental operational unit for ATC. The air traffic within each sector is monitored by one or two air-traffic controllers to ensure safe operations of aircraft. Hence, it is necessary to design sectors carefully to ensure that the workload capacities of air-traffic controllers are not exceeded, at least in sustainable periods of time, while balancing the distribution of workload among a set of sectors.

The current NAS has a rigid structure, where the aircraft are required to follow certain fixed air routes, which makes the current static sectorization of the NAS effectively regulate air-traffic flow [1]. For future operations, such as those under the Next Generation Air Transportation System (NEXTGEN) [2], it has been advocated that aircraft could be given more flexibility of changing their flight routes (or flight plans) in response to changing conditions in order to maximize flight efficiency. With increased flight flexibility, the current static sectorization will no longer be able to efficiently manage the changing traffic [3]. Hence, dynamic configuration of the airspace sectors is needed to handle the air traffic effectively [4].

The main objective of the dynamic airspace configuration (DAC) is the efficient air-traffic capacity management, which is a counterpart of traffic flow management (TFM) for the traffic demand management. So far, most research effort has been focused on TFM (i.e., how to control air-traffic flows to achieve maximum throughput and minimum delays). In general, control inputs for TFM include ground holding, airborne holding, and rerouting. However, the air traffic is also strongly dependent on the configuration of sectors and centers (and controllers). Thus, TFM alone is not enough, especially since NEXTGEN plans to manage two to three times of the current amount of air traffic more efficiently. Therefore, a complementary approach has been proposed to manage the capacity of air traffic, which is mainly governed by the configuration of sectors and controllers. Thus, the objective of the DAC research is to develop a method that can generate a new sectorization that can best accommodate the changing air traffic. One benefit of the DAC is to dynamically and efficiently manage controllers. For example, according to the changing traffic, sectorization changes over time and controllers are reassigned to each sector accordingly. Some limited resectorization has been done today in some centers [e.g., Cleveland center (ZOB)] in which, during the low traffic time, some sectors are combined together and, during the high traffic time, some sectors are divided into multiple sectors, which is based on the experience of controllers (especially the supervisors). In addition, NEXTGEN envisions generic sectors for high-altitude airspace. Controllers of the generic sectors can freely switch to new sectors with minimum training required. For more information about the necessity and importance of dynamic sectors, refer to the paper by Kopardekar et al. and the references therein [4]. The objective of this paper is to develop a method that can perform airspace sectorization efficiently, which can be used as an advisory tool for controllers or an automated system.

There are quite a few research on DAC in recent years. Delahaye et al. solved the airspace configuration problem using evolutionary algorithms [5,6]. In their papers, the initial sectors were constructed arbitrarily and then optimized by genetic algorithms or evolution computation. However, the algorithms were tested on the generic network flow graphs only. Most recently, a similar work using a genetic algorithm was applied to the real air-traffic data by Xue [7], where the initial sectors were generated using the Voronoi diagram. The Voronoi diagram is a computational geometry method to decompose a space based on distances to a specified discrete set of objects in the space [8]. The airspace configuration problem was also formulated and solved using constrained mathematical programming by Trandac et al. [9]. Some initial experimentation on the

Presented as Paper 6252 at the AIAA Guidance, Navigation, and Control Conference, Chicago, IL, 10–13 August 2009; received 19 October 2009; revision received 16 February 2010; accepted for publication 28 February 2010.

[*]Graduate Student; li31@purdue.edu.
[†]Graduate Student, Beijing Institute of Technology, People's Republic of China; wang171@purdue.edu.
[‡]Graduate Student, Purdue University; msavai@purdue.edu.
[§]Assistant Professor; ihwang@purdue.edu. Member AIAA.

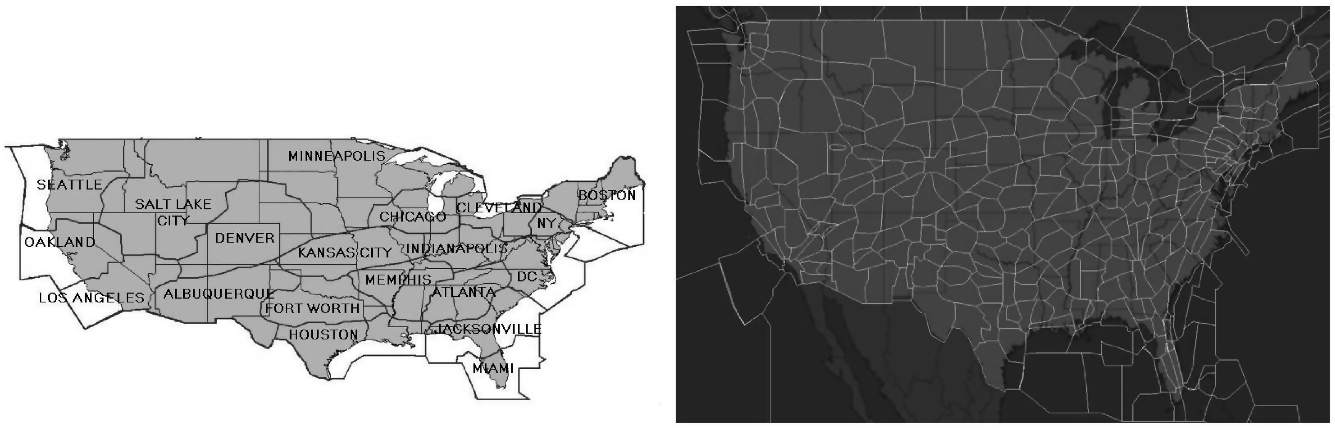


Fig. 1 Current ARTCCs (left), and ATC sectors in continental U.S. airspace (right) (pictures courtesy of Federal Aviation Administration).

generic network flow graphs suggested that their approach held merit. However, there were concerns that the algorithm might be restricted to a limited class of small network graphs due to the computational inefficiency of the proposed algorithm. Recently, airspace configuration was done using integer programming by Yousefi and Donohue [10]. The airspace was first divided into very fine hexagonal grid cells and the workloads of the cells were computed using a simulation tool called the Total Airport and Airspace Modeler. The airspace configuration problem was then formulated as a clustering problem and solved using integer programming to combine the grid cells into sectors. Another clustering-based DAC algorithm was proposed by Brinton and Pledgie [11]. In that paper, they explicitly used the dynamic density [12] as the objective function for airspace configuration. The airspace configuration problem was solved using a geometrical approach by Basu et al. [13] and Mitchell et al. [14]. First, the analytical solution to a one-dimensional problem was derived. The authors then applied the one-dimensional solution framework to the two-dimensional airspace configuration problem using binary airspace partition. Martinez et al. developed an airspace configuration algorithm based on graph partitioning [15]. In their method, an unweighted graph was first constructed based on the current airspace sectors, where the graph vertices are the entry and exit points of traffic flows to each existing sector. The graph was then bisected iteratively using the spectral clustering method until the capacity constraints were satisfied. The sectors and their workloads were constructed in the following ways: the airspace was first divided into fine grid cells, and each cell has a workload based on the aircraft count. These grid cells were then assigned to close vertices using a region-growth method to form the sectors, which have irregular boundaries that are undesirable for ATC. Klein solved the airspace configuration problem using a region-growth method [16], where the NAS was first divided into generic hexagonal grid cells and the sectors (or centers) were generated by growing from a set of seeding cells with the balanced workload as the objective. Recently, both Klein et al. [17] and Brinton et al. [18] proposed new airspace configuration methods based on sector boundary perturbation.

From the literature survey above, the followings are the observations on the design of DAC algorithms:

1) The air-traffic-controller workload is determined by multiple factors. However, many DAC algorithms [13,15,16] mainly focus on one factor, which is the number of aircraft in a sector. Because of the complexity of ATC operations, it is desirable that the sector workload consists of, but is not limited to, the following three components:

a) The first component is the monitoring workload, which is the workload of monitoring aircraft operating within a sector. The monitoring workload is represented by the number of aircraft in the sector.

b) The second component is the coordination workload, which is the workload due to information exchanges between two controllers or between the controllers and the pilot when the aircraft crosses the sector boundary. The coordination workload is represented by the number of aircraft entering or exiting the sector.

c) The third component is the conflict avoidance workload, which is the workload for monitoring crossing air traffic. The conflict avoidance workload is represented by the number of aircraft around the traffic merging/diverging/crossing points.

In this paper, the workload of a sector is computed by a weighted summation of monitoring, coordination, and conflict avoidance workloads, where the weight coefficients are obtained from empirical studies [12,19,20].

2) The designed sectors should satisfy certain constraints, including:

a) The constraint of workload capacity, which is the sector workload, should be below a capacity threshold. For example, the monitor alert parameter is used as the capacity threshold of the total number of aircraft allowed in a sector at a given time in the current ATC system.

b) The constraint of sector shape, which means the sector boundaries should be smooth polylines with a small number of edges. The sector shapes should avoid certain irregular forms, such as long and skinny extrusions.

c) The constraint of sector boundary, which means the sector boundary should be at least a minimum distance away from the airports, waypoints, and main air routes, so that the air-traffic controllers have sufficient time and space to resolve potential conflicts and/or optimize traffic flows.

In this paper, the above constraints are included explicitly into the DAC algorithm design.

3) There are generally three modeling techniques for DAC:

a) The first modeling technique is region based [10,16]. The entire airspace is divided into small cells in the shape of rectangles, hexagons, etc.

b) The second modeling technique is graph based [5,6]. A graph consists of vertices, links, and a weight matrix. The vertices can be airports, waypoints (navigational aids), and/or sector boundary crossing points; the links represent air routes, and the weight of a link represents the amount of traffic through the link.

c) The third modeling technique is trajectory based [7,11,13]. The airspace is built upon the trajectories of individual flights, either explicitly or implicitly, using dynamic densities as a performance measure. There are also methods that combine two different models (e.g., the model in [15] combines the graph-based model with the region-based model).

The benefit of region-based models is that the model is easy to build. The grid cells are grouped into sectors using various clustering algorithms. However, the region-based models cannot capture and use the air-route structure directly. In comparison, the graph-based models naturally inherit the air-route structure. It usually helps to reduce the size of the problem, hence the computational complexity, compared with the region-based models. The trajectory-based models are the most flexible among three models. However, the accuracy of the model is restricted by the trajectory prediction

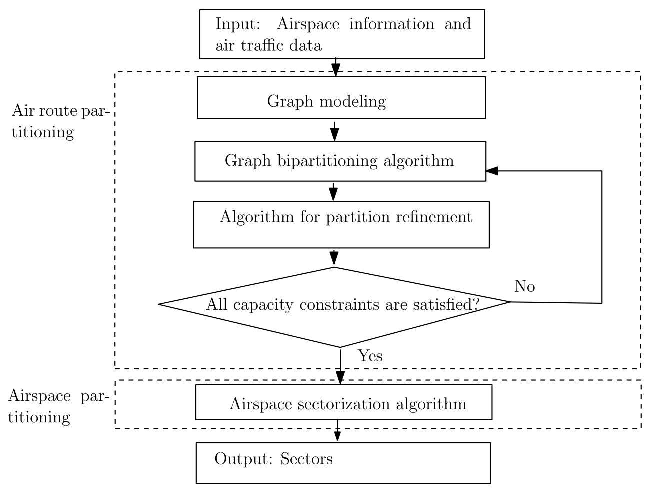
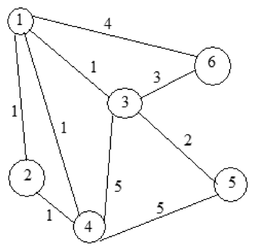

Input: Airspace information and air traffic data
Air route partitioning
Graph modeling
Graph bipartitioning algorithm
Algorithm for partition refinement
All capacity constraints are satisfied?
No
Yes
Airspace partitioning
Airspace sectorization algorithm
Output: Sectors

Fig. 2 Structure of the DAC algorithm.



$$\text{Weight matrix, } W = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 4 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 5 & 2 & 3 \\ 1 & 1 & 5 & 0 & 5 & 0 \\ 0 & 0 & 2 & 5 & 0 & 0 \\ 4 & 0 & 3 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 3 Graph with its corresponding weight matrix.

accuracy over a long period of time. Also, it is generally more computationally expensive than the graph-based models. In this paper, our airspace model is based on a graph. One unique property of our graph model is that all of the flight tracks are used to compute the graph weight matrix, so that the model represents the traffic flow more accurately than the other graph models [15] while remaining computationally efficient.

