Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 6048326, 21 pages
http://dx.doi.org/10.1155/2016/6048326

Research Article

Sectorization and Configuration Transition in Airspace Design

Xiang Zou, Peng Cheng, Bang An, and Jingyan Song

Department of Automation, Tsinghua University, Beijing 100084, China

Correspondence should be addressed to Peng Cheng; chengp@tsinghua.edu.cn

Received 20 February 2016; Accepted 24 May 2016

Academic Editor: Babak Shotorban

Current airspace is sectorized according to some predefined rules that are not flexible. To facilitate utilizing the airspace more efficiently, methods to design sectors need to be promoted. In this paper, we propose an undirected graph cut-based approach that employs a memetic local search-embedded constrained evolution algorithm, NSGA-II, to generate nondominated airspace configurations. We also propose a new concave hull-based method to automatically depict sector boundaries. In addition, we also study the configuration transition problem. We define the similarity of the two different configurations and calculate their similarity with a bisection diagram and a minimum cost flow algorithm. We build a forward network to represent configuration transitions across several consecutive time periods and use multiobjective dynamic programming to determine a series of nondominated configuration links from the first period to the end. We test our approaches by simulation in high-altitude airspace controlled by Beijing Area Control Center. The results show that our sectorization method outperforms the current configuration in practice, providing a lower sector number, lower intersector flow, more balanced workload distribution among the different sectors, and no constraint violations, so that the proposed approach shows its significant potential as practical applications for dynamic airspace configuration.

1. Introduction and Literature Review

Airspace sectors are basic controlling units in Air Transportation Systems (Figure 1). They were originally designed according to some predefined rules such as historical or geographic considerations or just according to experience. Sectors have essentially remained unchanged in terms of geometric shape and the total number of sectors inside a specific airspace. However, along with rapidly increasing air transportation, fixed sectors cannot accommodate varying traffic flows anymore; several problems have arisen, such as unbalanced workload distribution across different sectors, with overload in some sectors and very sparse flow density in others, and improper sector numbers, which means too many open sectors in off-peak time periods and too few sectors during busy times or too little flight time in a single sector for some flights.

Original ideas to deal with the problem of fixed airspace structure is the "Merge and Divide" operation, meaning combining two or more adjacent sectors together when the traffic flow is low and splitting one sector into several during peak hours or choosing one airspace structure from a predefined experienced structure set [1–4]. However, this approach is not flexible enough because the boundaries of these sectors remain unchanged across different time periods. A more advanced concept, called Dynamic Airspace Configuration (DAC), was therefore proposed [5]. In DAC, both the boundaries of the sectors and the number of sectors are allowed to change according to varying traffic situations.

One key issue in DAC is the sectorization problem, that is, how to divide an airspace into several sectors. The solution of a sectorization problem is always called the airspace configuration.

To the best of our knowledge, the work by Delahaye et al. [6] may be one of the earliest studies to systematically research the sectorization problem, in which the author utilized a genetic algorithm to generate an optimal airspace configuration. Since then, many approaches have been developed. To summarize, relevant methods can be sorted into three categories [7]:

(i) Methods based on geometric computation.

(ii) Methods by cells (grids) growth (gathering) or by directly clustering trajectory points.

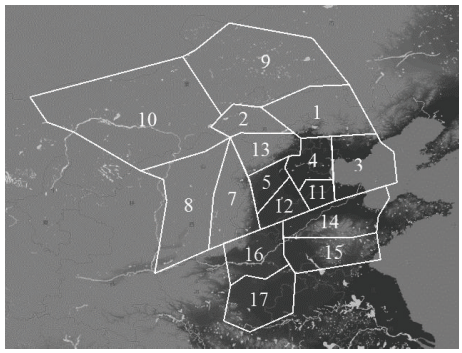


Figure 1: Configuration of Beijing Area Control.

(iii) Methods based on undirected graph cuts.

In the geometric computation category [8–13], approaches combining Voronoi diagrams with genetic algorithms were proposed in [8–10]. Tang et al. [11] used several kinds of geometric cuts such as bisection cuts and kd trees to split the airspace and compare different cutting methods.

In the cell-growth category [11, 14–20], Brinton directly clustered trajectory points to form sectors [14]. Yousefi and Donohue divided an airspace into three layers with different altitude ranges [16, 17]. Each layer was discretized into hexagonal cells, with information about the controller workload. The hexagonal cells were then gathered into sectors. Based on [16], Drew utilized a boundary-smoothing method to eliminating jagged boundary segments [18]. Klein also divided an airspace into hexagonal cells [19], but, in his approach, sectors grew up from a set of seeding cells.

The third category [20–24] is in fact another kind of clustering approach, but it is based on a weighted undirected graph and uses one subgraph to represent a sector. Li et al. constructed a weighted graph model that accurately represents the air-route network [21]. The sectorization problem was then formulated as a graph cut problem and solved by iterative spectral bisection. Martinez et al. proposed a method based on a weighted graph and a grid [22] and also utilized spectral bisection to cut the graph. Chen and Zhang proposed a spectral clustering-based approach to clustering vertices [23]. The spectral clustering solution was further refined by the ODLB algorithm and another heuristic algorithm to get better performance in terms of workload balancing. Trandac et al. proposed a method based on Constraint Programming [24].

In [25], Zelinski gave a comprehensive comparison of different approaches. The results showed completely different sector shapes according to the different approaches. The performance of these approaches was evaluated, which revealed their strengths and weaknesses. To summarize, methods using geometric computation are simple and straightforward, but they are optimally inferior because the methods used to cut the plane or space are limited. Although the second category may be the best in terms of workload balancing, it can hardly handle other objectives or constraints in sectorization problems. Approaches based on undirected graph cuts show great potential in managing multiple objectives and constraints. However, these approaches have to face two critical issues. The first one is the validity of the graph cut method, which must ensure conformance to all constraints of the problem. The second one, which is much more challenging, is how to depict exact sector boundaries based on the generated subgraphs. In this paper, we focus on high-altitude airspace in en route areas. The top and bottom covers of this kind of airspace match completely, so that horizontal and vertical partitioning operations can be executed separately [7]. Because the operations for vertical partitioning are always much simpler than the horizontal ones [7, 10], in our work, we only carry out sectorization in the horizontal plane, and we call the sectorization problem solved here the 2D en route sectorization problem (2D ERSP).

The 2D ERSP is a typical multiobjective optimization problem (MOOP) with several objectives, such as balancing the workload across different sectors and minimizing intersector flow, and several constraints, such as no overload in any sector and no reentry of any single sector for any flight.

The first main contribution of this paper is that we propose an approach that comprehensively handles multiple objectives and ensures no violations against any constraint. This approach belongs to the undirected graph cut category, and it is based on the reasonable airspace model and the Constrained NSGA-II [26] accompanied by memetic local search [27]. In addition, thanks to the evolution procedure, we do not need to predefine desired sector numbers as many previous studies have had to do.

Different from geometric computation-based and cell-growth-based approaches, undirected graph cut-based methods cannot naturally generate sector boundaries after vertex clustering. However, many studies of this type directly neglect the boundary depiction problem, only giving vertex clustering results. Among the very limited results in undirected graph cut methods that seriously consider the boundary depiction problem, Chen and Zhang used Voronoi polygons of vertices to form sector boundaries [23]. Li et al. proposed a method based on a shortest-path algorithm [21], and Trandac et al. used *constrained triangulation* [24] to form sector boundaries. These two studies both used manual selection. In fact, to the best of our knowledge, except for the methods employing Voronoi polygons, other boundary depiction methods serving undirected graph cut-based approaches generally require manual selection or manual modification.

Hence, the second main contribution of this paper is its proposal of a new concave hull-based approach to automatically depict exact sector boundaries.

In the DAC, although airspace configuration should remain fixed for a particular time period for practical reasons, such as the demand of air traffic controllers [2–4], it can and should be changed to match varying traffic situations. Generally, the change of airspace configuration should be carried out in conjunction with the change in air traffic controllers [2]. Hence, another important issue in DAC is how to avoid sharp changes in airspace configuration in order to maintain steady air traffic. This problem is called the configuration transition problem (CTP). Unfortunately, like the boundary depiction problem, the CTP is also seldom

reported in the existing literatures. To our best knowledge, only the results in [4, 28] presented several rules to minimize the workload caused by the configuration transition.

The third main contribution of this paper is that it represents a serious study of the CTP. We first define the similarity of the two configurations and calculate their similarity with a bisection diagram and a minimum cost flow algorithm [29]. Next, we build a forward network to represent configuration transitions across several consecutive time periods. Then, we take advantage of multiobjective dynamic programming to find a series of nondominated configuration links from the first period to the end.

This paper is organized as follows. In Section 2, we give a comprehensive description of the 2D ERSP, including its objectives and constraints. In Section 3, we discuss how to model the airspace network into an undirected graph and how to integrate traffic information into the model. In Section 4, approaches used to solve the 2D ERSP and to depict sector boundaries are presented. In Section 5, procedures to solve the CTP are stated in detail. The performance of the proposed approaches is tested and analyzed in Section 6. Conclusions and future works are discussed in the last section.

2. Description of the 2D ERSP

We describe the 2D ERSP by three objectives and five constraints that are commonly mentioned in the majority of the literatures working on the sectorization problem.

The three objectives are as follows.

Obj1. Balancing intrasector workload across different sectors.

Obj2. Minimizing aggregate intersector workload (coordination workload).

Obj3. Minimizing aggregate occurrences of the violations of the minimum staying time constraint.

Obj1 and *Obj2* are general objectives that are nearly mentioned in all relevant literatures. Nevertheless, *Obj1* always acts as the core objective, while *Obj2* is sometimes not really considered when solving the sectorization problem in previous studies. Besides, *Obj3* is sometimes considered as a constraint in literatures like [20]. However, when it acts as a constraint, it is always treated as "soft constraint"; that is, it may not be strictly satisfied. Therefore, we model it as an objective aiming at reducing its violations.

Along with these three objectives, the 2D ERSR has five constraints as follows.

Con1. Con1 is the connectivity constraint; that is, any sector must be connected, and no sector can consist of two or more separate parts. This is the basic constraint which can never be violated.

Con2. Con2 is the convexity constraint; that is, no flight can enter one single sector more than once. Practically, this is also a strict constraint. However, it cannot be ensured in some cases.

Con3. Con3 is the instantaneous flow constraints; that is, no sector can be overload, and the number of flights in one single sector at any moment cannot exceed a threshold. This constraint is often neglected in some literatures.

Con4. Con4 is the minimum distance between fixes and boundaries (MDFB) constraint; that is, the distance between any fix and any segment of the sector boundary should not be below a minimum threshold. Few literatures have had proposed powerful methods for *Con4*.

Con5. The sector boundary should be as compact as possible. Jagged boundary is unwanted.

3. Airspace Model

3.1. Initial Undirected Graph Construction. Basic elements in airspace are waypoints, navaids, ordinary intersections, and air routes connecting them. This natural network structure makes it very convenient to represent airspace by a graph.

We model airspace into an undirected graph $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices that a vertex stands for a waypoint, or a navaid, or an ordinary intersection, and $E = \{e_{ij} \mid e_{ij} = 1, v_i, v_j \in V\}$ is the set of edges. We set $e_{ij} = 1$, if v_i is connected with v_j by a segment of air route or if the Voronoi polygons of v_i and v_j have a common side.