4) Because of the complexity and the scale of the problem, it is infeasible to find the global optimal solution for airspace configuration. To compare the performances of DAC algorithms, the following are considered as desirable characteristics:

a) The algorithm should be computationally efficient, so that it is feasible to be implemented for the sectorization of the entire NAS in real time or in a short time period.

b) The algorithm should yield consistent performance (for this reason, stochastic optimization techniques or heuristic algorithms are avoided for which the results are inconsistent, even under the same air-traffic conditions).

c) The algorithm should not generate a solution based on a fixed number of sectors given as an input. So the number of sectors should be a parameter to be optimized.

In this paper, a DAC algorithm is developed that has many good characteristics mentioned in the previous paragraph. The proposed DAC algorithm consists of three steps: in the first step, a weighted graph model is built using flight plans and flight tracks that accurately represent the air-route structure and air traffic in the NAS, and in the second step, the workload of a graph (sector) is a weighted summation of monitoring, coordination, and conflict avoidance workloads, which are commonly recognized as three most important factors in the dynamic density. The graph is iteratively bisected into subgraphs with the objective of evenly distributing the workload under the capacity threshold using a spectral clustering method. Also, a refinement algorithm is developed after each graph bisection to remove the disconnected subgraphs and further improve workload balancing and, in the last step, a novel and computationally efficient algorithm is developed, based on the graph search method to sectorize the airspace into sectors that have smooth boundaries while satisfying the sector boundary constraint. Finally, the performance of the proposed DAC algorithm is validated through various air-traffic scenarios with enhanced-traffic-management-system (ETMS) air-traffic data.

The rest of the paper is organized as follows. In Sec. II, the basic concepts of the modeling technique, partitioning algorithm, and sectorization algorithm of the DAC algorithm are reviewed and then presented in detail. In Sec. III, the performance of the proposed DAC algorithm is demonstrated and analyzed through simulations with ETMS air-traffic data. Section IV contains conclusions.

II. Dynamic Airspace Configuration Algorithm

The current Atlanta Center (ZTL) is used to demonstrate the DAC algorithm implementation. Figure 2 shows the structure of the proposed DAC algorithm.

Given air routes and air-traffic data in the airspace, a corresponding graph model is constructed. A graph consists of a set of vertices, links, and a weight matrix (see Fig. 3). In the graph model, the vertices represent waypoints and airports, the links represent segments of air routes, and the amount of air traffic along a link is represented by the corresponding element in the weight matrix. The graph is then simplified to capture only the main air routes and the corresponding graph weight matrix is computed using the real flight tracks by projecting them to their closest graph links. The detailed modeling techniques will be presented in Sec. II.A.

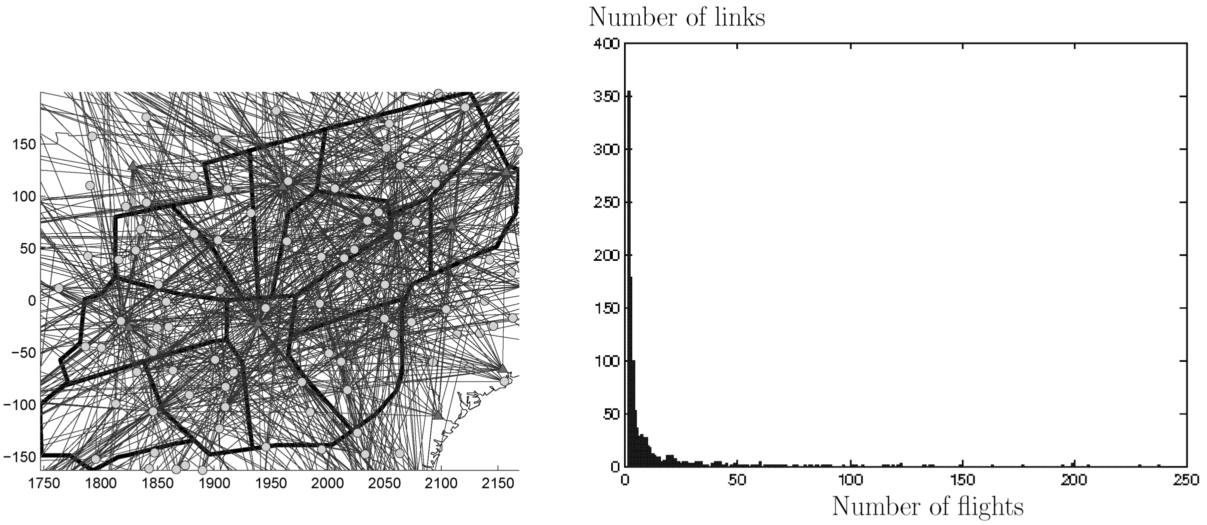


Fig. 4 Air-route structure of ZTL based on the flight plans (left) (triangles: main airports, circles: waypoints, thick lines: current sector boundaries) and histogram of the 24-h accumulated air traffic along the links (right).

Next, the graph is bisected iteratively based on a spectral clustering method until all of the subgraph workloads are below the capacity. Moreover, a refinement algorithm is developed to remove the disconnected subgraphs and further improve the workload balancing. Details of the spectral clustering method and the partition refining algorithm will be discussed in Sec. II.B.

With the partitioned graph as an input, the airspace is sectorized into polygon-shaped sectors such that: a) sectors have smooth boundaries with the small number of edges; and b) the sector boundaries are at least a minimum distance away from the airports, waypoints, and main air routes. The details of the airspace sectorization algorithm will be explained in Sec. II.C.

A. Graph Modeling

The objective is to create a graph model that preserves the important airspace information, such as the locations of airports, waypoints and main air routes using vertices and links, while accurately representing the majority of air traffic using a graph weight matrix. The air-route structure within ZTL is first extracted from the ETMS data based on flight plans. Each flight plan consists of an ordered sequence of waypoints over which an aircraft flies. Those waypoints are at fixed positions. All of the flights are then extracted that pass through ZTL and its 100 nm surrounding area at altitudes ranging between 18,000 and 60,000 ft in one day. Two waypoints are connected by a link only if there are flights passing directly between them. For example, consider a flight plan from the Hartsfield–Jackson Atlanta International Airport (ATL) to the Los Angeles International Airport (LAX), ATL.GEETK4.VUZ.J14.IRW..BGD. J8.GUP.J96.DRK.J231.TNP.SEAVU1.LAX/0416, that contains a section of an air route in ZTL as ATL.GEETK4.VUZ. So there is a direct connection between waypoints ATL and VUZ (Vulcan VORTAC). Figure 4 (left) shows all of the links around ZTL using the 24-h ETMS data on a given day (27 March 2007), where the triangles represent main airports and the circles represent waypoints. From Fig. 4 (left), it is observed that the complete 24-h air-route structure has a very high graph density. However, many of the links have low air traffic, as shown in Fig. 4 (right). To find the main air routes, the links in Fig. 4 (left) are ranked by their 24-h accumulated traffic counts in a descending order. The top links are then extracted to account for 70% of the total air traffic, and the result is shown in Fig. 5.

The following steps are then applied to Fig. 5 to obtain the final graph model: first, the close waypoints and airports are merged if their distance is less than 12 nm; second, all of the crossing points between any pair of links on the graph are detected and then added as new graph vertices (Fig. 6); and third, another graph vertex merging is performed if the distance between any two vertices is less than 8 nm. Then each link is divided evenly into small links, such that the

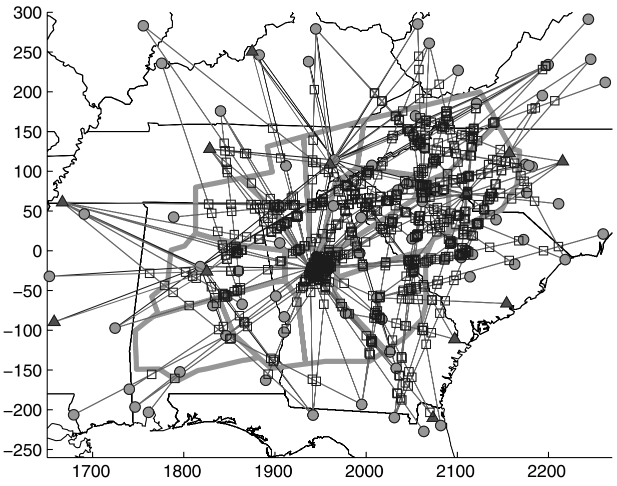


Fig. 6 The graph after merging closer waypoints and then adding all crossing points represented by squares.

length of each new link is approximately 8 nm, which is close to the average aircraft cruising distance in 1 min. This implies that the majority of the flights near a link will move to the adjacent link within a 1-min. time period. Thus, the sum of the weights of the common links between two subgraphs is approximately to be the coordination workload between the two corresponding sectors. Note that in the graph modeling process the locations of 89 major airports in the U.S. are extracted and added as new graph vertices. A new property level is added to each graph vertex to represent the importance of that vertex in the following descending order: the vertices representing the major airports (level = 0), the vertices representing the waypoints (level = 1), the vertices representing the crossing points (level = 2), and the other vertices (level = 3). So the smaller the level of a vertex, the more important the vertex. A set of rules for the vertex merging is described as follows (assume two vertices A and B are to be merged):

1) If $\text{level}_A < \text{level}_B$ (e.g., A is an airport and B is a waypoint), the algorithm will merge B to A, together with all of the links connected to B.

2) If $\text{level}_A < \text{level}_B$, the algorithm will merge A to B together with all of the links connected to A.

3) If $\text{level}_A < \text{level}_B$, the algorithm will select the middle point between A and B as the new vertex that replaces both A and B together with all of the links connected to them. The purpose of the rules is to keep the important components of the center intact or with a minimum amount of change in the final graph model, so that the graph can accurately represent the air routes. Figure 7 shows the final graph model of ZTL.

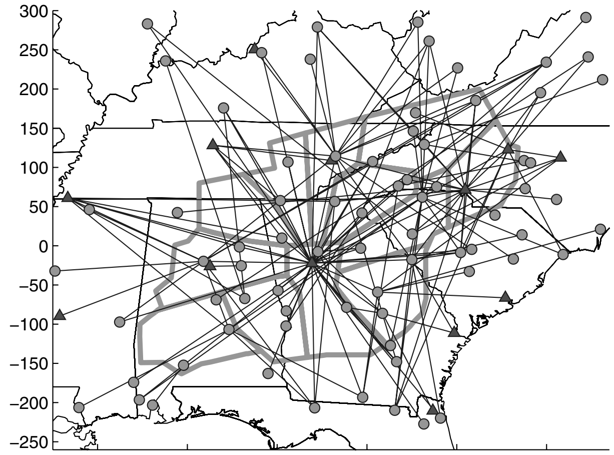


Fig. 5 Air-route structure in ZTL that accounts for 70% of the total air traffic from flight plans (triangles represent the main airports, and circles represent waypoints).

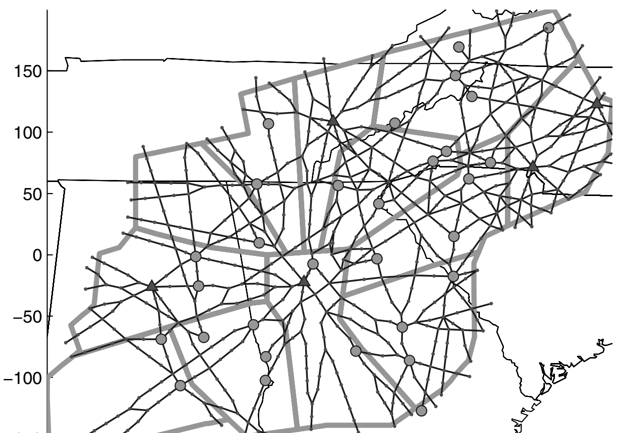


Fig. 7 Final graph model of the ZTL (triangles: main airports, circles: waypoints, thick lines: current sector boundaries).

Finally, in order to determine the graph weight matrix, the real flight tracks are projected to their closest graph links at each minute and the weight of a graph link is the average number of flight tracks that are projected on that link within a given period of time. Note that here 100% flight tracks in ZTL are used to compute the graph weight matrix while only 70% of the total flight plans are used to build the graph links (as shown in Fig. 5). As a result, the weighted graph model represents only the main air-route structure but accounts for all of the air traffic accurately.

Actual flights may deviate from the air routes represented by the graph links. So a metric is defined to measure how accurately the graph model represents the real air traffic. The deviation of a flight from the graph model is denoted as the projection distance between its current position to the closest link of the graph. Figure 8 shows the histogram of deviations of flights from the graph model obtained by using the 2-h air traffic between 1600–1800 hrs Eastern Standard Time (EST) on 27 March 2007 in ZTL. Figure 8 shows that about 70% of the total flights during the 2-h period are within 4 nm of the graph model, about 92% of the total flights are within 8 nm, and only less than 1% of the total flights are deviated from the graph model more than 16 nm, which demonstrates that the final graph model in Fig. 7 accurately represents the main air-traffic structure in ZTL.

B. Graph Partitioning

In this section, the graph model of ZTL (Fig. 7) is discussed formally using graph notation $\mathbb{G} = \mathbb{G}(\mathcal{V}, W)$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices and $W = [w_{ij}] \in \mathbb{R}^{n\times n}$ is a weight matrix whose elements are weights of links. Then, the airspace partitioning problem is to partition $\mathbb{G}$ into a set of K disjoint subgraphs $\mathbb{G}_i$ $(i = 1, 2, \ldots, K)$, with the objective of evenly distributing the monitoring workload while minimizing the total coordination workload, where K is the number of sectors that is to be computed. By disjoint, it means that the two graphs have no vertex in common. Additionally,

$$\mathbb{G} = \cup_{i=1}^{K}\mathbb{G}_i$$

and

$$\mathbb{G}_i \cap \mathbb{G}_j = \emptyset; \qquad i \neq j; \qquad i, j \in \{1, 2, \ldots, K\}$$

From graph notations in the Appendix, it is not difficult to see that the weight $[\omega(\mathbb{G}_i)]$ or degree $[\deg(\mathbb{G}_i)]$ of a subgraph corresponds to the monitoring workload of the sector represented by the subgraph, and the graph cut cost $[\mathrm{cut}(\mathbb{G})]$ corresponds to the total coordination workload among the sectors. Thus, the airspace partitioning problem can be posed as a weighted minimum cut problem, called normalized cut (NCUT) [21], with the following cost function:

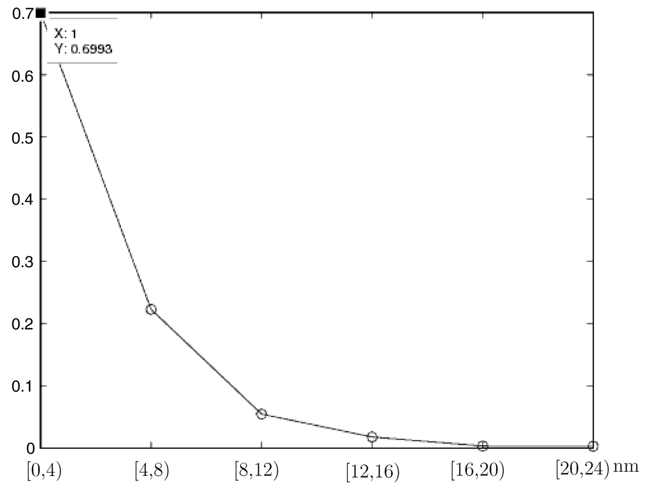


Fig. 8 Histogram of flight-track deviations from the graph model in ZTL, obtained from 2-h ETMS data between 1600 and 1800 hrs EST on 27 March 2007.

$$\min_{H}\left\{J = \mathrm{cut}(\mathbb{G})\sum_{i=1}^{K}\frac{1}{\deg(\mathbb{G}_i)}\right\} \tag{1}$$

where H is the graph indicator matrix.

From (1), it is observed that: 1) the smaller the cut cost, the smaller the cost function J, and b) with a given cut cost, the cost function J is minimum when all subgraph degrees are equal. Thus, the graph cut cost is minimized while balancing the subgraph weight by minimizing the cost function J in (1), which is equivalent to minimizing the coordination workload while balancing the sector monitoring workload for the airspace configuration problem.

Finding the exact solution to the NCUT problem is proved to be nondeterministic polynomial-time hard [22]. The NCUT problem is relaxed to an eigenvalue problem using the spectral clustering method which can be solved efficiently [23]. For the standard spectral clustering algorithm, the computational complexity is $\mathcal{O}(n^3)$, where n is the number of graph vertices. Note that the graph weight matrix in this application is a sparse matrix, so that the spectral clustering algorithm can be implemented more efficiently. The graph is bisected iteratively until all of the subgraph workloads are below their capacity thresholds. Here, the workload of a subgraph $\mathbb{G}_k$, $\mathrm{WL}(\mathbb{G}_k)$, is computed by a weighted summation of monitoring, coordination, and conflict avoidance workloads as follows:

$$\mathrm{WL}\,(\mathbb{G}_k) = \alpha\mathrm{WL}_{\mathrm{monitor}} + \beta\mathrm{WL}_{\mathrm{coordination}} + \gamma\mathrm{WL}_{\mathrm{conflict}} \tag{2}$$

where

$$\mathrm{WL}_{\mathrm{monitor}} = \frac{1}{2}\sum_{v_i,v_j\in V_k}\omega_{i,j}$$

$$\mathrm{WL}_{\mathrm{coordination}} = \sum_{v_i\in V_k, v_j\notin V_k}\omega_{i,j}$$

$$\mathrm{WL}_{\mathrm{conflict}} = \frac{1}{T}\sum_{m=1}^{T}\sum_{\mathrm{level}_j=1}\left[\sum_{v_i\in V_k}\omega_{i,j}(m) - 1\right]$$

where $W = [\omega_{i,j}]$ is the graph weight matrix computed by averaging the accumulated air traffic projected on link ij over a time period of T, and $\omega_{i,j}(m)$ represents the number of aircraft projected on the link ij at time m. Thus,

$$\omega_{i,j} = \frac{1}{T}\sum_{m=1}^{T}\omega_{i,j}(m) \tag{3}$$

Equation (3) computes the link weight by taking the arithmetic mean of air traffic through the link over a time period of T. In contrast, the link weight can also be computed using the worst-case air-traffic conditions:

$$\omega_{i,j} = \max_{m=1,2,\ldots,T}\omega_{i,j}(m) \tag{4}$$

Through simulations using ETMS air-traffic data, it is found that sectorization using the worst-case weight matrix (4) is overly conservative most of the time, and yet sectorization using the mean weight matrix (3) shows a good average performance and comparable or slightly improved worst-case performance during the majority of the time, compared with the current sectorization. The formula to compute the graph weight matrix is modified in such a way that it would provide us the flexibility to balance between the average and the worst-case performances in two ways:

1) Assume $\omega_{i,j}$ is a random variable and $\omega_{i,j}(m)$ and $(m = 1, 2, \ldots, T)$ are T random samples of $\omega_{i,j}$. Let $\mu_{i,j}$ and $\sigma_{i,j}$ be the sample mean and sample standard deviation of $\omega_{i,j}(m)$ and $(m = 1, 2, \ldots, T)$, respectively. Then the weight of link ij is computed by

$$\omega_{i,j} = \mu_{i,j} + \delta\sigma_{i,j} \tag{5}$$

where $\delta \geq 0$ is a design parameter. Note that if $\delta = 0$, then $\omega_{i,j}$ in (5) equals to $\omega_{i,j}$ in (3). If $\mu_{i,j} + \delta\sigma_{i,j} > \max_{m=1,2,\ldots,T}\omega_{i,j}(m)$, set $\omega_{i,j} = \max_{m=1,2,\ldots,T}\omega_{i,j}(m)$.

2) The weight of link ij can also be computed using the generalized mean:

$$\omega^{p}_{i,j} = \left\{\frac{1}{T}\sum_{m=1}^{T}[\omega_{i,j}(m)]^{p}\right\}^{1/p} \tag{6}$$

where $p \geq 1$ is a design parameter.

The generalized mean in (6) has the following properties which are desirable to compute the graph weight:

1) If $p = 1$, $\omega^{1}_{i,j}$ is the arithmetic mean in (3).

2) If $p \to \infty$, $\omega^{\infty}_{i,j}$ is the worst-case mean (i.e., the maximum) in (4).

3) In general, $\omega^{p}_{i,j} \geq \omega^{q}_{i,j}$ if $p > q$; they are equal if, and only if, $\omega_{i,j}(k)$ is a constant.

By using (5) and (6) instead of (3) and (4) for computing the graph weight matrix, the values of the elements of the graph weight matrix can be varied between its mean and maximum by choosing different parameters $\delta \in [0, \infty)$ in (5) or $p \in [1, \infty)$ in (6). The optimal values of δ and p can be determined from experiments.

Next, the spectral clustering algorithm for graph bisection is summarized as follows:

Algorithm 1 is a graph bisection using spectral clustering with the input of degree matrix D and the Laplacian matrix L of a graph to be bisected.

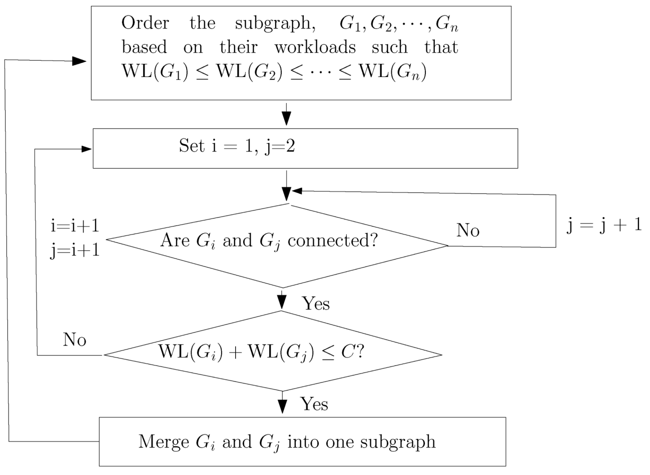

Order the subgraph, $G_1, G_2, \cdots, G_n$ based on their workloads such that $\mathrm{WL}(G_1) \leq \mathrm{WL}(G_2) \leq \cdots \leq \mathrm{WL}(G_n)$
Set i = 1, j=2
i=i+1
j=i+1
Are G_i and G_j connected?
No
j = j + 1
Yes
No
$\mathrm{WL}(G_i) + \mathrm{WL}(G_j) \leq C$?
Yes
Merge G_i and G_j into one subgraph

Fig. 9 Flowchart for subgraph merging in the refinement algorithm.