It is an expansion to existing literatures considering the situation of common side of Voronoi polygons when constructing the graph, in which only air routes are treated as edges of the undirected graph. One of the major reasons of introducing these extra edges is to cover the shortcomings in the connectivity check in the existing graph cut-based methods. In Figure 2(a), we assume a regional airspace consisting of three vertices and two route segments. If we want to cluster v_a and v_b into one sector and split v_c alone into another sector, the connectivity constraint in existing methods would be violated while it is satisfied in the expanded graph (see Figure 2(b)) proposed here, because v_a and v_b are treated as connected to the common side of their Voronoi polygons. This situation is possible in practice, such as a regional hub-spoke structure in which all surrounding fixes are connected to the hub fix with air routes while no routes exist between these surrounding fixes. To relax the burden of the controller in charge of the airspace near the hub fix, one general choice is to divide the airspace into more parts so that some route-unconnected fixes may be included in one sector. This dividing action could lead connectivity violation based on existing type of graph, while it still functions well if our expanded graph is employed.

3.2. Graph Simplification and Modification. We set four rules to simplify and modify the initial undirected graph.

Rule 1. If any fix is only crossed by one air route and there is no difference between the inbound course and the outbound course, directly delete this fix. In this situation, the flow density on both sides of the fix is nearly the same so that it is not necessary to consider these two route segments separately.

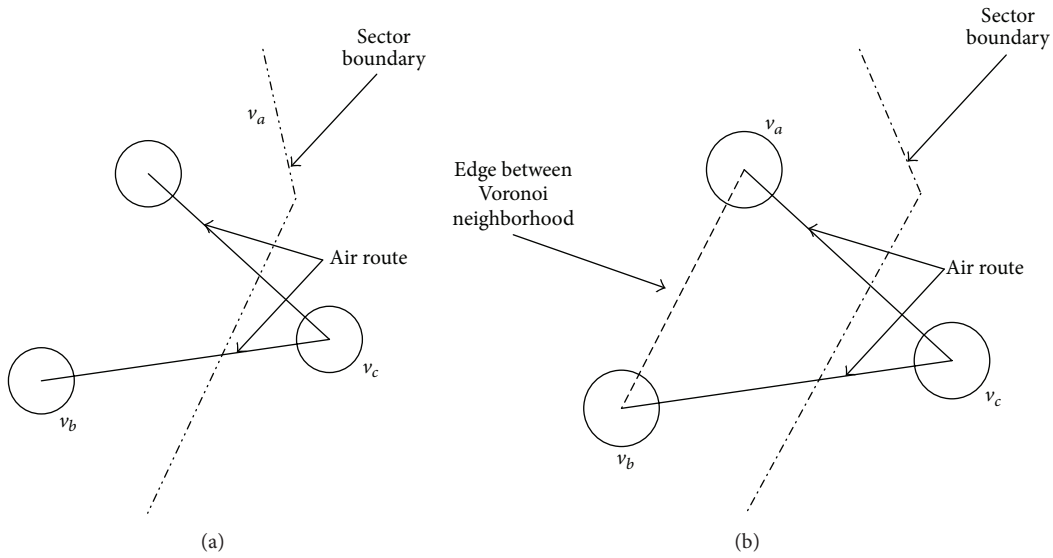


Figure 2: Edge between Voronoi neighborhood.

Rule 2. If several fixes are less than twice of the MDFB to each other, merge them as a fix union. This rule is to avoid any generated sector boundaries crossing the gap between two fixes that are too near to each other. In other words, it is to satisfy the MDFB constraint. For any fix union, we represent it with a "union-point" at the geometric center of the union and define the radius of the "union-point" as $r = maxdis + \text{MDFB}$, where $maxdis$ is the largest distance from the "union-point" to any member fixes in the fix union. Edges completely inside one fix union are deleted; edges connecting fixes belonging to different unions are represented by edges to connect the corresponding "union-points." After Rule 2, we set each "union-point" as a vertex of the modified undirected graph.

Rule 3. If any edge e_{ij} is less than the distance of MDFB from vertex v_k that is not one of its endpoints, we split e_{ij} into two segments, e_{ik} and e_{kj}.

Rule 4. If any edge e_{ij} representing Voronoi neighborhood intersects with any other edge representing air route, delete e_{ij}. This action is to avoid generating unnecessary vertices.

3.3. Traffic Information Representation. Many previous studies only employ flow density information, such as the number of flights or the number of trajectory points, to describe the traffic situation. Although the flow density is indeed the most important factor when evaluating the workload of air traffic controllers, it is oversimplified in some extent. In fact, several comprehensive metrics [30–32] evaluating airspace complexity or air traffic controllers' workload have been proposed. These metrics can be identified into four types: flow density, altitude change, speed change, and course change [2, 31, 32]. Accurately calculating each metric is very time-consuming, especially when solving the sectorization problem that needs iterative optimization. Hence, simplification or approximation must be used if we want to consider all these metrics simultaneously. There are examples such as representing several metrics with one integrated metric [2, 8], ignoring several metrics in the optimizing objectives [14], limiting the size of the target airspace and using much simpler airspace model [8], and precalculating the metrics in some basic airspace cells and directly using these values in the optimization process [15]. In the proposed approach, we introduce ten matrices to store these four types of information.

(1) Flow Density Matrix $FD_{N\times N}$. This matrix stores the number of flights flying through each edge of G, where N is the number of vertices in G. If k flights fly through e_{ij}, $\text{FD}_{ij} = k$. Besides, if $e_{ij} = 1$ and $k = 0$, we still set $\text{FD}_{ij} = \varepsilon$, where ε is a small positive value, to remain the connectivity of v_i and v_j.

(2) Symmetric Altitude Change Matrix $SAC_{N\times N}$. SAC stores the information about altitude changes. Suppose a set of trajectory points of flight f that are projected on e_{ij}, denoted as tp_{ij}^f. We calculate the standard variance of the altitudes of tp_{ij}^f, denoted as ac_{ij}^f. Define the set of flights flying through e_{ij} as F_{ij}, and we have

$$\text{SAC}_{ij} = \text{SAC}_{ji} = \sum_{f\in F_{ij}} \left|\text{tp}_{ij}^f\right| \text{ac}_{ij}^f, \tag{1}$$

where $|\text{tp}_{ij}^f|$ is the count of trajectory points in tp_{ij}^f. Other elements in SAC with $e_{ij} = 0$ or $e_{ij} = 1$, and $\text{FD}_{ij} = \varepsilon$ are all set to 0. As G is an undirected graph, SAC is symmetric.

(3) Asymmetric Altitude Change Matrix $AAC_{N\times N}$. AAC also stores the information about altitude changes. We divide trajectory points that are projected on edge e_{ij} into two parts,

$\mathrm{tp}^f_{ij,i}$ and $\mathrm{tp}^f_{ij,j}$. The trajectory points nearer to v_i than v_j are put into $\mathrm{tp}^f_{ij,i}$, while the others are put into $\mathrm{tp}^f_{ij,j}$. Then, we calculate the standard variances of these two sets, denoted as $\mathrm{ac}^f_{ij,i}$ and $\mathrm{ac}^f_{ij,j}$. Hence,

$$\begin{aligned}\mathrm{AAC}_{ij} &= \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij,i}\right| \mathrm{ac}^f_{ij,i},\\ \mathrm{AAC}_{ji} &= \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij,j}\right| \mathrm{ac}^f_{ij,j}.\end{aligned} \tag{2}$$

(4) Symmetric Speed Change Matrix $SSC_{N\times N}$. Similar to those notations in SAC, we define sc^f_{ij} and thus

$$\mathrm{SSC}_{ij} = \mathrm{SSC}_{ji} = \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij}\right| \mathrm{sc}^f_{ij}. \tag{3}$$

Other elements in SSC with $e_{ij} = 0$ or $e_{ij} = 1$ and $\mathrm{FD}_{ij} = \varepsilon$ are all set to 0.

(5) Asymmetric Speed Change Matrix $ASC_{N\times N}$. Similar to those notations in AAC,

$$\begin{aligned}\mathrm{ASC}_{ij} &= \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij,i}\right| \mathrm{ASC}^f_{ij,i},\\ \mathrm{ASC}_{ji} &= \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij,j}\right| \mathrm{ASC}^f_{ij,j}.\end{aligned} \tag{4}$$

(6) Symmetric Trajectory Point Count Matrix $STP_{N\times N}$. STP stores the count of trajectory points that are projected on each edge:

$$\mathrm{STP}_{ij} = \mathrm{STP}_{ji} = \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij}\right|. \tag{5}$$

(7) Asymmetric Trajectory Point Count Matrix $ATP_{N\times N}$. Consider

$$\begin{aligned}\mathrm{ATP}_{ij} &= \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij,i}\right|,\\ \mathrm{ATP}_{ji} &= \sum_{f\in F_{ij}} \left|\mathrm{tp}^f_{ij,j}\right|.\end{aligned} \tag{6}$$

(8) Flight Course Change Matrix $FCC_{F\times N}$. F is the number of flights considered in the sectorization problem and N is the number of vertices in G. Recall the rules in Section 3.2. In fact, any vertex in G represents a fix union. Hence, for any flight-vertex pair, such as f and v_i, we set FCC_{fi} as the aggregate course change times when flight f flies through v_i.

(9) Vertex Arrival Time Matrix $VAT_{F\times N}$. The element VAT_{fi} is defined as the earliest arriving time of flight f at any fix that belongs to v_i.

(10) Vertex Leaving Time Matrix $VLT_{F\times N}$. The element VLT_{fi} is defined as the latest leaving time of flight f from any fix that belongs to v_i.

4. Solve the 2D ERSP

4.1. Mathematical Expression. As stated before, the airspace is modeled as an undirected graph, G. Solving the 2D ERSP is to cut G into k subgraphs, $G_s(V_s, E_s)$, where $s \in S = \{1, 2, \ldots, k\}$,

$$\begin{aligned}G &= \bigcup_{s=1}^{k} G_s,\\ V &= \bigcup_{s=1}^{k} V_s,\\ V_i \cap V_j &= \emptyset, \quad i, j \in S,\ i \neq j.\end{aligned} \tag{7}$$

Each subgraph stands for one sector.

4.1.1. Objectives

(1) Balancing the Intrasector Workload. Consider

$$\min \sqrt{\frac{1}{k}\sum_{s=1}^{k}\left(\mathrm{WL}\left(G_s\right) - \overline{\mathrm{WL}}\right)^2}. \tag{8}$$

$\mathrm{WL}(G_s)$ is the intraworkload of sector s and $\overline{\mathrm{WL}}$ is the mean value of $\mathrm{WL}(G_s)$, $s \in S$. $\mathrm{WL}(G_s)$ is a weighted and scaled summation of the values of four metrics of sector s, including information about the flow density, the altitude change, the speed change, and the course change:

$$\mathrm{WL}\left(G_s\right) = w_{\mathrm{FD}}\widetilde{\mathrm{FD}}_s + w_{\mathrm{ac}}\widetilde{\mathrm{AC}}_s + w_{\mathrm{sc}}\widetilde{\mathrm{SC}}_s + w_{\mathrm{cc}}\widetilde{\mathrm{CC}}_s, \tag{9}$$

where w_{FD}, w_{ac}, w_{sc}, and w_{cc} are weights of these four parts.

We define E'_s as the set of intrasector edges of sector s, where $E'_s = \{e_{ij} \mid e_{ij} = 1, v_i, v_j \in V_s\}$ and E''_s are the set of intersector edges between s and other sectors, where $E''_s = \{e_{ij} \mid e_{ij} = 1, v_i \in V_s, v_j \notin V_s\}$; then

$$\begin{aligned}\mathrm{FD}_s &= \sum_{e_{ij}\in E'_s} \mathrm{STP}_{ij} + \sum_{e_{ij}\in E''_s} \mathrm{ATP}_{ij},\\ \mathrm{AC}_s &= \frac{\left(\sum_{e_{ij}\in E'_s} \mathrm{SAC}_{ij} + \sum_{e_{ij}\in E''_s} \mathrm{AAC}_{ij}\right)}{\mathrm{FD}_s},\\ \mathrm{SC}_s &= \frac{\left(\sum_{e_{ij}\in E''_s} \mathrm{SSC}_{ij} + \sum_{e_{ij}\in E''_s} \mathrm{ASC}_{ij}\right)}{\mathrm{FD}_s},\\ \mathrm{CC}_s &= \frac{NT_s}{NF_s},\end{aligned} \tag{10}$$

where NT_s is the total course change times of flights when flying inside sector s and NF_s is the total number of flights flying through s. Hence, CC_s stands for average course change

times of single flight in sector s. Suppose the flight set considered in the sectorization problem as F, and we have

$$NT_s = \sum_{\substack{f \in F \\ v_i \in V_s}} \mathrm{FCC}_{fi}. \tag{11}$$

Because of different ranges of the four parts in (9), and a more meaningful understanding of the summation, we normalize them to the range of $[0, 1]$. We set four standard values, FD_{sd}, AC_{sd}, SC_{sd}, and CC_{sd}, that are large enough and do not change during the optimization. Then, we define

$$\begin{aligned} \widetilde{\mathrm{FD}}_s &= \frac{\mathrm{FD}_s}{\mathrm{FD}_{sd}}, \\ \widetilde{\mathrm{AC}}_s &= \frac{\mathrm{AC}_s}{\mathrm{AC}_{sd}}, \\ \widetilde{\mathrm{SC}}_s &= \frac{\mathrm{SC}_s}{\mathrm{SC}_{sd}}, \\ \widetilde{\mathrm{CC}}_s &= \frac{\mathrm{CC}_s}{\mathrm{CC}_{sd}}. \end{aligned} \tag{12}$$

(2) Minimizing the Aggregate Coordination Workload. We use the density of intersector flow to stand for the coordination workload. This objective can be expressed by

$$\min \sum_{\substack{s \neq r \\ s,r \in S}} \mathrm{WL}_{\mathrm{co}}(G_s, G_r), \tag{13}$$

where $\mathrm{WL}_{\mathrm{co}}(G_s, G_r)$ is the intersector flow density between G_s and G_r. It is the number of flights flying across G_s and G_r:

$$\mathrm{WL}_{\mathrm{co}}(G_s, G_r) = \sum_{\substack{v_p \in V_s \\ v_q \in V_r}} \mathrm{FD}_{pq}. \tag{14}$$

(3) Minimizing the Occurrences of the Violations of the Minimum Staying Time Constraint. If any flight stays in the sector less than a minimum time threshold after it enters a sector, the minimum staying time constraint is violated. *Obj3* is to minimize the occurrences of these situations.

As we do not depict exact sector boundaries during the optimization process, it is impossible to accurately calculate the staying time of any flight. We approximate the accurate staying time as follows.

Still suppose the flight path of flight f as $fp_f = (v_{f,1}, v_{f,2}, \ldots, v_{f,k})$, $v_{f,1}, v_{f,2}, \ldots, v_{f,k} \in V$, where $v_{f,i}$ is the ith vertex along the path. Then, we calculate the staying time of flight f in sector s as follows:

$$\begin{aligned} dt_{fs} = {} & \frac{1}{2}\left(\mathrm{VAT}_{f,a} - \mathrm{VLT}_{f,a-1}\right) \\ & + \frac{1}{2}\left(\mathrm{VAT}_{f,e+1} - \mathrm{VLT}_{f,e}\right) + \left(\mathrm{VLT}_{f,e} - \mathrm{VAT}_{f,e}\right) \\ & + \sum_{\substack{v_{f,m} \in V_s \\ v_{f,m+1} \in V_s}} \left(\mathrm{VAT}_{f,m+1} - \mathrm{VAT}_{f,m}\right), \end{aligned} \tag{15}$$

where v_a and v_e are the first and the last vertices inside sector s along fp_f. The first term of (15) stands for the time spent to reach v_a after entering the sector. The second term stands for the time spent to leave the sector after leaving v_e. The third term stands for the time spent in v_e. The fourth term stands for other times spent inside sector s.

We define the set of all considered flights as F and set $mdtv_{fs} = 1$, if any flight $f \in F$ violates the minimum staying time constraint in sector s. Hence, *Obj3* can be expressed as

$$\min \sum_{f \in F} \sum_{s \in S} mdtv_{fs}. \tag{16}$$

4.1.2. Constraints

(1) Connectivity Constraint. Each subgraph representing one sector must be self-connected.

(2) Flight-Route-Based Convexity Constraint. For any flight f, $fp_f = (v_{f,1}, v_{f,2}, \ldots, v_{f,k})$, $v_{f,1}, v_{f,2}, \ldots, v_{f,k} \in V$. If $v_{f,i}, v_{f,j} \in V_s$, $i \leq j$, for $\forall v_{f,m}$, $i \leq m \leq j$, $v_{f,m} \in V_s$.

(3) Instantaneous Flow Constraint. Similar to the expressions above, we suppose v_a and v_e as the first and last vertex inside sector s along fp_f. Hence, the entry time of sector s can be denoted as $\mathrm{mid}(\mathrm{VLT}_{a-1}, \mathrm{VAT}_a)$, which is the midtime of the time leaving v_{a-1} and the time reaching v_a. Similarly, we denote the departure time of sector s as $\mathrm{mid}(\mathrm{VLT}_e, \mathrm{VAT}_{e+1})$. With the entry time and the departure time for every flight, we can calculate the number of flights in any sector at any moment. Suppose the maximum instantaneous flight number in sector s as $\mathrm{Instflow}_s$ and the instantaneous flow threshold as $\mathrm{Thre}_{\mathrm{insflow}}$, and *Con3* can be expressed as

$$\mathrm{Instflow}_{f,s} \leq \mathrm{Thre}_{\mathrm{insflow}}, \quad \forall s. \tag{17}$$

Con4 and *Con5* are not considered in the optimization phase. But they are ensured by the airspace modeling in Section 3 and the boundary depiction method in Section 4.3.

4.2. Constrained NSGA-II Based Evolution. Some objectives and constraints of the 2D ERSP cannot be written in explicit mathematical formulas [2, 24]. For example, consider the constraint that no flights can enter any single sector more than once. It is affected by the trajectory of flights and the boundary of a sector, which is too complex to express this constraint by an explicit inequality. Hence, it is impossible to construct the 2D ERSP into a typical form of mathematical programming below:

$$\begin{aligned} \text{Min} \quad & f(x) \\ & Ax \leq b \\ & x \in X. \end{aligned} \tag{18}$$

As evolution algorithms can still work even if the constraints can only be expressed by text descriptions, they are good choices to solve the 2D ERSP. The Constrained Nondominated Sorting Genetic Algorithm II (CNSGA-II)

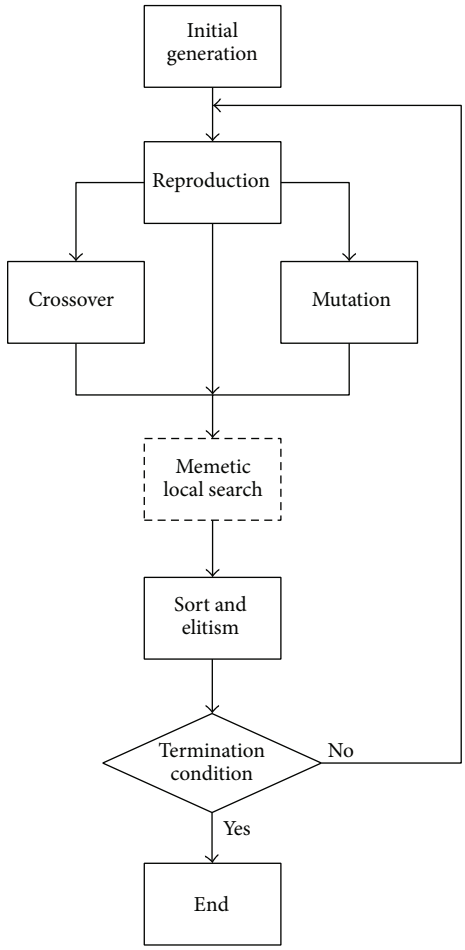


Figure 3: Diagram of the evolution algorithm.

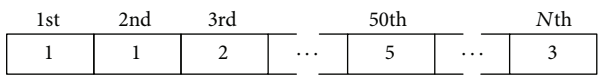


Figure 4: Chromosome.

[26] is a fast and elitist multiobjective genetic algorithm that is very appropriate for the 2D ERSP, considering its characteristics of multiobjectives and the strong capability in handling nonclassic constraints. We utilize the CNSGA-II accompanied with the memetic local search [27] to solve the 2D ERSP. The main flow diagram of the evolution algorithm is shown in Figure 3.

4.2.1. Chromosome. Every chromosome has N genes representing N vertices in graph G. The value of each gene is the index of the sector to which the related vertex belongs. For example, in Figure 4, the third vertex is assigned to sector 2 and the 50th vertex is assigned to sector 5.

We set a common largest possible index of sectors for every individual; that is, the possible maximum number of sectors is fixed. On one hand, this number is large enough so that it has no impact on the results. On the other hand, because we only set the possible maximum number, indexes of sectors in different individuals are totally independent, which means that two sectors with the same index in different individuals may refer to different sectors.

4.2.2. Reproduction. Tournament selection is used to select individuals in parent generation.

4.2.3. Crossover. Two individuals can crossover only when they have large enough hamming distance. In single crossover operation, two randomly chosen genes are exchanged. To avoid generating individuals that violates connectivity constraint as much as possible, the crossover can be granted only when at least one of the vertices (genes) is moved to an adjacent sector or stays in original sector after the crossover.

4.2.4. Mutation. In single mutation operation, one gene is randomly selected. The value of the gene is randomly changed and it is bounded by the value of the maximum sector number. However, similar to the crossover operation, only when the related vertex is changed to an adjacent sector, the mutation operation can be granted.

4.2.5. Memetic Local Search. The memetic algorithm [27] is a local search method embedded in one global search iteration of evolution algorithms that can help improving the individuals in one generation. In this paper, two heuristic polices of memetic local search are applied.

Policy A. For any sector, if $\mathrm{FD}_s > \mathrm{FD}_{\mathrm{up}}$, where $\mathrm{FD}_{\mathrm{up}}$ is an upper bound, we split the sector into two or more parts. For the subgraph G_s corresponding to sector s, we build its similarity matrix $W^{G_s}_{N_s \times N_s}$, where N_s is the number of vertices that belong to G_s. For any element $w^{G_s}_{ij}$, $1 \le i \le N_s$, $1 \le j \le N_s$, suppose the indices of its two related vertices in graph G as p and q, and then $w^{G_s}_{ij} = \mathrm{FD}_{pq}$. We add up all elements of each row of $W^{G_s}_{N_s \times N_s}$ to build a diagonal matrix D^{G_s}, where $d^{G_s}_{ii} = \sum_{j=1}^{N_s} w^{G_s}_{ij}$. We define the *Laplacian* matrix of G_s, $L^{G_s} = D^{G_s} - W^{G_s}$. Besides, we introduce three lemmas in the Spectral Graph Theory [33].