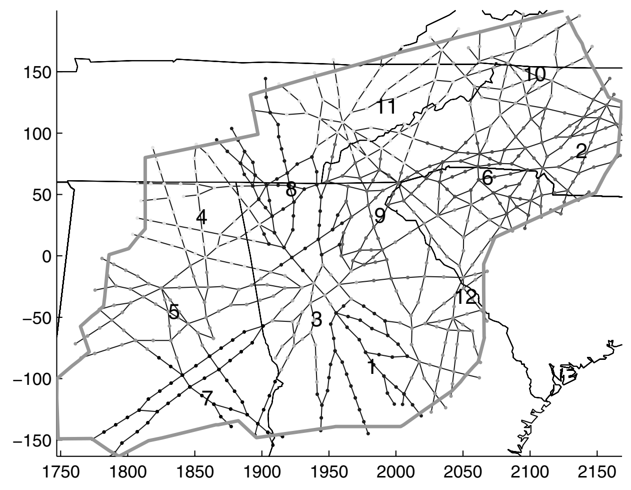


Fig. 10 A partitioned graph model of ZTL with 12 subgraphs.

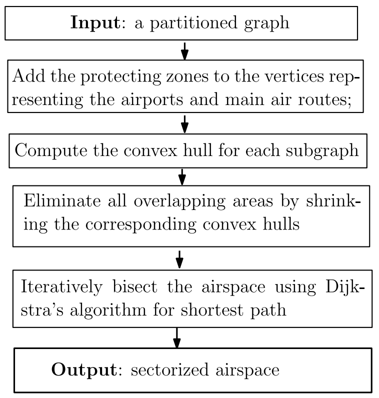

Input: a partitioned graph
Add the protecting zones to the vertices representing the airports and main air routes;
Compute the convex hull for each subgraph
Eliminate all overlapping areas by shrinking the corresponding convex hulls
Iteratively bisect the airspace using Dijkstra's algorithm for shortest path
Output: sectorized airspace

Fig. 11 The flowchart for the airspace sectorization algorithm.

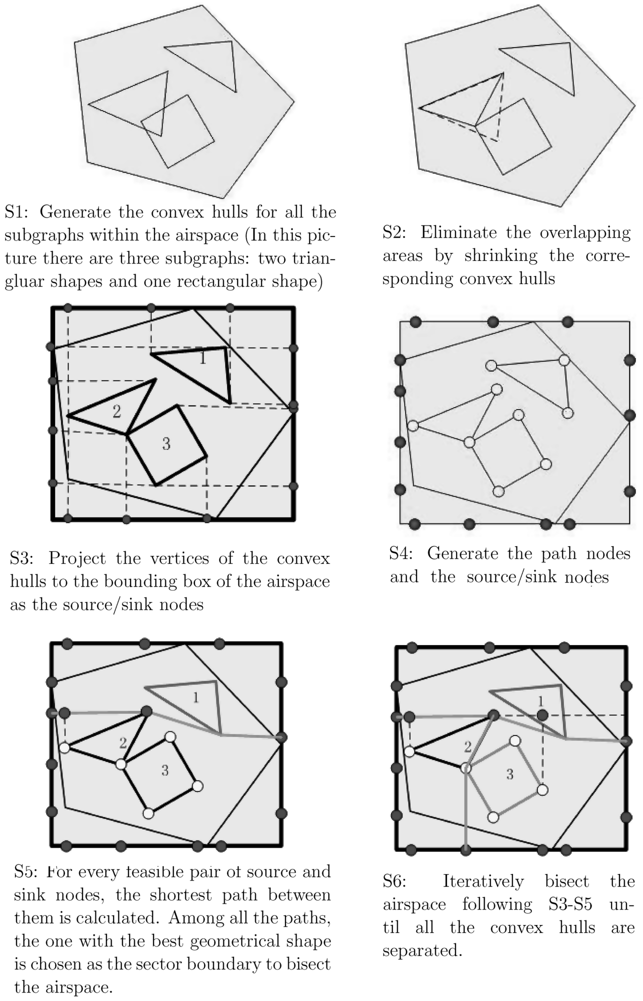
1) Compute the eigenvectors, $u_1, u_2, \ldots, u_k (k \geq 2)$, corresponding to the first k smallest eigenvalues of the generalized eigenvalue problem:

$$Lu = \lambda Du \tag{7}$$

2) Set $U = [u_1, u_2, \ldots, u_k] \in \mathbb{R}^{n\times k}$. Let $y_i (i = 1, 2, \ldots, n)$ be the vector corresponding to the ith row of U. Apply the K-means algorithm to cluster n points in the k-dimensional space (i.e., $\{y_i\}^{n}_{i=1}$) into two clusters, C_1 and C_2. Then,

$$v_i \begin{cases} \in \mathcal{V}_1, & \text{if } y_i \in C_1 \\ \in \mathcal{V}_2, & \text{if } y_i \in C_2 \end{cases}; \qquad i = 1, 2, \ldots, n \tag{8}$$

□



S1: Generate the convex hulls for all the subgraphs within the airspace (In this picture there are three subgraphs: two trianlgluar shapes and one rectangular shape)

S2: Eliminate the overlapping areas by shrinking the corresponding convex hulls

S3: Project the vertices of the convex hulls to the bounding box of the airspace as the source/sink nodes

S4: Generate the path nodes and the source/sink nodes

S5: For every feasible pair of source and sink nodes, the shortest path between them is calculated. Among all the paths, the one with the best geometrical shape is chosen as the sector boundary to bisect the airspace.

S6: Iteratively bisect the airspace following S3-S5 until all the convex hulls are separated.

Fig. 12 Illustration of the airspace sectorization algorithm.

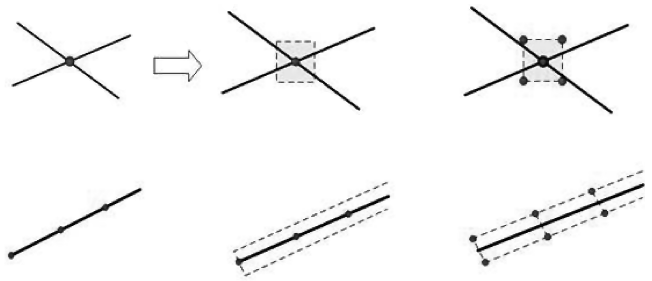


Fig. 13 Illustration for incorporating the minimum distance constraint by adding the protection zone to the vertex representing airport/ waypoint (top) and air route (bottom).

The preceding spectral clustering algorithm shows good performance for partitioning large graphs, with the objective function defined in (1) at low computational costs. However, there are still two problems that need to be addressed after each graph bisection. First, the partitioned subgraphs may have disconnected components, which is unacceptable because all of the sectors must be connected. Second, the subgraph workload is not well balanced. So a refinement algorithm is proposed to address both problems, and the functions of the refinement algorithm include 1) separating disconnected components of each subgraph, 2) rebalancing the subgraph workload, and 3) merging the adjacent subgraphs.

1) Separating disconnected components can be done based on the following Lemma, using the graph Laplacian matrix.

Lemma 1: Let $\mathbb{G}$ be an undirected graph. Then, the multiplicity k of the eigenvalue 0 of the graph Laplacian matrix L is equal to the number of components in the graph. Moreover, the eigenspace of the eigenvalue 0 is spanned by the indicator vectors of these components [23].

Lemma 1 is implemented in the following way:

Algorithm 2 identifies and separates the disconnected components of a graph. First, compute all of the eigenvectors, $u_1, u_2, \ldots, u_k$, corresponding to the eigenvalue 0 of L. If $k = 1$, then stop, because the graph is connected. Otherwise, set $U = [u_1, u_2, \ldots, u_k] \in \mathbb{R}^{n\times k}$. Let $y_i (i = 1, 2, \ldots, n)$ be the vector corresponding to the ith row of U. Apply the K-means algorithm to partition $\{y_i\}_{i=1}^n$ into k clusters. Each cluster represents a connected component of the original subgraph. □

2) The rebalancing of the subgraph workload, is improved by shifting boundary vertices from the subgraph with the highest workload to the adjacent subgraphs, as well as to the subgraph with the lowest workload from the adjacent subgraphs, iteratively. Because of shifting of vertices, the subgraphs may be disconnected. Such disconnected components will be separated and merged with their adjacent low workload subgraphs at each iteration.

3) To merge the adjacent sectors, the low workload subgraphs are merged to their adjacent subgraphs if their combined subgraphs do not violate the capacity threshold. As a result, it reduces the total number of subgraphs (thus sectors) and further improves subgraph workload balancing. The flowchart of the algorithm for subgraph merging is shown in Fig. 9.

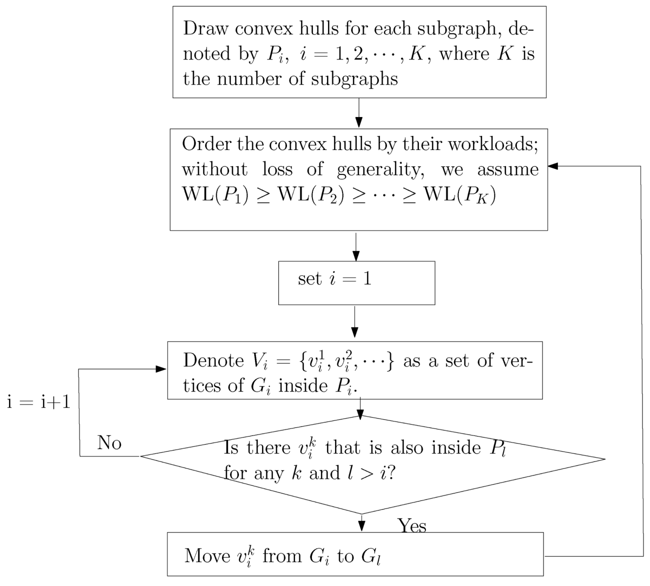

Draw convex hulls for each subgraph, denoted by P_i, $i = 1, 2, \cdots, K$, where K is the number of subgraphs
Order the convex hulls by their workloads; without loss of generality, we assume $\mathrm{WL}(P_1) \geq \mathrm{WL}(P_2) \geq \cdots \geq \mathrm{WL}(P_K)$
set $i = 1$
Denote $V_i = \{v_i^1, v_i^2, \cdots\}$ as a set of vertices of G_i inside P_i.
Is there v_i^k that is also inside P_l for any k and $l > i$?
No
i = i+1
Yes
Move v_i^k from G_i to G_l

Fig. 15 The flowchart of an algorithm for removing the overlapping areas between any pair of convex hulls.

A final partitioned graph of ZTL is shown in Fig. 10, where the graph is partitioned into 12 subgraphs, and all of the subgraph workloads are below the capacity threshold.

C. Airspace Sectorization

Given the partitioned graph (Fig. 10) as an input, the final step is to sectorize the entire airspace into mutually exclusive areas (called sectors) for ATC. Figure 11 shows the flowchart for the proposed airspace sectorization algorithm, and Fig. 12 illustrates the algorithm implementation steps.