Lemma 1. *For a weighed undirected graph with nonnegative edge weights, its Laplacian matrix has at least one eigenvalue with value of 0.*

Lemma 2. *The number of isolated subgraphs of a weighed undirected graph with nonnegative edge weights equals the number of 0 eigenvalues of its Laplacian matrix.*

Lemma 3. *For a weighed undirected graph with nonnegative edge weights, denoted as G, suppose the number of 0 eigenvalues of its Laplacian matrix as k. Then, the dimension of the eigenspace of 0 eigenvalues is k. This linear subspace is spanned by such a group of base vectors: $\mathbf{1}_{s1}, \mathbf{1}_{s2}, \ldots, \mathbf{1}_{sk}$, where $\mathbf{1}_{si}$ $(1 \le i \le k)$ is the indicator vector of ith isolated subgraph, denoted as G_i. If a vertex is in G_i, the corresponding element in $\mathbf{1}_{si}$ is 1; otherwise, it is 0.*

With these three lemmas, we calculate the eigenvalues of L^{G_s} and the corresponding eigenvectors. If the multiplicity of 0 eigenvalues, k, is larger than 1, we just divide G_s into k parts according to the base vectors. If $k = 1$, we get the eigenvector corresponding to the second smallest eigenvalue, called *Fiedler Vector* [34]. Then, the vertices of G_s are clustered into two groups. One is with the positive elements of *Fiedler Vector* and the other is with the negative ones. Vertices corresponding to the elements with value of 0 are randomly assigned to one of these two groups. Hence, when $k = 1$, we split G_s into two parts.

Policy B. Policy B is operated after the division by Policy A. For any individual, if $\mathrm{FD}_s < \mathrm{FD}_{\mathrm{low}}$ for sector s, where $\mathrm{FD}_{\mathrm{low}}$ is a lower bound, we treat the flow density in this sector as too low. Thus, we merge s into one of its adjacent sectors, assumed as s'. Values of genes (vertices) belonging to s are changed to the index of s'.

To avoid loss of diversity and being trapped in regional optima, the memetic local search is operated with gradually changing intensity and frequency [35]. The frequency refers to the gap of generations between two callings of the local search, and the intensity refers to the proportion of individuals in one single generation that are applied with the local search. At the beginning of the evolution, the qualities of the individuals are still low, and, in general, we set low frequency and intensity. Along with the evolution, the searching space approaches the optimal solutions, and thus we need more accurate search. Therefore, the frequency and intensity of the local search both increase.

4.2.6. Fitness Functions. Fitness functions correspond to the expressions in formulas (8), (13), and (16).

4.2.7. Constraint Check

Connectivity. Similar to those stated in "Policy A" of the memetic local search, for each subgraph G_s (sector s), we calculate its *Laplacian* matrix L^{G_s} and check the multiplicity of 0 eigenvalues. If the number is larger than 1, we record one more time of violation of the connectivity constraint for current individual.

Flight-Route-Based-Convexity. Still suppose the flight path of flight f as $(v_{f,1}, v_{f,2}, \ldots, v_{f,k})$. If $v_{f,p} \in G_s$ and $v_{f,p+1} \notin G_s$, we check if $v_{f,q} \in G_s$, $q > p + 1$. Once the condition $v_{f,q} \in G_s$, $q > p + 1$ is met, we record one more time of the violation of the convexity constraint for current individual.

Instantaneous Flow Constraint. We check the validation of inequality (17). If inequality (17) is unsatisfied, we record one more time of the violation of the instantaneous flow constraint for current individual.

4.2.8. Sort and Elitism. In the CNSGA-II, the elitism operation is executed for every generation. The dominating relationship between two individuals (solutions), i and j, is defined below:

(1) $i \prec j$, if i is feasible (no constraint violation) while j is not.

(2) When solutions i and j are both feasible, $i \prec j$ if the values of all fitness functions of i are not more than that of j and at least one of such values is smaller than that of j.

(3) When solutions i and j are both infeasible, we compare their times of constraint violations in the following order of decreasing priority: connectivity > convexity > instantaneous flow. That is, $i \prec j$, if the connectivity constraint violating times of solution i is less than that of solution j. Or, if the connectivity constraint violating times are equal, we continue to compare the violating times of the convexity constraint and so on.

(4) When solution i and j are both infeasible and they are equivalent in terms of the violating times of all three constraints, $i \prec j$ if the values of all fitness functions of i are not more than that of j and at least one such value is smaller than that of j.

4.2.9. Initial Generation. Most previous studies using evolutionary algorithms always concern about how to refine the "genetic operations" (such as the crossover or mutation rules) or try to propose heuristics to improve the evolution procedure. However, for those problems that the gross optima or a boundary of the gross optima cannot be preliminarily determined (such as the 2D ERSP), no matter how excellent a heuristic or an improved operation seems to be, we are unable to determine whether the gross optima can be found with the help of these heuristics. In these problems, we cannot guarantee reaching the gross optima from any type of initial generations. Hence, we have to realize that different initial generations have different impact on final solutions. However, discussions about the initial generation of evolution algorithms and its impact to the final solutions are absent in relevant literatures.

In this paper, we propose four types of initial generations and compare them with the corresponding nondominated solution sets produced by the CNSGA-II.

The first type is the most general choice that just randomly produces the initial generation. In following text, we call this case *Randomly Initialization*, or RI.

The second type takes advantage of the current configuration in practice. We represent the current configuration with a chromosome and randomly mutate half of the genes several times to produce the initial generation. In following text, we call this case *Current Initialization*, or CI.

Thirdly, we employ a method based on the normalized spectral clustering algorithm (NSCA) [36]. The NSCA is a data clustering algorithm that shows its talent in the graph N-cut problem if each data point is represented by one vertex in a weighted graph and the weights of edges stand for similarities between two data points [37]. We choose the NSCA because of two objectives of the 2D ERSP, balancing intrasector workload and minimizing gross intersector workload, which are very similar to the goals of the undirected graph N-cut

(1) Input D_G, W_G and desired cluster number, k.
(2) Prune D_G and W_G by deleting rows and columns where $d_{ii}^G = 0$. Define the pruned matrices as PD_G and PW_G. Define the set of deleted vertices as VD where $|VD| = r$ and the set of remained vertices as RD. Define the dimension of PD_G and PW_G as $PN = N - r$.
(3) Calculate the *Laplacian* matrix, $PL_G = PD_G - PW_G$.
(4) Compute the normalized *Laplacian* matrix, $PL_{\text{nor}} = I_{PN\times PN} - PD_G^{-1/2} PW_G PD_G^{-1/2}$.
(5) Compute the eigenvector corresponding to the smallest k eigenvalues of PL_{nor}.
(6) Construct a matrix $E_{PN\times k}$, each of its column corresponds to the eigenvector of one of the smallest k eigenvalue.
(7) Normalize every row of $E_{PN\times k}$, meaning normalizing $E_{PN\times k}$ into $\widetilde{E}_{PN\times k}$, $\widetilde{E}_i = E_i/(\sum_k e_{ij}^2)^{1/2}$, where $\widetilde{E}_i$ and E_i stand for the ith row of $\widetilde{E}$ and E respectively, e_{ij} is the element at the ith row and jth column of E.
(8) For each row of $\widetilde{E}$, set a corresponding vector y_i $(i = 1, 2, \ldots, PN)$.
(9) Cluster vectors $y_1, y_2, \ldots, y_{PN}$ into k clusters using *k-means* cluster. The index of the cluster for each row is the index of the sector for the ith vertex of G.
(10) For any vertex in VD, assign it to the sector to which its nearest vertex in RD belongs.

Procedure 1: Procedures of the method based on the NSCA.

problem. We first change the values of elements in matrix FD with value of ε to 0 and call the modified density matrix FD$'$. Then, FD$'$ is defined as the similarity matrix of G; that is, W_G = FD$'$ and we construct a diagonal matrix D_G, where $d_{ii}^G = \sum_{j=1}^N w_{ij}^G$. Following steps of the method based on the NSCA are listed in Procedure 1. Note that d_{ii}^G may equal to 0 if no aircraft flies through any routes connecting to vertex v_i. This may lead singularity of D_G and cause an error in the 4th step in Procedure 1. Hence, we must prune W_G and D_G firstly, deleting rows and columns where $d_{ii}^G = 0$. The method in Procedure 1 outputs one configuration of the airspace. We also use a chromosome to represent this solution and produce the initial generation by randomly mutating half genes of this individual several times. In this method, we need to set desired cluster number for the NSCA. However, this number is not the final counts of sectors because of possible change in the following evolution procedure. Hence, we directly set it as the count of sectors in current configuration. In following text, we call this case *NSCA Initialization*, or NI.

In the fourth type, half individuals in the initial generation are produced by CI and the other half is produced by NI. In following text, we call this case *Mix Initialization*, or MI.

4.3. Sector Boundary Depiction. As stated in Section 1, the boundary depiction is much more challenging in graph cut-based approaches than that in geometric based and "cell gathering" based approaches. The methods directly using Voronoi polygons of vertices to form the boundaries are simple and straightforward. But they may not be very suitable in the sectorization problem for two reasons. Firstly, the shape of sectors is not strictly convex so that Voronoi polygons may not fit the outline of concave vertex cluster very well for its characteristic of convexity. Secondly, the MDFB constraint cannot be ensured by Voronoi polygons.

We propose a concave hull-based method to depict the sector boundaries in this section. We firstly construct the concave hull for each vertex cluster (representing one sector) in Procedure 2 with necessary explanations below.

4.3.1. Concave Hull Generation

Input Data. Input information of the concave hull generation method is in three categories.

The first is the cut of G:

$$G = \bigcup_{s=1}^{k} G_s,$$
$$V = \bigcup_{s=1}^{k} V_s. \tag{19}$$

The second is the set of midpoints of edges connecting two neighboring sectors, $MP = \{mp_{i,j} \mid e_{ij} = 1, v_i \in V_{s_m}, v_j \in V_{s_n}, s_m \neq s_n\}$. For any sector s, define the set of midpoints of edges connecting it and other sectors as $MP_s = \{mp_{i,j} \mid e_{ij} = 1, v_i \in V_s, v_j \notin V_s\}$. Hence, we get that $MP = \bigcup_{s\in S} MP_s$ and $MP_{s1} \cap MP_{s2} \neq \emptyset$ if $s1$ and $s2$ are neighboring sectors.

The third is a set of points located on the outerboundary of the target airspace. We suppose points on the outerboundary of the target airspace are distributed clockwise for the purpose of unification and define the point set as $OP = \{op_i \mid 1 \leq i \leq n\}$, $OP = \bigcup_s OP_s$, $s \in S$, and $OP_s = \emptyset$ if s is not next to the outerboundary.

Obtain BMP_s. For any sector s (subgraph G_s) adjacent to the outerboundary, we find out one or more couples of points in MP_s, and denote the subset of MP_s as BMP_s. For any point $mp_{i,j} \in BMP_s$ with corresponding edge e_{ij}, the following conditions must be satisfied:

(i) Both endpoints of e_{ij}, denoted as v_i and v_j, are next to the outerboundary.

(1) Initialize, input $G = \bigcup_{s\in S} G_s$, $V = \bigcup_{s\in S} V_s$, $MP = \bigcup_{s\in S} MP_s$, and $OP = \bigcup_s OP_s$
(2) **For** $s \in S$
(3) Set $B_s = \emptyset$
(4) **If** s is adjacent to the outer-boundary of the target airspace
(5) Obtain BMP_s (See explain in Section 4.3.1), and set $NBM_s = \emptyset$
(6) **For** each point $pt \in BMP_s$
(7) Find point pto, that $pto \in OP$, and pto is nearest to pt
(8) $NBM_s = NBM_s \cup pto$
(9) **End**
(10) Obtain OP_s (See explain in Section 4.3.1)
(11) $B_s = OP_s \cup MP_s$
(12) **End**
(13) **Else**
(14) $B_s = MP_s$
(15) **End**
(16) Construct $Concave(B_s)$ using α-shape algorithm (See explain in Section 4.3.1)
(17) **End**

Procedure 2: Procedures to construct $Concave(B_s)$.

(ii) The Voronoi polygons of v_i and v_j have one common side and this side intersects with the outerboundary.

Note that, as the shape of any sector is not strictly convex, there may be more than one couple of points in BMP_s. In Figure 5(a), we assume that $V2$ and $V3$ belong to sector s and $V1$ and $V4$ belong to the other two different sectors, respectively. In this situation, BMP_s is made up of $MP1$ and $MP2$. In Figure 5(b), we assume that sector s consists of $V2$, $V3$, and $V4$, which surround $V1$. The shape of s is concave, but s still conforms to the convexity constraint because no air routes connect $V3V1$ or $V4V1$. In this situation, $MP1$, $MP2$, $MP3$, and $MP4$ are all elements of BMP_s. BMP_s with more than two couples of points is also possible which is not illustrated here. Note that $BMP_{s1} \cap BMP_{s2} \neq \emptyset$, if $s1$ and $s2$ are neighboring sectors.

Obtain OP_s. For any sector s adjacent to the outerboundary, sort NBM_s clockwise (as points in NBM_s all belong to OP, sorting clockwise means sorting with increasing index in OP) as $NBM_s = \{op_1^s, op_2^s, \ldots, op_{2k-1}^s, op_{2k}^s\}$, where k is the number of couples n BMP_s. Then, we define

$$OP_s = \bigcup_{i=1}^{k} \{op_{2i-1}^s, \ldots, op_{2i}^s\}. \tag{20}$$

We are aware that two neighboring points in NBM_s may not be consecutive in OP. We define

$$\begin{aligned} &\{op_{2i-1}^s, \ldots, op_{2i}^s\} \\ &\quad = \{op_{\text{ind1}}, op_{\text{ind1}+1}, \ldots, op_{\text{ind2}-1}, op_{\text{ind2}}\}, \end{aligned} \tag{21}$$

where ind1 is the index of op_{2i-1}^s in OP and ind2 is the index of op_{2i}^s in OP.

Construct Concave Hull of B^s. We use the concave hull of B^s, denoted as $Concave(B^s)$, to fit the outline of the vertex cluster of sector s. α-shape algorithm [38] is applied here to generate the concave hull. The basic idea of α-shape algorithm is to iteratively search α-extreme points and α-neighbors and to connect the found α-neighbors sequentially to generate α-hull. As large amount of extra definitions and notations must be introduced to formulate α-shape algorithm clearly, we do not present the details here.

4.3.2. Concave Hull Amendment. $Concave(B^s)$ is a promising candidate to be the boundary of sector s because all vertices of G_s are inside it and it fits the shape of s very well. However, two more considerations make it necessary to recheck $Concave(B^s)$. The first is the compactness of the boundary; that is, fewer segments are preferred on the boundary. The second is about the MDFB constraint; that is, the distance between some segments of $Concave(B^s)$ and some vertices of G may be less than the MDFB. Hence, we propose the Concave Hull Amendment Method (*CHAM*) to deal with these two considerations (Procedure 3).

There are two ideas in the *CHAM*. The first is to replace the polygonal line connecting several vertices on the concave hull with one line segment from the first vertex to the last one. The second idea is to avoid obstacle circles centering at the vertices of G with radius of that vertex (refer to Section 3.1 for the definition of the radius of a vertex). We call modified concave hulls after the *CHAM* boundary polygons and present two definitions about the concave hull and the boundary polygon first.

Definition 4 (path(s) on the concave hull). One scans the vertices of $Concave(B^s)$ clockwise. If s is adjacent to the outerboundary of the target airspace, whenever we encounter a sequence of vertices $V_1^p - V_2^p - \cdots - V_{r-1}^p, V_r^p$, where V_1^p, $V_r^p \in OP$ or others belonging to MP, we define such a sequence as a path on the concave hull. If s is not adjacent to the outerboundary, that is, it is surrounded by other sectors, we simply define a path on $Concave(B^s)$ as a clockwise closed walk of all vertices of $Concave(B^s)$.

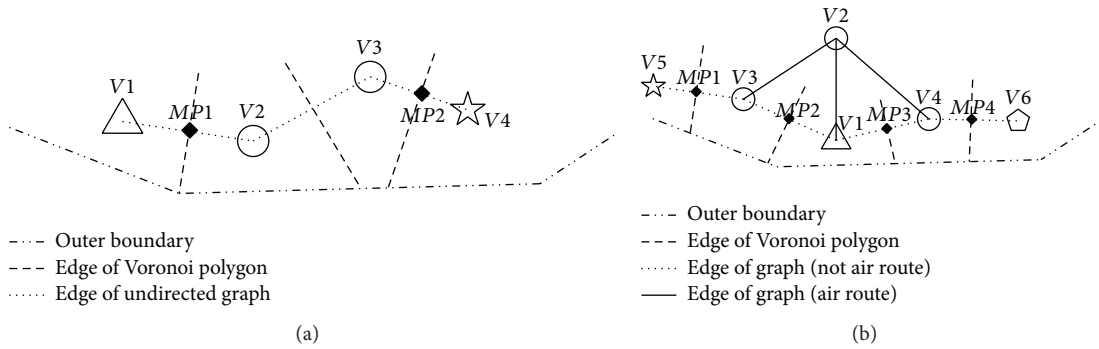


Figure 5: Midpoints adjacent to the outerboundary.

(1) **For** $s \in S$
(2) Get path(s) on $Concave(B_s)$, set as $P_1^s, P_2^s, \ldots, P_{k_s}^s$ (See Definition 4 in Section 4.3.2)
(3) **For** each path P_i^s, $1 \le i \le k_s$
(4) Define vertices along P_i^s as $V_1^p, V_2^p, \ldots, V_{N_i}^p$ and corresponding $BI_1^p, BI_2^p, \ldots, BI_{N_i}^p$ (note that $BI_j^p = V_j^p$ when corresponding cross-sector edge has not been designed)
(5) Set $j = 1$, gap = 1.
(6) **While** $j < N_i$
(7) **If** cross-sector edges referred by BI_j^p and BI_{j+1}^p are both designed
(8) **If** line segment $BI_j^p BI_{j+1}^p$ is infeasible (See Definition 5 in Section 4.3.2)
(9) Execute the *OABD* to $BI_j^p BI_{j+1}^p$ (See Explain in Section 4.3.2)
(10) **End**
(11) $j = j + 1$, go to line (6)
(12) **End**
(13) **While** line segment $BI_j^p BI_{j+\text{gap}}^p$ is feasible (See Definition 5 in Section 4.3.2)
(14) gap = gap + 1
(15) **If** cross-sector edges referred by $BI_{j+\text{gap}-1}^p$ and $BI_{j+\text{gap}}^p$ are both designed
(16) **break**
(17) **End**
(18) **End**
(19) **If** gap = 1
(20) Execute the *OABD* to $BI_j^p BI_{j+1}^p$ (See Explain in Section 4.3.2).
(21) Set the cross-sector edge referred by BI_{j+1}^p as designed, remain $BI_{j+1}^p = V_{j+1}^p$
(22) $j = j + 1$, go to line (6)
(23) **End**
(24) **Else**
(25) gap = gap − 1
(26) Draw line segment $BI_j^p BI_{j+\text{gap}}^p$,
(27) Set cross-sector edges referred by $BI_{j+1}^p, BI_{j+2}^p, \ldots, BI_{j+\text{gap}}^p$ as designed
(28) Set $BI_{j+1}^p, BI_{j+2}^p, \ldots, BI_{j+\text{gap}}^p$ as the intersections of related cross-sector edges and line segment $BI_j^p BI_{j+\text{gap}}^p$
(29) $j = j + \text{gap}$, reset gap = 1, go to line (6)
(30) **End**
(31) **End**
(32) **End**
(33) **End**

Procedure 3: Procedures of the *CHAM*.

Definition 5. A line segment connecting any two vertices on the boundary polygon of one sector is defined as feasible if it does not intersect with any inner-edges of any sector and it does not go through any obstacle circles centering at the vertices of G.

Next, we present another definition to avoid redepiction of common boundary between two neighboring sectors.

Definition 6. In the *CHAM*, once a segment on the boundary polygon is depicted, set all cross-sector edges that intersect

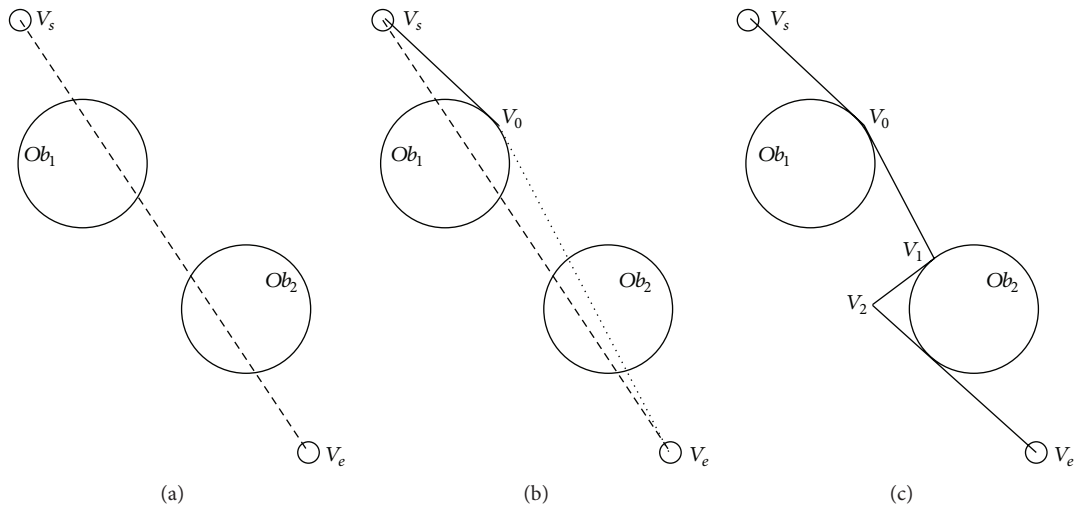


Figure 6: Procedures of the *OABD*.

with this segment as designed. Define the intersection of this segment and a cross-sector edge, e_{ij}, as the boundary crossing point of e_{ij}, denoted as BI_{ij}. When one cross-sector edge has not been designed, its default boundary crossing point is set as its midpoint, that is, the intersection of this edge and the concave hull. To ensure unification, when one endpoint of a depicted segment lies on the outerboundary of the target airspace, we also define a virtual cross-sector edge corresponding to this endpoint as designed and set the boundary crossing point as the endpoint itself.

We also give two properties about the concave hull.

Property 1. For any sector s, line segments on $Concave(B^s)$ do not intersect with any inner-edges of any sector. This property is obvious from the definition and the generating process of α-shape [38].

Property 2. The distance between any vertex on the concave hull, representing the midpoint of a cross-sector edge, and any vertex of G is at least MDFB.

Proof. For any vertex on the concave hull, if the corresponding cross-sector edge represents a real air-route segment, the edge is at least MDFB away from any vertex of G according to Rule 3 in Section 3.2 except its two endpoints. In addition, as all edges of G are at least twice as long as MDFB, their midpoints are at least MDFB away from the two endpoints. For one vertex on the concave hull that is the midpoint of a sector-cross edge only standing for Voronoi neighboring relationship, it locates on the Voronoi polygons of both endpoints of this edge, which means that the nearest vertices of G to this midpoint are just the two endpoints of the edge. Considering the possible distance between this midpoint and the two endpoints is at least MDFB, if any other vertex of G is less than MDFB from this midpoint, a contradictory occurs. □

These two properties show two pieces of important information when we construct the boundary polygons:

(1) For any two consecutive vertices on the concave hull of a sector, if we can find a line segment or a polygonal line that does not go through any obstacle circles, this line segment or polygonal line can be set as boundary segment(s) of that sector.

(2) The boundary polygons after the *CHAM* can cross any vertices on the concave hulls.

We show the procedures of the *CHAM* in Table 1 with necessary explanations below.