1. Generating the Convex Hulls

For ATC, the sector boundaries need to be a certain distance away from the airport terminal areas and the main air routes, so that the controller will have sufficient time and space to control the traffic flow and avoid any potential conflicts. The minimum distance constraints between airports/waypoints and the sector boundary are imposed by adding the protection zones centered at the vertices representing the airports/waypoints as shown in Fig. 13 (top). The length of the square protection zone is $2d$ nm, where d is the required minimum distance. So the minimum distance between the vertex at the center of the protection zone and the boundary of the protection zone is at least d nm. Next, the vertex at the center is replaced with the

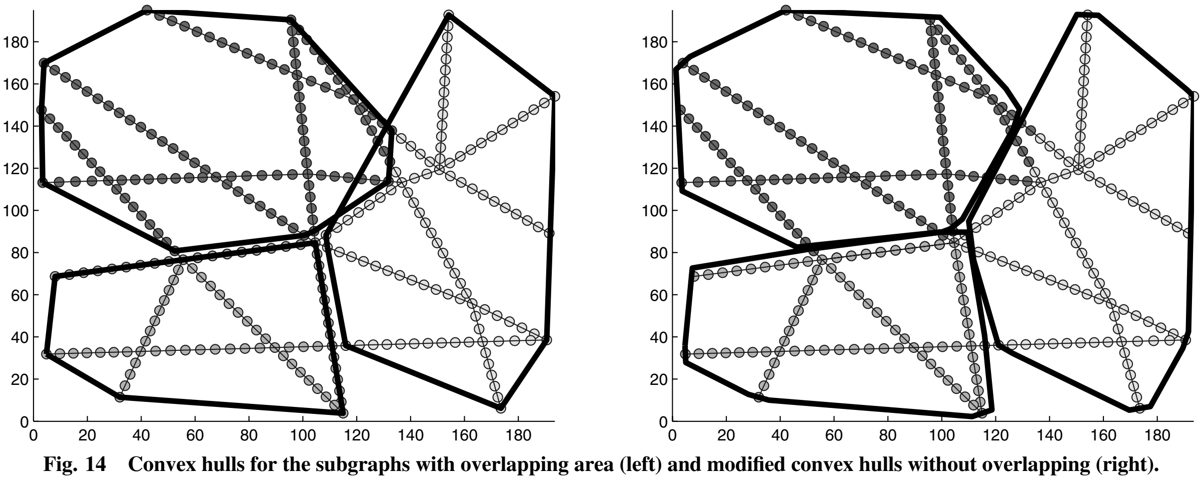


Fig. 14 Convex hulls for the subgraphs with overlapping area (left) and modified convex hulls without overlapping (right).

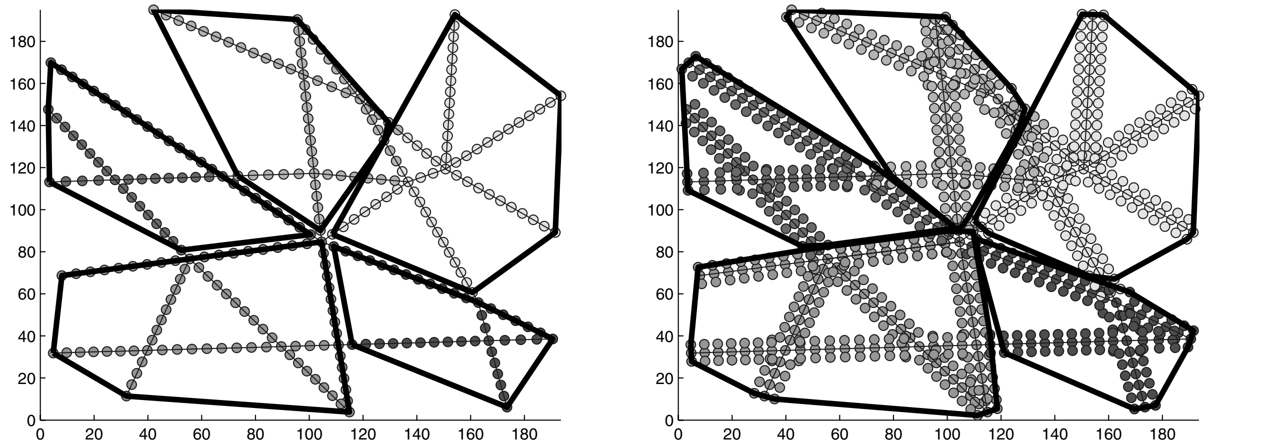


Fig. 16 Nonoverlapping convex hulls without adding the protection zones (left) and nonoverlapping convex hulls with the protection zones (right).

corner points of the protection zone as new subgraph vertices, so that a path (sector boundary) can pass through the boundary of the protection zone only. By doing so, it guarantees that the vertices representing the airports/waypoints will be at least d nm away from the sector boundary. A similar idea is used to incorporate the minimum distance constraint between air routes and the sector boundary, as illustrated in Fig. 13 (bottom).

Next, a convex hull is computed for each subgraph [see Fig. 14 (left) for illustration]. Then the sectorization problem becomes a problem of partitioning the airspace into sectors such that each sector contains one convex hull. From Fig. 14 (left), such a sectorization does not exist when two or more convex hulls overlap each other since the sector boundaries cannot cut through the convex hulls. An algorithm (Fig. 15) is proposed to remove the overlapping areas between any pair of convex hulls by moving the boundary vertices from the high workload subgraph to the low workload subgraph and then recompute convex hulls until all of the convex hulls do not overlap each other. Figure 14 compares the convex hulls before and after the implementation of the algorithm. Figure 16 shows nonoverlapping convex hulls with and without the protection zones. Next, the gap areas are assigned to their neighboring convex hulls to form the sectors. An iterative airspace bisection algorithm is proposed based on the graph search method, which will be discussed in Sec. II.C.2.

2. *Iterative Bisection Algorithm Using Shortest-Path Searching*

As shown in Fig. 12, the idea of bisecting the airspace is to find a path as a sector boundary that passes along a set of nodes in the gap area. Such airspace bisection can be divided into two steps:

1) Step 1 is to choose the source and sink nodes on the sector boundary and a set of nodes inside the sector that guarantee that there exists at least one path to bisect the airspace, which means that there is at least one convex hull on each side of the path.

2) Step 2 is to select one optimal path among all feasible paths, in the sense that it provides the best sector boundary from the ATC perspective. The rest of this section is focused on explaining how the sectorization algorithm is implemented in an effective and optimal way.

A path bisecting the airspace exists if the set of nodes are chosen based on the following fact.

Fact 1: If the convex hulls are pairwise disjoint, there must exist at least one path through a set of nodes within the airspace that bisects the airspace if the set of nodes includes all of the convex hull vertices. □

For example, such nodes satisfying fact 1 are the points in Fig. 12 (S4). Moreover, there is generally more than one path that can bisect the airspace, which is good for our application because it gives the flexibility to choose the one with the best geometrical shape among all feasible paths for ATC.

Next, the convex hull vertices are projected to their closest airspace boundary edges as the source and sink nodes. To reduce the computational complexity, those convex hull vertices are actually projected to the closest edges of the bounding box of the airspace as shown in S3 and S4 in Fig. 12 (if not blocked by convex hulls), where the points in S4 are the final source and sink nodes. Finally, for all feasible pairs of source and sink nodes (e.g., it is infeasible if the source and sink nodes are on the same edges), the shortest paths are computed using Dijkstra's algorithm [8]. The computational complexity of the standard Dijkstra's algorithm is $\mathcal{O}(V^2)$, where V is the number of vertices and, in this application, V is the summation of the number of convex hull vertices. Among all shortest paths, the one with the best geometrical shape is chosen as a sector boundary, in the sense that it bisects the airspace with the minimum number of edges and without having sharp turning angles between the connected edges. The bisection algorithm runs iteratively until all of the convex hulls are separated by the shortest paths. Figure 17 shows the sectorization results of Fig. 16, using the proposed sectorization algorithm.

In summary, our sectorization algorithm has the following benefits: the proposed algorithm can sectorize the airspace into sectors with smoother polygonal boundaries than some other algorithms [15,16]. Our sectorization algorithm is computationally more efficient than the other methods that use constrained mathematical programming or region-growth methods to compute the sector boundary [10,15,16,24]. The sector boundaries computed by the proposed DAC algorithm satisfy the minimum distance constraint. The proposed DAC algorithm does not need an initial sectorization, which is different from the work done by Xue [7]. It is unnecessary to apply any curve and/or polygon simplification algorithms, such as the Douglas–Peucker algorithm [8], to smooth the sector boundaries at the final step [11,13], since the sector boundaries generated by our algorithm are smooth enough.

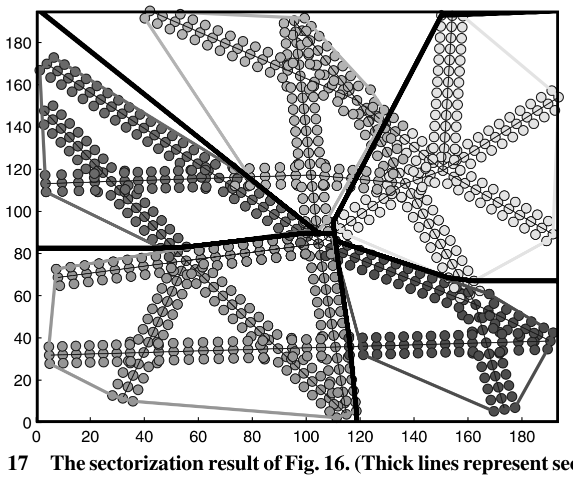


Fig. 17 The sectorization result of Fig. 16. (Thick lines represent sector boundaries.)