Obstacle Avoidance Boundary Design (OABD). We illustrate the *OABD* with Figure 6. Firstly, we assume two consecutive vertices on the concave hull as V_s and V_e. V_sV_e is assumed to intrude one or more obstacle circles (Figure 6(a)). The *OABD* examines the first obstacle (such as Ob_1 in Figure 6) intruded by V_sV_e. Then, we draw two tangents from V_s and V_e to the obstacle circle, respectively. The intersection of these two tangents is denoted as V_0. Thus, the boundary between V_s and V_e is now $V_sV_0V_e$ (Figure 6(b)). Next, we examine if $V_sV_0V_e$ still intrudes any other obstacle (such as Ob_2 in Figure 6). If it does, we truncate V_0V_e at the boundary of the second obstacle (V_1 in Figure 6(c)). Then, we draw two tangents of Ob_2 originating from V_1 and V_e, respectively, and get their intersection (V_2 in Figure 6(c)). The boundary between V_s and V_e is now changed to $V_sV_0V_1V_2V_e$. The *OABD* continues until the curve $V_sV_0V_e$ (1 iteration) or $V_sV_0V_1 \cdots V_{k+1}V_e$ (k iterations, $k > 1$) is free of all obstacles (Figure 6(c)). If too many iterations are needed in the *OABD*, the generated sector boundaries may be jagged, which is prohibited. Throughout our simulation, such situations do not appear. Relevant discussions will be in Section 6.5.

In the *OABD*, there are always two intersections when we draw tangents, such as I_1 and I_2, in Figure 7. As I_1 and I_2 always locate symmetrically on opposite sides of the obstacle circle, one of the generated polygonal lines definitely intersects with inner-edges of sectors while the other does not. The one without intersections with inner-edges is our choice (such as $V_sI_2V_e$ in Figure 7).

Table 1: Results of six sectorization methods.

Time period	Sectorization method	*Obj1*	*Obj2*	*Obj3*	*Con1*	*Con2*	*Con3*	Nos	Density gap	RT (s)
12:00–14:00	Current in practice	0.8145	674	238	0	1	0	16	4218	—
	Only NSCA	1.0518	471	133	0	6	1	16	6027	—
	RI	0.3158	404	84	1	0	0	9	2521	934
	CI	0.3124	360	59	0	0	0	9	2232	940
	NI	0.3404	375	58	0	0	0	8	2356	942
	MI	0.4329	386	68	0	0	0	8	2512	964
16:00–18:00	Current in practice	1.1466	707	234	0	4	2	16	6705	—
	Only NSCA	1.6230	428	86	0	0	3	16	6732	—
	RI	0.3148	525	106	1	0	0	10	2409	926
	CI	0.4508	608	130	0	0	0	10	3007	937
	NI	0.4259	499	104	0	0	0	10	2521	924
	MI	0.4502	511	91	0	0	0	10	2619	940
20:00–22:00	Current in practice	1.3666	710	247	0	3	2	16	7198	—
	Only NSCA	2.0712	403	72	0	1	5	16	9150	—
	RI	0.4514	684	206	2	0	0	12	2375	936
	CI	0.3487	719	218	0	0	0	12	1902	934
	NI	0.4142	592	134	0	0	0	12	2258	930
	MI	0.3993	598	136	0	0	0	12	2017	915
2:00–24:00	Current in practice	1.3850	427	166	0	1	2	16	7604	—
	Only NSCA	1.6230	428	86	1	2	3	16	9058	—
	RI	0.4318	468	205	2	0	0	12	2635	944
	CI	0.4100	335	79	0	0	0	12	2239	937
	NI	0.3678	350	105	0	0	0	12	2159	924
	MI	0.2982	351	118	0	0	0	12	2073	930

4.3.3. Vacuum Space Handle. After the *CHAM*, there still remains several vacuum spaces at the intersecting areas of three or more sectors, such as quadrilateral *ABCD* in Figure 8(a). These vacuum spaces contain no vertices or edges and are not assigned to any sector. In this section, we propose a heuristic method to assign vacuum spaces. Firstly, we represent a vacuum space with a polygon, calling the vacuum polygon. The vacuum polygons can be sorted into four categories according to types of their sides:

(I) Triangles with no side from the *OABD*.

(II) Polygons with more than three sides and no side that is from the *OABD*.

(III) Quadrangles with two sides from the *OABD*.

(IV) Other types.

For vacuum polygons in Category (I), we propose two choices, one is directly erasing the opposite side of the biggest inner angle (Figure 9(a)), and the other is to find the "most vertical" midline and erase those two sides on which the midpoint does not locate (Figure 9(b)).

For vacuum polygons in Category (II), we first give the following property.

Property 3. The vacuum polygon in Category (II) is convex.

Proof. Suppose there is a concave vacuum polygon in Category (II) (such as the one in Figure 8(b)). We consider the vertex causing the concavity (point *C* in Figure 8(b)). As this vertex is on a cross-sector edge, one of the endpoints of this cross-sector edge locates inside the vacuum polygon because of the concavity, obviously causing a contradictory against the definition of the vacuum space. In another situation, it locates outside the polygon while the cross-sector edge must intersect with one extra side of the polygon. However, neither of the two endpoints of this extra side is the vertex that causes the concavity; that is, a cross-sector edge between two sectors intersects with the boundary of an irrelevant sector. It is also impossible. Hence, we show that the vacuum polygons in Category (II) must be convex. □

With Property 3, we can calculate the geometric center of a polygon in Category (II), which must locate inside the polygon because of its convexity (point *O* in Figure 8(b)). Then, we connect the center to every vertices of the polygon and erase original sides of the polygon (Figure 8(c)).

For polygons in Category (III), which stand for an intersecting area of three sectors, we find the vertex produced by the OABD (point *I* in Figure 9(c)), draw the diagonal from it, and erase those two opposite sides of the vertex produced by the OABD.

For polygons in Category (IV), situation of their sides may be very complex. We directly assign the entire vacuum space to one sector, whose boundary segments are sides of the polygon after *OABD*.

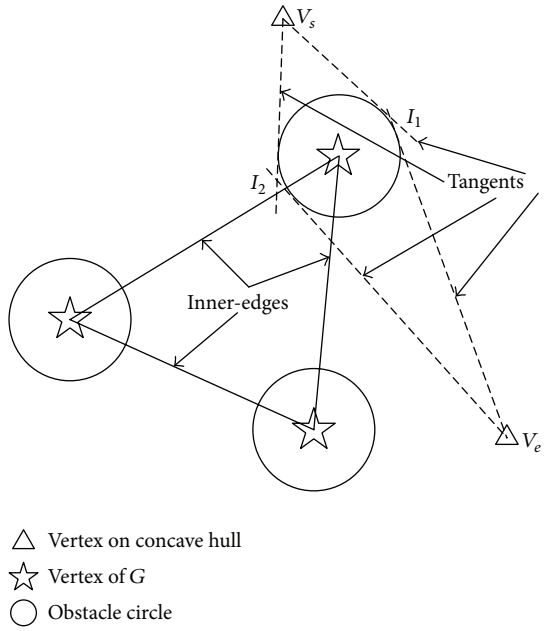


Figure 7: Selection when drawing tangents.

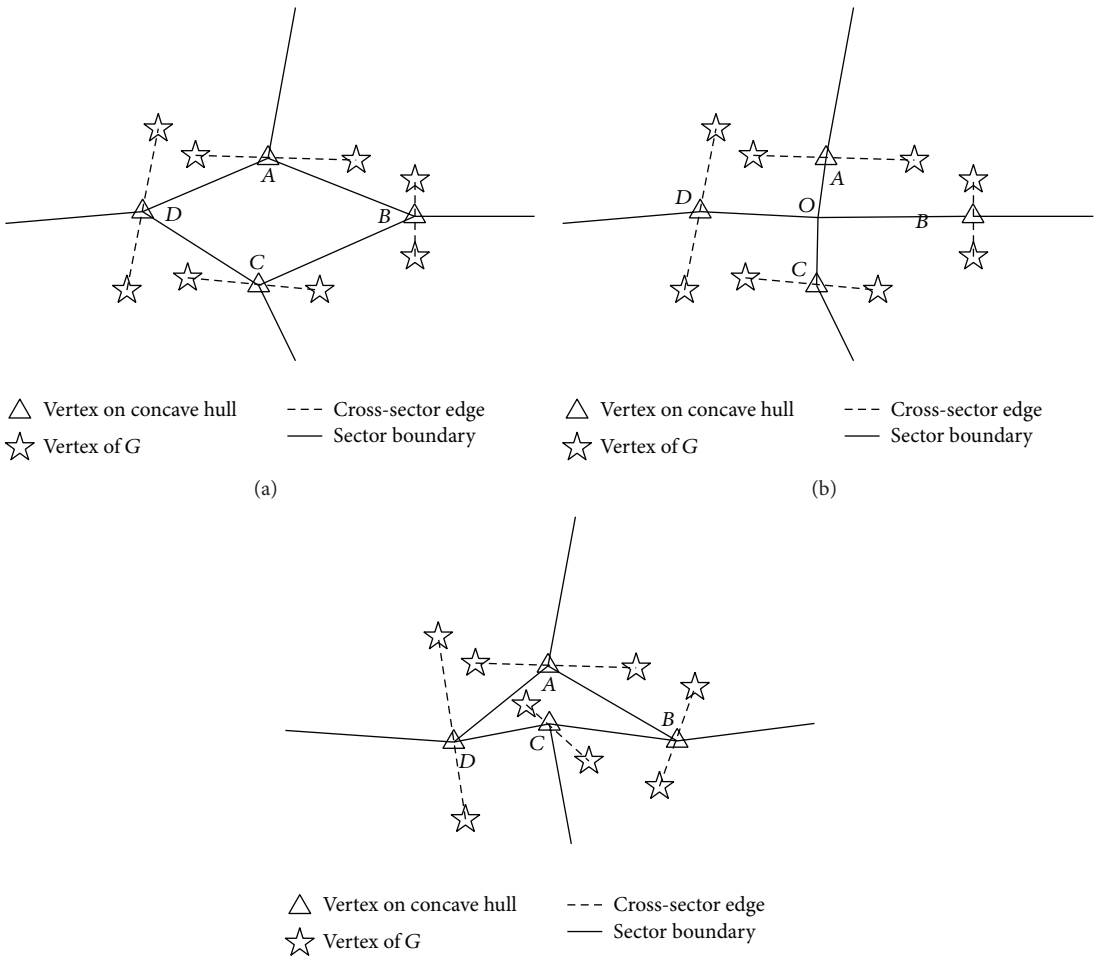


Figure 8: Vacuum polygon handle, Category (II).

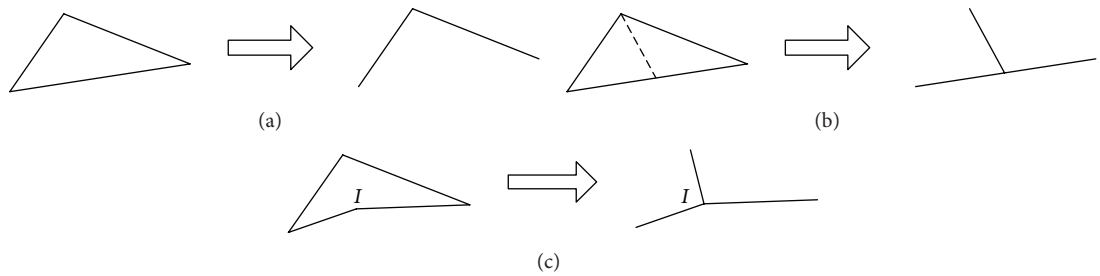


Figure 9: Vacuum polygon handle, Categories (I) and (III).

5. Configuration Transition

The methods presented from Sections 2–4 are for the sectorization problem in single time period. This section focuses on solving the CTP. First of all, we present the definition of the similarity of two different configurations followed by its calculation method.

5.1. Similarity of Two Configurations. We still use graph-based notations to denote configurations. We still model the whole target airspace as G, and define $C1$ and $C2$ as two different configurations:

$$\begin{aligned} C1:\quad & G = \bigcup_{s=1}^{k_1} G_s^1, \\ & V = \bigcup_{s=1}^{k_1} V_s^1, \\ C2:\quad & G = \bigcup_{s=1}^{k_2} G_s^2, \\ & V = \bigcup_{s=1}^{k_2} V_s^2. \end{aligned} \tag{22}$$

We construct a bisection diagram (Figure 10(a)), where nodes on each side stand for the sectors in each configuration. The weight of an edge connecting two nodes on opposite sides is the similarity of two sectors, which is defined as the number of same vertices of these two sectors:

$$\begin{aligned} \mathrm{Sim}\left(G_{s1}^1, G_{s2}^2\right) &= \left\| V_{s1}^1 \cap V_{s2}^2 \right\| \\ & s1 \in \{1, 2, \ldots, k_1\},\ s2 \in \{1, 2, \ldots, k_2\}. \end{aligned} \tag{23}$$

To calculate the similarity of $C1$ and $C2$, we need to get the maximum match of the bisection diagram, which ensures a monoprojection for every node on the side with less nodes while the aggregate weights of selected edges are maximized. Without loss of generality, we suppose $k_1 \le k_2$ (it is similar when $k_2 \le k_1$), the similarity of $C1$ and $C2$ is defined as:

$$\begin{aligned} \mathrm{Sim}\,(C1, C2) = \max & \sum_{s1=1}^{k_1} \sum_{s2=1}^{k_2} x_{s2}^{s1} \omega_{s2}^{s1}, \\ & \sum_{s2=1}^{k_2} x_{s2}^{s1} = 1, \quad \forall s1 \in \{1, 2, \ldots, k_1\}, \\ & \sum_{s1=1}^{k_1} x_{s2}^{s1} \le 1, \quad \forall s2 \in \{1, 2, \ldots, k_2\}, \\ & x_{s2}^{s1} \in \{0, 1\}, \end{aligned} \tag{24}$$

where $\omega_{s2}^{s1} = \mathrm{Sim}(G_{s1}^1, G_{s2}^2)$.

This is a typical 0-1 programming problem, which can be solved with network flow algorithms. We modify the bisection diagram (Figure 10(b)), adding one virtual source node to the side with less nodes and connect the source node with each node on the same side. Besides, we also add a virtual sink node to the side with more nodes, connecting the sink node with each node on the same side. The weights on all newly added edges are 0 so that they have no impact on the result. The weights on original edges are also changed. For an edge connecting G_{s1}^1 and G_{s2}^2, its weight is changed to $M - \mathrm{Sim}(G_{s1}^1, G_{s2}^2)$ from $\mathrm{Sim}(G_{s1}^1, G_{s2}^2)$, where M is a constant. So the objective function is changed to the form of $k_1 M - f(x)$. Because k_1 is known, original maximization problem in (24) is now transformed to a minimization problem. The supply of the source node and the demand of the sink node are both equal to k_1 and the supply/demand at other nodes are all 0. Finally, the capacities on all edges are set as 1. Based on these modifications, we build a typical network that applies to any generic minimum cost flow algorithms [29]. We do not list the procedures of the minimum cost flow algorithms in detail. Readers can easily find them in [29] or any other textbooks on network flow or Discrete Optimization.

5.2. Nondominated Configuration Links. If we want to make a configuration plan for next several time periods, we should consider the quality of selected configurations in terms of the values in *Obj1* to *Obj3* and the similarities between configurations of consecutive time periods. Hence, finding an optimal configuration plan is another multiobjective problem.

We build a forward network to represent the CTP (Figure 10(c)). Each column of nodes represents one of several nondominated configurations in one period. A node in the network is denoted as C_t^i, $1 \le t \le T$, where T is the number of time periods and $1 \le i \le k_t$, where k_t is the

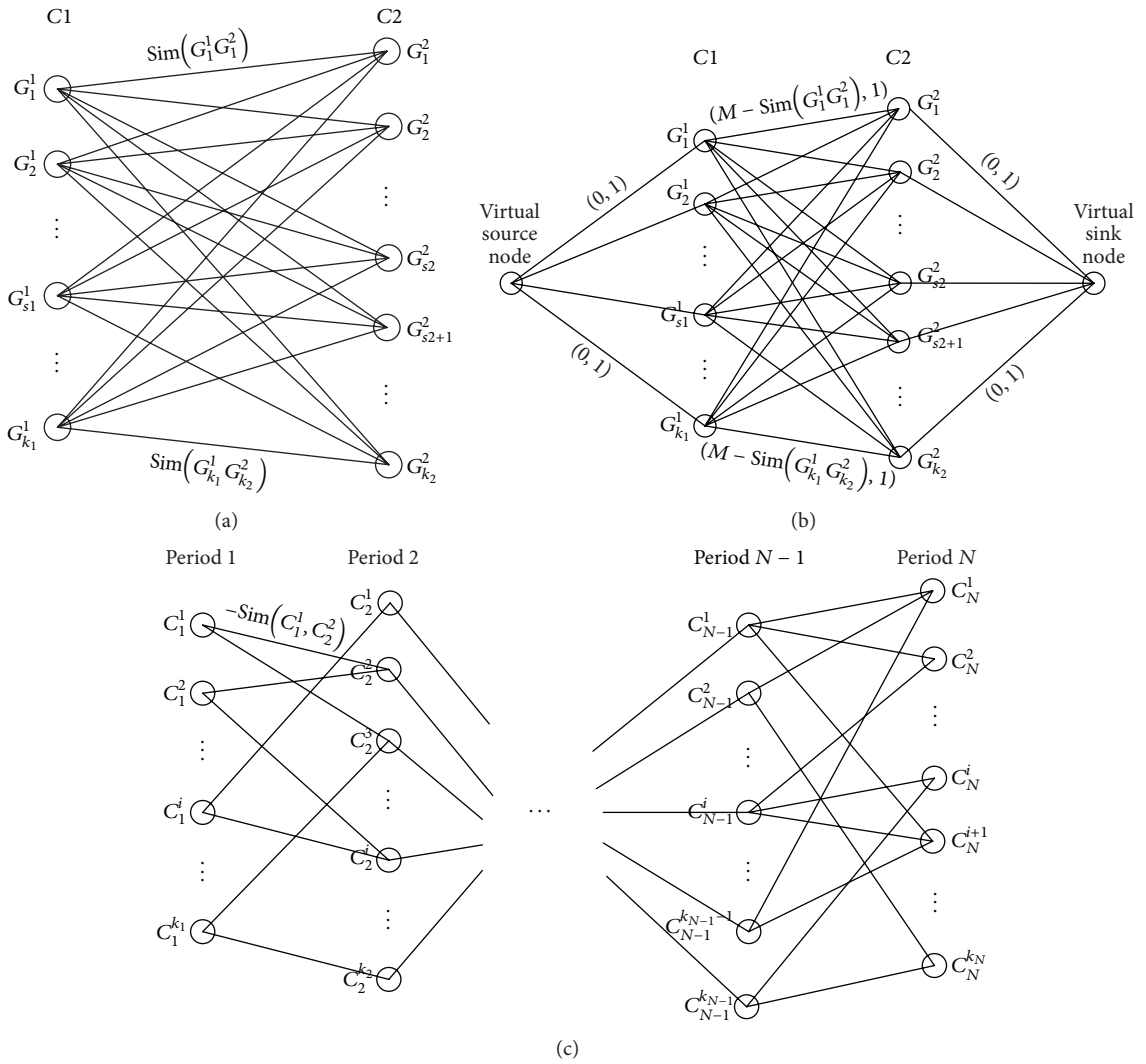


Figure 10: Configuration similarity and transition.

number of nondominated configurations in period t. Each edge in the network stands for a possible transition from the previous period to the next period. Edges standing for transitions with too low similarity are truncated.

We define two kinds of parameters in the network. The first one is the weight of edge $e_{t,t+1}^{ij}$, denoted as $w_{t,t+1}^{ij}$, set as the inverse of the similarity of configuration C_t^i and C_{t+1}^j. The second one is the values of *Obj1* to *Obj3* of node C_t^i, denoted as $f_t^i = (f_{t,1}^i, f_{t,2}^i, f_{t,3}^i)$.

Definition 7. One defines a route in the forward network from one node to any node in the last period as a configuration link. A link without its starting node is called a path.

For node C_t^i, one denotes the cost along a path $p_{i,t}^p$ from it to any node in the last period as $g^{p_{i,t}^p} = (g_1^{p_{i,t}^p}, g_2^{p_{i,t}^p}, g_3^{p_{i,t}^p}, g_4^{p_{i,t}^p})$, where $g_1^{p_{i,t}^p}$ is the sum of similarities along the path and the other three items correspond to the sum of values of three objective functions of nodes along the path. Hence, one can define the dominating relationship between two paths.

Definition 8. Assuming two paths form the same time period t to the last period as $p_{i,t}^p$ and $p_{j,t}^q$, $p_{i,t}^p \prec p_{j,t}^q$, if $g^{p_{i,t}^p} \le g^{p_{j,t}^q}$ and $\exists m, 1 \le m \le 4$, $g_m^{p_{i,t}^p} < g_m^{p_{j,t}^q}$.

From Definition 8, we can define a nondominated set of paths from node C_t^i, denoted as ND_t^i. Then, we assume another path from one node in previous period, C_{t-1}^j, which goes across C_t^i as $p_{j,t-1}^r$, and the part after C_t^i along $p_{j,t-1}^r$ is $p_{i,t}^p$. Then, we can get the following relationships:

$$p_{j,t-1}^r = p_{i,t}^p \cup \left(e_{t-1,t}^{ji}, C_t^i\right), \tag{25a}$$

$$g_1^{p_{j,t-1}^r} = g_1^{p_{i,t}^p} + w_{t-1,t}^{ji}, \tag{25b}$$

$$g_m^{p_{j,t-1}^r} = g_m^{p_{i,t}^p} + f_{t,m-1}^i, \quad m = 2, 3, 4. \tag{25c}$$

Hence, we obtain the following proposition.

Proposition 9. *Assuming a nondominated set of paths originating from C_{t-1}^j and paths in this set go across C_t^i, the part after C_t^i along such path belongs to* ND_t^i.

Proof. Assume two paths originating from C_{t-1}^j and going across C_t^i as $p_{j,t-1}^p$ and $p_{j,t-1}^q$. Also, assume their parts after C_t^i as $p_{i,t}^r$ and $p_{i,t}^s$, respectively. Considering the calculations in (25b) and (25c), if $p_{i,t}^r \prec p_{i,t}^s$, then $p_{j,t-1}^p \prec p_{j,t-1}^q$. □

Proposition 9 shows that the subpaths not being in ND_t^i can be discarded when we calculate ND_{t-1}^j. We consider all nodes, C_t^i, $1 \le i \le k_t$, in period t, which can be transitioned from C_{t-1}^j. We denote the nondominated paths originating from C_{t-1}^j and going across C_t^i as $\mathrm{ND}_{t-1,t}^{ji}$, and we get

$$\overset{\smile}{\mathrm{ND}}_{t-1}^{\,j} = \bigcup_i \mathrm{ND}_{t-1,t}^{ji}. \tag{26}$$

We elite out nondominated paths in $\overset{\smile}{\mathrm{ND}}_{t-1}^{\,j}$ and get the nondominated set of paths originating from C_{t-1}^j, denoted as ND_{t-1}^j.

We have presented the procedure to obtain ND_{t-1}^j so far when ND_t^i is known. This is similar to one of the steps in typical dynamic programming except multiple objectives. In fact, the *principle of optimality* in monoobjective DP also works in multiobjective cases [39] and Proposition 9 just acts the same as the *principle of optimality*.