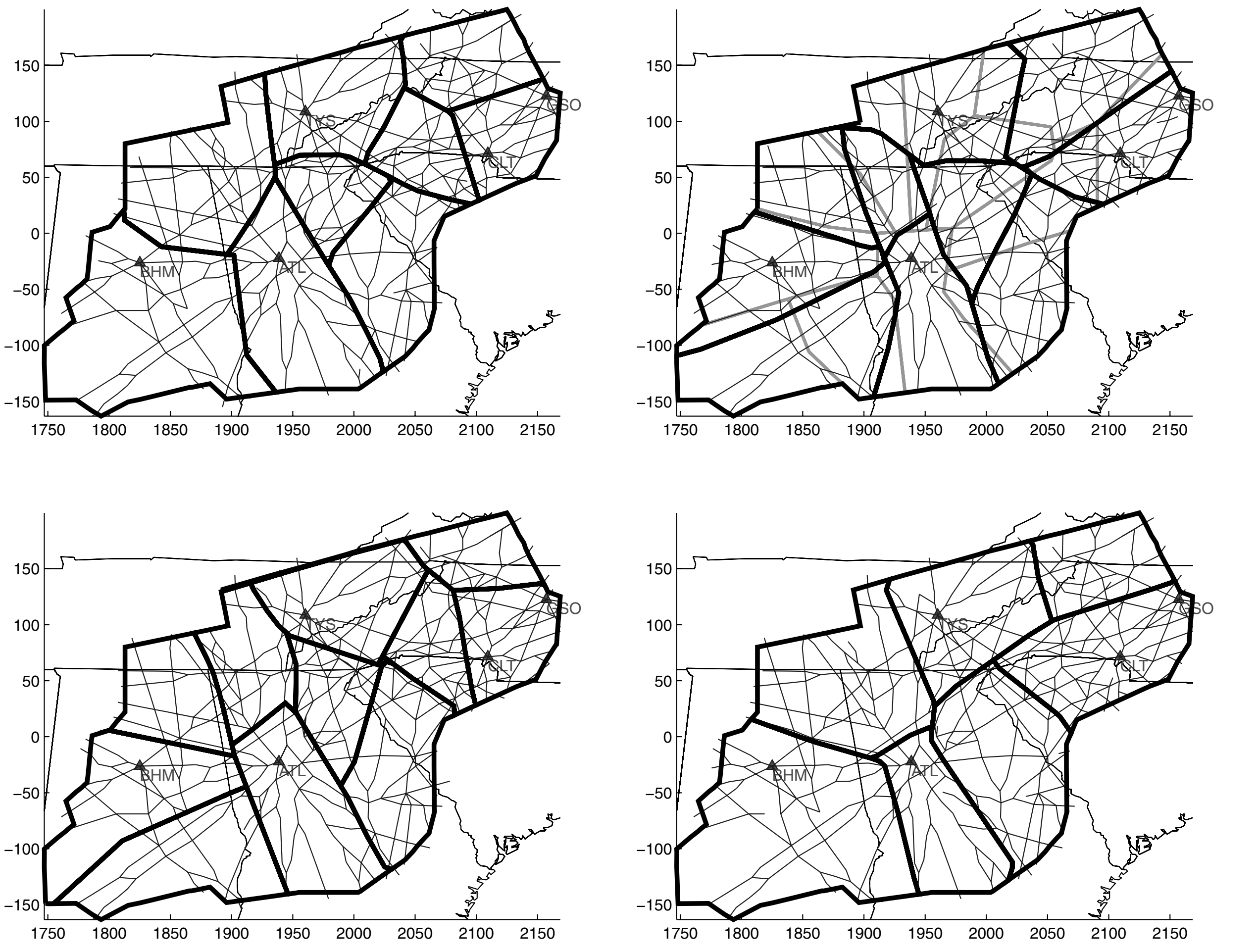


Fig. 18 New sectors of ZTL, generated by the proposed DAC algorithm for four different time intervals (ETMS data on 27 March 2007): (top left) 0700–0900 hrs EST (9 sectors), (top right) 1100–1300 hrs EST (10 sectors), (bottom left) 1700–1900 hrs EST (11 sectors), and (bottom right) 2100–2300 hrs EST (7 sectors).

III. Simulations

In this section, our DAC algorithm is validated with real air-traffic data. The ZTL is used for simulations. For initialization, the workload capacity is set to be 130, based on the historical air-traffic data for ZTL. The minimum distance is set 15 nm between airports and the sector boundary, 9 nm between waypoints and the sector boundary, and 3 nm between air routes and the sector boundary. To compute the subgraph (sector) workload using (2), the coefficients are chosen based on a study by Masalonis et al. [19], where

$$\alpha = 4.25; \qquad \beta = 3.06; \qquad \gamma = 4.5 \tag{9}$$

From experiments, it is found that when $\delta = 0.3$ in (5) or $p = 1.4$ in (6), the DAC algorithm gives good sectorization results with the balanced average and worst-case performances. So in this section, all of the graph weight matrices are computed using (6) with $p = 1.4$. Figures 18 and 19 show different sectorizations of ZTL, computed by the proposed DAC algorithm for five different time intervals on 27 March 2007. From Fig. 18 and 19, it is observed that 1) all of the sectors have smooth boundaries and good geometrical shapes, and 2) the number of sectors computed by the proposed DAC algorithm varies over time (Table 1). This is reasonable, because the amount of air traffic is varying over time. So there are fewer sectors when the traffic is low, and there are more sectors when the traffic is high. This demonstrates that the new sectorization computed by our DAC algorithm is adaptive to the time-varying air traffic with a varying number of sectors. Furthermore, our DAC algorithm is tested for the other centers, and Fig. 20 shows the sectorizations of the ZOB, Dallas Center (ZFW), Kansas Center (ZKC), and Denver Center (ZDV), where all graph weight matrices are computed using 2-h ETMS data

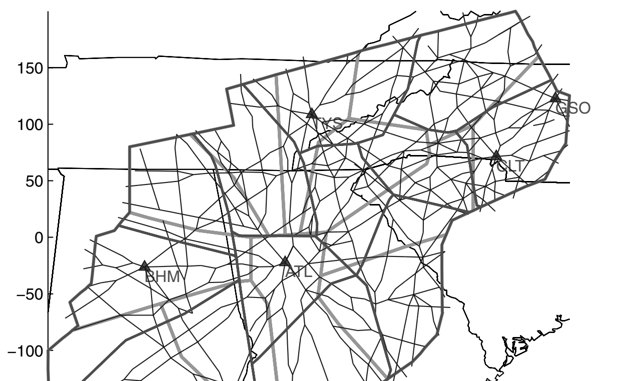


Fig. 19 New sectors of ZTL, generated by the proposed DAC algorithm with the 1500–1700 hrs EST ETMS data on 27 March 2007 (12 sectors).

Table 1 Number of sectors in the new sectorizations for ZTL in Figs. 18 and 19

Time, hrs EST	No. of sectors
0700–0900	9
1100–1300	10
1500–1700	12
1700–1900	11
2100–2300	7

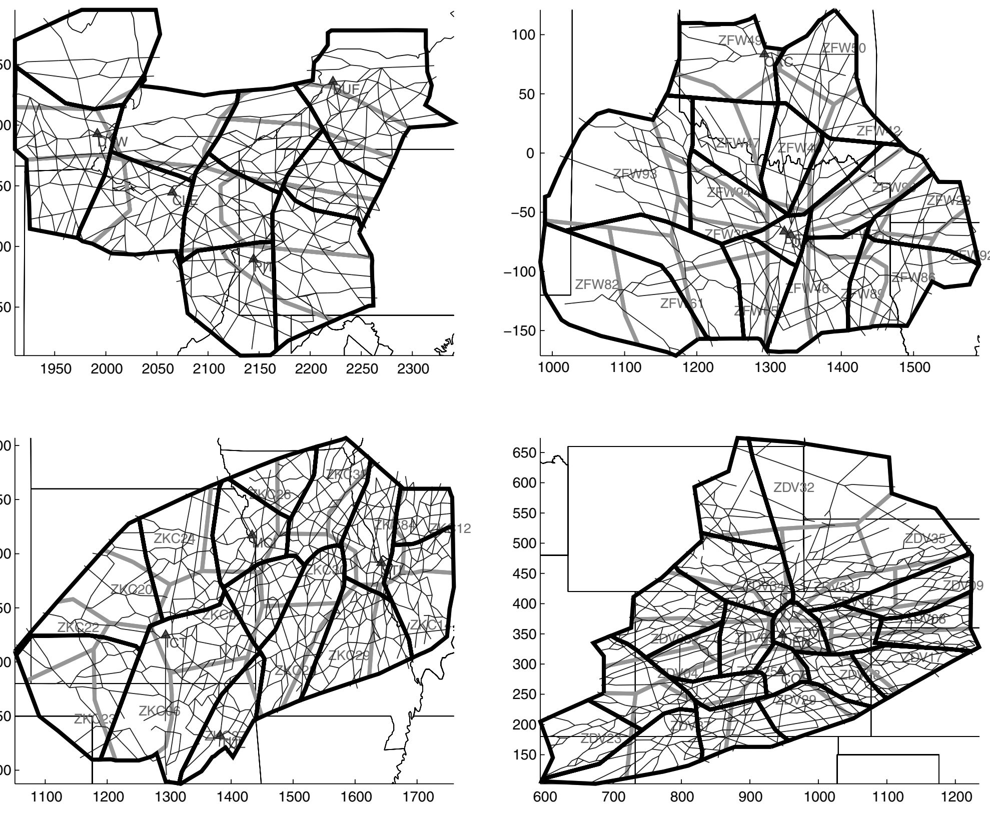


Fig. 20 New sectorizations of four centers, generated by the proposed DAC algorithm: (top left) ZOB, (top right) ZFW, (bottom left) ZKC, and (bottom right) ZDV (thick lines represent boundaries of new sectorization and current sectorization).

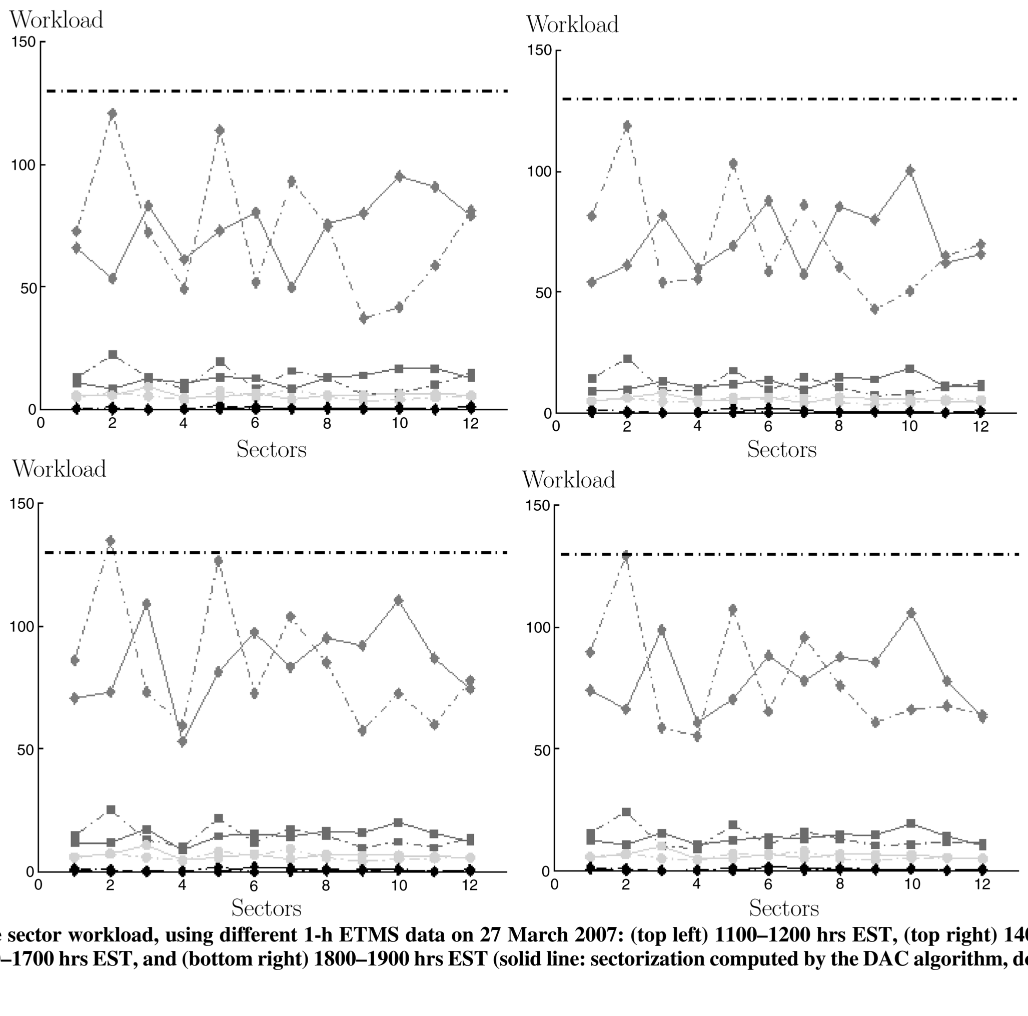


Fig. 21 Average sector workload, using different 1-h ETMS data on 27 March 2007: (top left) 1100–1200 hrs EST, (top right) 1400–1500 hrs EST, (bottom left) 1600–1700 hrs EST, and (bottom right) 1800–1900 hrs EST (solid line: sectorization computed by the DAC algorithm, dotted line: current sectorization).

Table 2 Performance comparison between current sectors and new sectors (the values are workloads)

	Current sectors			New sectors		
Time, hrs EST	Mean	Stdev	c_{bal}	Mean	Stdev	c_{bal}
1100–1200	72.30	26.77	69%	73.86	15.27	50%
1400–1500	70.54	22.77	64%	72.08	14.58	46%
1600–1700	84.18	25.35	57%	85.61	16.58	52%
1800–1900	78.03	22.83	57%	79.74	14.01	42%

between 1500–1700 hrs EST on 27 March 2007. Note that the numbers of sectors in the centers can vary based on the workload thresholds, which are the design parameters.

In the rest of this section, the performance of new sectorization at ZTL is analyzed and compared with the current sectorization. The new sectorization for performance evaluation is shown in Fig. 19, which is generated by the proposed DAC algorithm with 1500–1700 hrs EST ETMS data.

A. Average Performance

For the average performance, the average sector workload is computed at different 1 h intervals on 27 March 2007, and the results are shown in Fig. 21. Note that, here, the ETMS data for evaluation are different from those used to compute the graph model to show the robustness of the proposed DAC algorithm. Figure 21 shows that the sectors computed by the proposed DAC algorithm have more balanced workload than the current sectors. The mean of the average sector workload, the standard deviation of the average sector workload, and the coefficient of sector workload balancing (c_{bal}) are calculated from Fig. 21, and the results are summarized in Table 2, where c_{bal} is defined as

$$c_{\text{bal}} = (\text{WL}_{\max} - \text{WL}_{\min})/\text{WL}_{\max} \times 100\%$$

where $\text{WL}_{\max}$ and $\text{WL}_{\min}$ represent the maximum and minimum sector workloads, respectively. The standard deviation measures how much the sector workload deviates from the mean workload, while c_{bal} measures the maximum difference among sector workloads. Thus, the less the standard deviation and/or c_{bal}, the more balanced the sector workload. From Table 2, new sectors have significantly smaller values of the standard deviation and c_{bal} than those of the current sectors at all four different hours, which validates that the new sectorization has more balanced workload than the current sectorization.

B. Worst-Case Performance

It is important to evaluate the worst-case performance of sectorization for safe operations of aircraft in sectors. In this section, two metrics are used to measure the worst-case performance of sectorization. The first metric is to count the number of capacity violations, which is shown in Fig. 22, where the x axis is time in minutes, and the y axis is the number of sectors that violate the capacity threshold (130). Figure 22 clearly shows that the sectors computed by our DAC algorithm have significantly fewer capacity violations at all four different hours than the current sectors.

For safe operation of aircraft, not only are the number of capacity violations (shown in Fig. 22) important but also the durations of capacity violations, since sustained capacity violations can degrade the performance of air-traffic controllers significantly. Thus, another

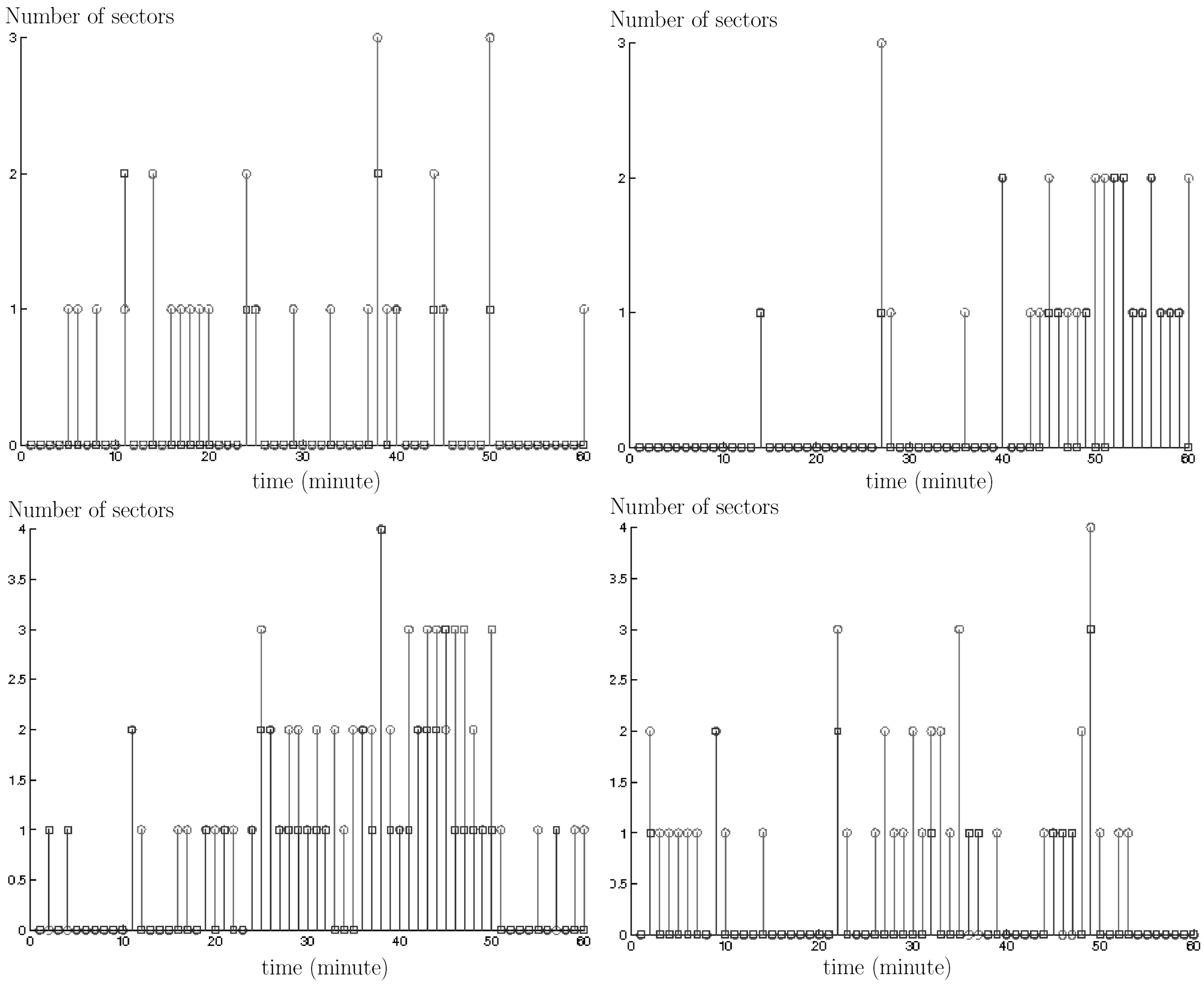


Fig. 22 Number of overcapacity sectors, using different 1-h ETMS data on 27 March 2007: (top left) 1100–1200 hrs EST, (top right) 1400–1500 hrs EST, (bottom left) 1600–1700 hrs EST, and (bottom right) 1800–1900 EST (circle: current sectorization, square: sectorization computed by the DAC algorithm).

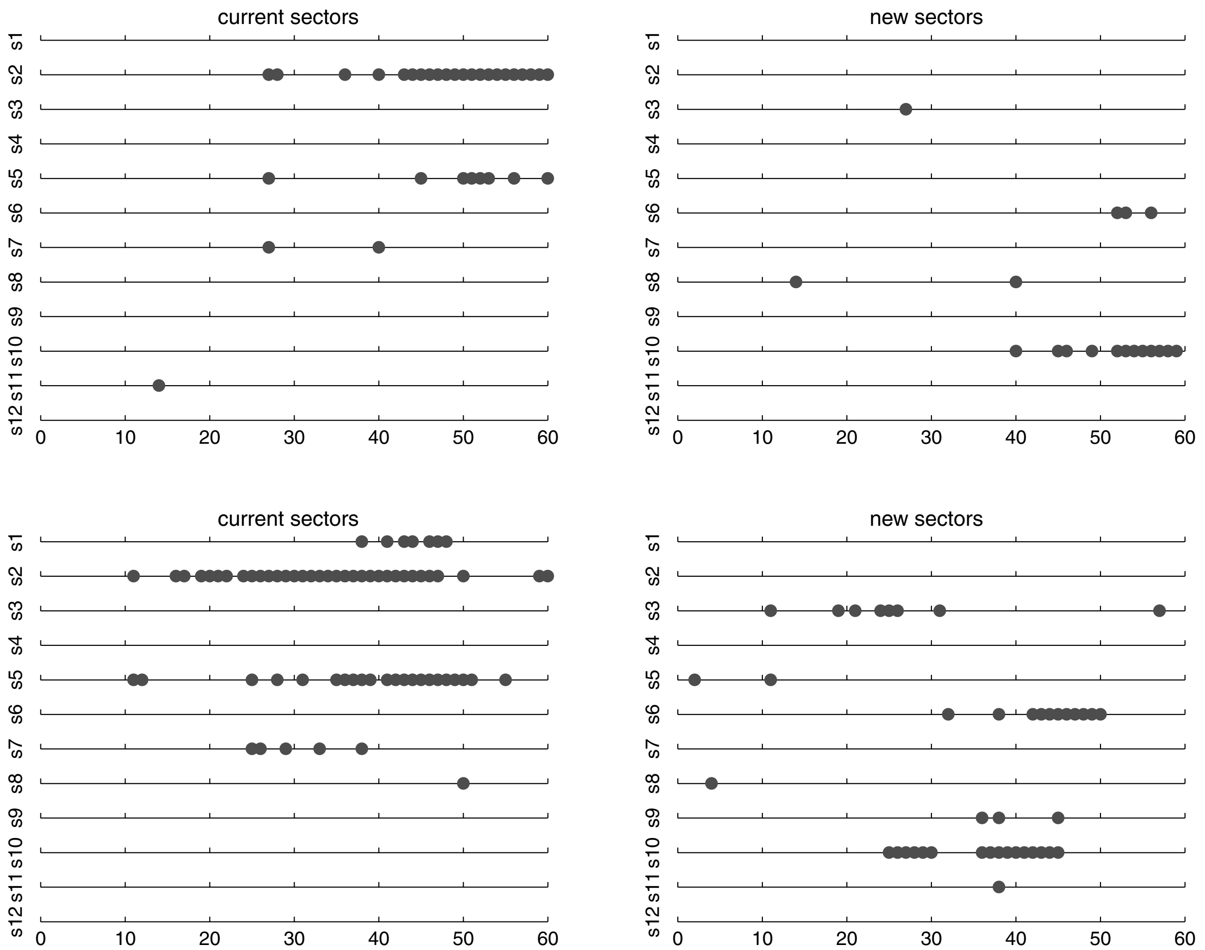


Fig. 23 Sector capacity violations (dots) at 1-min. interval, using different 1-h ETMS data on 27 March 2007: (top left) current sectorization between 1400 and 1500 hrs EST, (top right) sectorization computed by the DAC algorithm between 1400 and 1500 hrs EST, (bottom left) current sectorization between 1600 and 1700 hrs EST, and (bottom right) sectorization computed by the DAC algorithm between 1600 and 1700 hrs EST (*x* axis represents time in minutes, and *y* axis represents the sectors).

metric, the time of sustained capacity violation, is introduced as shown in Fig. 23. Figure 23 shows the exact times of the sector capacity violations for both the current sectors and the new sectors computed by the DAC algorithm. Even if the amounts of accumulated times of the sector capacity violations of the two sectorizations are similar in some cases, it is much undesirable if overcapacity occurs continuously rather than sporadically. Based on this metric, Fig. 23 shows that the new sectors are better than the current sectors, because the durations of capacity violations of the sectorizations computed by the DAC algorithm are much shorter than those of the current sectors. For example, for the current sectorization, the largest time interval of continuous capacity violations is 17 min. during 1400–1500 hrs EST [s2 in Fig. 23 (top left)] and 24 min. during 1600–1700 hrs EST [s2 in Fig. 23 (bottom left)] on 17 March 2007. In comparison, they are reduced to 7 min. [s10 in Fig. 23 (top right)] and 10 min. [s10 in Fig. 23 (bottom right)] in the new sectorization (more than 50% reduction).

IV. Conclusions

In this paper, a graph-based DAC algorithm has been developed. The airspace graph model is built by using flight plans and flight tracks that accurately represent the air-route structure and air-traffic flow. The airspace configuration problem is then formulated as a graph-partitioning problem, which can be efficiently solved using the spectral clustering method. With the partitioned graph as an input, a novel airspace sectorization algorithm is proposed using the graph search method, so that the new sectors have smooth boundaries and good geometrical shapes. The sector boundaries also satisfy the minimum distance requirement. The proposed DAC algorithm has been tested and validated with real air-traffic data. The simulation results show that 1) the proposed algorithm can generate sectors in response to the air-traffic variation over time, thus the airspace capacity can be managed more efficiently, and 2) the new sectors computed by the proposed algorithm outperform the current operational sectors in almost all of the metrics we have defined for both the average and worst-case behaviors. As a result, it distributes the workload more evenly to the sectors (air-traffic controllers) and thus improves the overall efficiency.

Appendix

Definition 1: The cut cost of a graph $\mathbb{G}$ with respect to a given partition of $\mathbb{G}$ into $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_K$, denoted as cut($\mathbb{G}$), is the total weight of all of the links that connect any pair of subgraphs:

$$\text{cut}\,(\mathbb{G}) = \sum_{i=1}^{K} \text{cut}(\mathbb{G}_i, \bar{\mathbb{G}}_i) \tag{A1}$$

where

$$\mathbb{G} = \mathbb{G}_i \cup \bar{\mathbb{G}}_i \tag{A2}$$

$$\text{cut}\,(\mathbb{G}_i, \mathbb{G}_j) \triangleq \sum_{v_r \in \mathcal{V}_i} \sum_{v_l \in \mathcal{V}_j, l \neq r} \omega_{r,l} \tag{A3}$$

$$\mathbb{G} = \cup_{i=1}^{K} \mathbb{G}_i; \qquad \mathbb{G}_i \cap \mathbb{G}_j = \emptyset; \quad \forall\ i \neq j \tag{A4}$$

Definition 2: The weight of graph $\mathbb{G}$ (or subgraph $\mathbb{G}_i$), denoted as $\omega(\mathbb{G})$ [$\omega(\mathbb{G}_i)$], is the sum of the weights of all of the links within the graph:

$$\omega(\mathbb{G}) = \mathrm{cut}(\mathbb{G}, \mathbb{G}) = \sum_{v_r \in \mathcal{V}} \sum_{v_l \in \mathcal{V}, l \neq r} \omega_{r,l} \tag{A5}$$

$$\omega(\mathbb{G}_i) = \mathrm{cut}(\mathbb{G}_i, \mathbb{G}_i) = \sum_{v_r \in \mathcal{V}_i} \sum_{v_l \in \mathcal{V}_i, l \neq r} \omega_{r,l} \tag{A6}$$

where $\omega_{r,l}$ represents the weight of link (r, l).

Definition 3: The vertex degree and graph degree matrix $D = [d_l]$ is a n by n diagonal matrix, where the diagonal element d_l $(l = 1, 2, \ldots, n)$ is the degree of vertex l computed by

$$d_l = \sum_{r=1}^{n} \omega_{l,r}$$

Definition 4: The degree of graph $\mathbb{G}$ (or subgraph $\mathbb{G}_i$), denoted as $\deg(\mathbb{G})$ [$\deg(\mathbb{G}_i)$], is defined as the sum of vertex degrees:

$$\deg(\mathbb{G}) = 2\omega(\mathbb{G}) \tag{A7}$$

$$\deg(\mathbb{G}_i) = 2\omega(\mathbb{G}_i) + \mathrm{cut}(\mathbb{G}_i) \tag{A8}$$

Definition 5: An indicator vector of a subgraph G_i, denoted as $h^{\mathbb{G}_i} = [h_l^{\mathbb{G}_i}]$, is a binary n vector:

$$h_l^{\mathbb{G}_i} = \begin{cases} 1 & v_l \in \mathcal{V}_i \\ 0 & v_l \notin \mathcal{V}_i \end{cases} \tag{A9}$$

Definition 6: An indicator matrix of a graph, denoted as H, is a binary $n \times K$ matrix whose columns are the subgraph indicator vectors:

$$H = [h^{\mathbb{G}_1}, h^{\mathbb{G}_2}, \ldots, h^{\mathbb{G}_K}] \tag{A10}$$

Definition 7: The graph Laplacian matrix L is defined by

$$L := D - W$$

where D and W are the graph degree matrix and weight matrix, respectively.

Acknowledgments

The authors would like to acknowledge that this work is supported by NASA and Mosaic Air Traffic Management (ATM), Inc. through NASA Research Announcements, subtopic 18. The authors would also want to thank Chris Brinton and Ken Leiden from Mosaic ATM, Inc. and Michael Bloem from NASA Ames Research Center for their valuable comments and support.

References

[1] "Air Traffic Control," Order JO 7110.65T, U.S. Dept. of Transportation, Feb. 2008, http://www.faa.gov/air_traffic/publications/atpubs/ATC/index.htm [retrieved 01 Oct. 2009].

[2] Swenson, H., Barhydt, R., and Landis, M., "Next Generation Air Transportation System Air Traffic Management: Airspace Project (Version 6.0)," NASA, June 2006.

[3] Arbuckle, D., Rhodes, C. D., Andrews, M., Roberts, D., Hallowell, S., Baker, D., Burleson, C., Howell, J., and Howell, J., "U.S. Vision for 2025 Air Transportation," *Journal of Air Traffic Control*, Vol. 48, No. 1, 2006, pp. 15–21.

[4] Kopardekar, P., Bilimoria, K., and Sridhar, B., "Initial Concepts for Dynamic Airspace Configuration," 7th AIAA Aviation Technology, Integration and Operations Conference, AIAA Paper 2007-7763, Sept. 2007.

[5] Delahaye, D., Alliot, J.-M., Schoenauer, M., and Farges, J.-L., "Genetic Algorithms for Partitioning Air Space," *Proceedings of the Tenth Conference on Artificial Intelligence for Applications*, IEEE Computer Society Press, Washington, DCMarch 1993, pp. 291–297.

[6] Delahaye, D., Schoenauer, M., and Alliot, J.-M., "Airspace Sectoring by Evolutionary Computation," *Proceedings of the IEEE International Conference on Evolutionary Computation*, IEEE Publ., Piscataway, NJ, May 1998, pp. 218–223.

[7] Xue, M., "Airspace Sector Redesign Based on Voronoi Diagrams," AIAA Guidance, Navigation, and Control Conference, AIAA Paper 2008-7223, Aug. 2008.

[8] Cormen, T., Leiserson, C., Rivest, R., and Stein, C., *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[9] Trandac, H., Baptiste, P., and Duong, V., "Optimized Sectorization of Airspace with Constraints," *The 5th USA/Europe Air Traffice Management Research and Development Seminar* [online proceedings], Budapest, Hungary, June 2003, http://www.atmseminar.org/past-seminars/5th-seminar-budapest-hungary-june-2003/papers/paper_071/view.

[10] Yousefi, A., and Donohue, G., "Temporal and Spatial Distribution of Airspace Complexity for Air Traffic Controller Workload-Based Sectorization," AIAA 4th Aviation Technology, Integration, and Operations Conference, AIAA Paper 2004-6455, Sept. 2004.

[11] Brinton, C., and Pledgie, S., "Airspace Partitioning Using Flight Clustering and Computational Geometry," *27th Digital Avionics Systems Conference*, Vol. 1, IEEE Publ., Piscataway, NJ, Oct. 2008, pp. 3.B.3–1–3.B.3–10.

[12] Kopardekar, P., and Magyarits, S., "Measurement and Prediction of Dynamic Density," *The 5th USA/Europe Air Traffice Management Research and Development Seminar* [online proceedings], Budapest, 2003, http://www.atmseminar.org/past-seminars/5th-seminar-budapest-hungary-june-2003/papers/paper_007/view.

[13] Basu, A., Mitchell, J., and Sabhnani, G., "Geometric Algorithms for Optimal Airspace Design and Air Traffic Controller Workload Balancing," *16th Fall Workshop on Computational and Combinatorial Geometry*, MIT Press, Cambridge, MA, Nov. 2008.

[14] Mitchell, J., Sabhnani, G., Krozel, J., Hoffman, B., and Yousefi, A., "Dynamic Airspace Configuration Management Based on Computational Geometry Techniques," AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA Paper 2008-7225, Aug. 2008.

[15] Martinez, S., Chatterji, G., Sun, D., and Bayen, A., "A Weighted-Graph Approach for Dynamic Airspace Configuration," AIAA Aviation Technology, Integration, and Operations Conference, AIAA Paper 2007–6448, Aug. 2007.

[16] Klein, A., "An Efficient Method for Airspace Analysis and Partitioning Based on Equalized Traffic Mass," *6th USA/Europe Air Traffic Management Research and Development Seminar* [online proceedings], Baltimore, MD, Sept. 2004, http://www.atmseminar.org/past-seminars/6th-seminar-baltimore-md-usa-june-2005/papers/paper_001/view.

[17] Klein, A., Rodgers, M., and Kaing, H., "Dynamic FPAS: A New Method for Dynamic Airspace Configuration," *Integrated Communications, Navigation and Surveillance Conference*, IEEE Publ., Piscataway, NJ, May 2008, pp. 1–11.

[18] Brinton, C., Leiden, K., and Hinkey, J., "Airspace Sectorization by Dynamic Density," 9th AIAA Aviation Techonology, Integration, and Operations Conference, AIAA Paper 2009-7102, Sept. 2009.

[19] Masalonis, A., Callaham, M., and Wanke, C., "Dynamic Density and Complexity Metrics for Realtime Traffic Flow Management," *The 5th USA/Europe Air Traffice Management Research and Development Seminar* [online proceedings], Budapest, Vol. 1, June 2003, http://www.atmseminar.org/past-seminars/5th-seminar-budapest-hungary-june-2003/papers/paper_086/view.

[20] Chatterji, G., Zheng, Y., and Kopardekar, P., "Analysis of Current Sectors Based on Traffic and Geometry," AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAAPaper 2008-7227, Aug. 2008.

[21] Ng, A. Y., Jordan, M., and Weiss, Y., "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems*, Vol. 14, MIT Press, Cambridge, MA, 2001.

[22] Shi, J., and Malik, J., "Normalized Cuts and Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, Aug. 2000, pp. 888–905. doi:10.1109/34.868688

[23] von Luxburg, U., "A Tutorial on Spectral Clustering," *Statistics and Computing*, Vol. 17, No. 4, Dec. 2007, pp. 395–416. doi:10.1007/s11222-007-9033-z

[24] Trandac, H., Baptiste, P., and Duong, V., "A Constraint-Programming Formulation for Dynamic Airspace Sectorization," *Proceedings of 21st Digital Avionics Systems Conference*, Vol. 1, IEEE Publ., Piscataway, NJ, 2002, pp. 1C5–1–1C5–11.