Hence, we apply a backward multiobjective DP algorithm starting at stage $T-1$, getting the nondominated path set for each node in period $T-1$. In stage n, $1 \le n \le T-1$, we deal with edges from period t to $t+1$ and the nodes in period $t+1$ to generate sets of nondominated paths from nodes in period t.

After the calculation in stage 1, we generate nondominated path sets from each node in period 1, denoted as $\mathrm{ND}_1^1, \mathrm{ND}_1^2, \ldots, \mathrm{ND}_1^{k_1}$. However, values of objective functions of configurations in period 1 are not involved yet. Next, we integrate nodes in period 1 into the paths to get final nondominated links.

For each configuration in period 1, C_1^i, $1 \le i \le k_1$ and corresponding ND_1^i, suppose a path in ND_1^i as $p_{i,1}^p$ and the link $l_{i,1}^p$ that integrates C_1^i with $p_{i,1}^p$; then

$$l_{i,1}^p = p_{i,1}^p \cup C_1^i, \tag{27a}$$

$$g_m^{l_{i,1}^p} = g_m^{p_{i,1}^p} + f_{1,m-1}^i, \quad m = 2, 3, 4. \tag{27b}$$

We take operations like (27a) and (27b) for all nondominated paths in ND_1^i, $1 \le i \le k_1$. Then, we classify the generated links into a set $\overset{\smile}{\mathrm{FND}}$. Finally, we select nondominated individuals from $\overset{\smile}{\mathrm{FND}}$ to obtain final nondominated configuration link set FND.

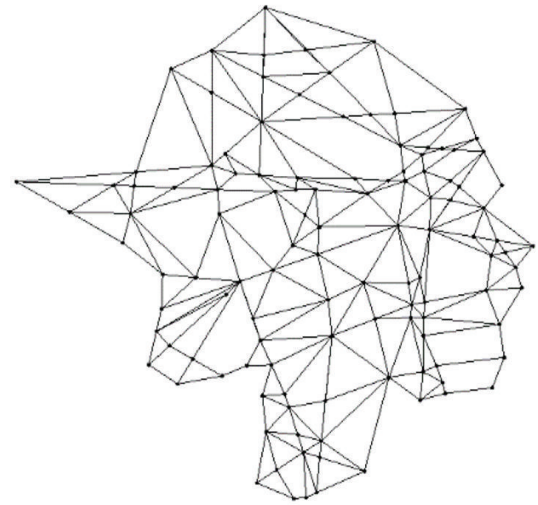


Figure 11: The structure of the undirected graph.

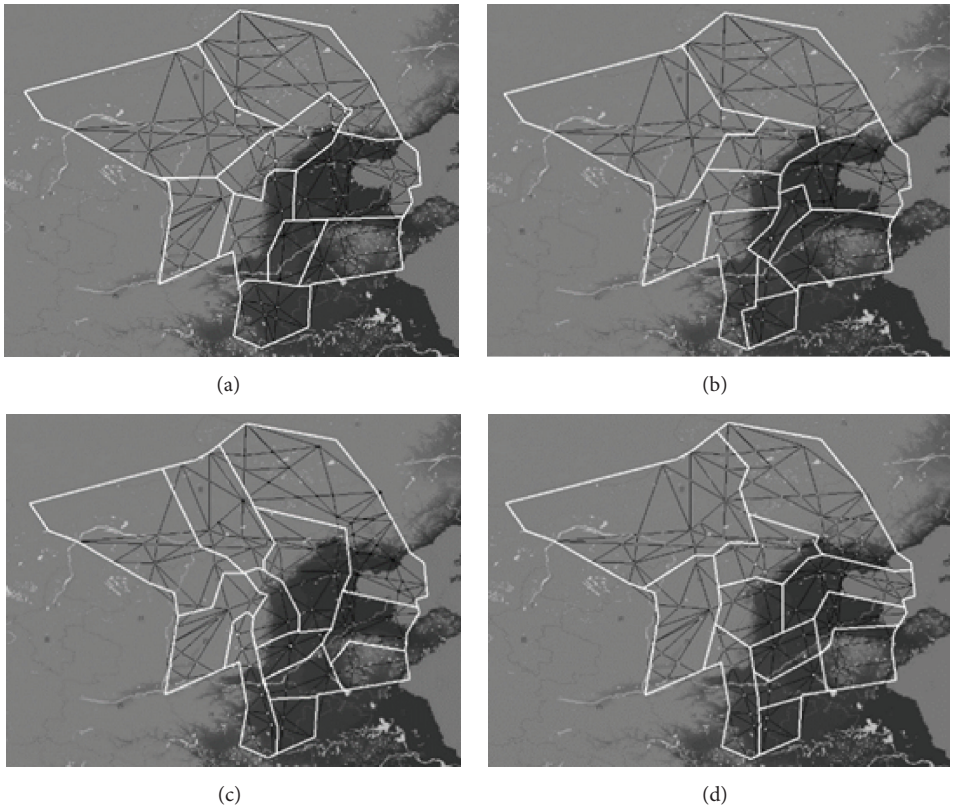
6. Experiment and Result Analysis

6.1. Setting. We test our methods with practical navigation database, recorded radar track, and practical flight plan teletext on July 20th, 2014. The target airspace is the high-altitude airspace controlled by Beijing Area Control Center (as in Figure 1). Graph G representing the airspace is made up of 117 vertices and 293 edges (239 edges represent air routes), which is shown in Figure 11.

The MDFB is set as 15 nautical miles and the minimum staying time in one sector is set as 7 minutes. The threshold of instantaneous flow, $\mathrm{Thre}_{\mathrm{insflow}}$, is set as 15. Other two thresholds in the memetic local search are set as $\mathrm{FD}_{\mathrm{up}} = 5400$ and $\mathrm{FD}_{\mathrm{low}} = 2500$. Weights in (9) are determined by interviews with air traffic controllers, $w_{\mathrm{FD}} = 5$, $w_{\mathrm{ac}} = 1$, $w_{\mathrm{sc}} = 1$, $w_{\mathrm{cc}} = 1$. Besides, we divide one day into 10 time periods, 0 a.m. to 6 a.m. is set as the first period considering its sparse traffic. After that, every two hours are set as one period such as 10 a.m. to 12 a.m. In the evolution algorithm, we set 100 individuals in one generation and the maximum times of iteration are set as 500. All simulations are executed on a PC with Core-i3 2120 CPU and 4 GB RAM.

6.2. Configuration Examples. To save space, we only demonstrate examples in four periods, 12:00–14:00, 16:00–18:00, 20:00–22:00, and 22:00–24:00. The period between 20:00–22:00 is the busiest period according to the recorded data.

We show the four nondominated configurations in the nondominated configuration link with the largest gross similarity (see Section 6.7) in Figure 12. The airspace is split into 9, 10, and 12 sectors in 12:00–14:00, 16:00–18:00, and 20:00–22:00 along with the increasing flow density. Sector number in the last period is also 12. However the traffic flow in this period is not so dense. In fact, at the beginning of period 22:00–24:00, the traffic continues the situation in 20:00–22:00, which is very dense and 12 sectors are necessary. The density gradually decreases and becomes much sparser near the midnight. However, as the configuration in one time



(a) (b)

(c) (d)

Figure 12: Configuration examples by the proposed approach.

period is fixed and it has to cover the busiest situation in this period because of the instantaneous flow constraint, the sector number during period 22:00–24:00 is also 12. If we can identify the flow density change more accurately, the pattern of time period division could be better. This is another critical and important problem in the DAC but not the scope of this paper.

In almost all periods except the period 20:00–22:00 with huge high density, northeastern, northern, and northwestern regions of the airspace are almost served by at most two sectors. This corresponds to the fact that the economy in northeast and northwest China is not so developed and flights from these regions are limited. On the contrary, in the southeastern and southern regions of the airspace, configurations changes frequently between two adjacent periods and much more sectors are needed during busy periods. In some situations, small sectors that even contain only one vertex are produced around the main stem of China's north-south air route, "A461." The peak flow density along "A461" occurs because flights from Guangzhou, Changsha, Wuhan, Zhengzhou and other big cities to Beijing are all probable to fly along "A461." Hence, we have to limit the control area of any single sector so as to limit the controllers' workload during high flow density periods.

6.3. Comparisons. In Table 1, we summarize the majority of the metrics of configurations using six different methods, respectively, including current configuration in practice, the configuration generated only by the NSCA-based method and four evolution-based methods, that is, RI, CI, NI, and MI. We compare these results in terms of values of the three objective functions (*Obj1* to *Obj3*) and violating times of the three constraints (*Con1* to *Con3*). "Nos" in Table 1 stands for the number of sectors. Density gap stands for the gap of the maximum and the minimum count of trajectory points in different sectors. For last four evolution-based methods, the values from the third column to the values in the fifth column are all averaged values of nondominated individuals in the final generation; values from the sixth to the eighth column and values in the column "density gap" are the biggest related values in the nondominated set. Column "Nos" gives the smallest sector number of single configuration in the nondominated set. Column "RT" gives the running time of evolution-based methods.

6.3.1. Comparison with Current Configuration in Practice. In Table 1, we notice the reductions in values of all three objective functions as well as the reduction of the density gap from configuration in practice to those by our approach. In addition, there are several occurrences of constraint violations in current configuration while any kind of violations are absent in our approach. Furthermore, much less sectors are needed in the proposed method comparing to the configuration in practice.

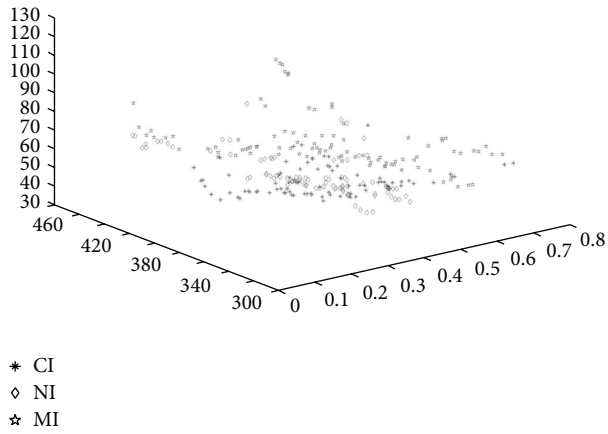


Figure 13: Nondominated sets of the period 12:00–14:00.

6.3.2. Approximation of the NSCA. The values in rows named with "Only NSCA" show that the NSCA is just an approximation of the graph N-cut. Even if we do not consider constraint violations, the configuration obtained by only applying the NSCA shows no superiority to the current configuration in practice, especially from the aspect of balancing intrasector workloads. However, "Only NSCA" often performs well in terms of *Obj2* or *Obj3*. Hence, we may expect "nice genes" from the NSCA-based method and this is why we execute NI and MI to produce the initial generation.

6.3.3. Comparison of Different Initial Generations. For every period, we set four cases about the initial generation of the CNSGA-II and list related results in last four rows of that period in Table 1. Solutions obtained in RI are the worst considering even no feasible individuals can be produced (We have tried much more iterations but no feasible individual can be produced yet.). All individuals in the final generation in CI, NI, and MI are feasible, but there is no obvious dominance considering the averaged values of three objective functions. Although in the period between 12:00–14:00, the averaged value in CI dominates the averaged results in NI and MI, and several individuals in NI and MI are still not dominated by individuals in CI (see Figure 13). Similar situations occur in other periods. That is, if we unite the nondominated sets by CI, NI, and MI together and select once again, the final approximate Pareto Front is always made up by solutions from all these three cases.

6.4. Final Nondominated Set. From the discussion in Section 6.3, we propose the idea of parallel computation with CI, NI, and MI when executing the evolution algorithms. Then we merge their final generation together and select nondominated individuals from this merged set so as to get the final solutions of the 2D ERSP in one time period.

6.5. Applicability of the Automatic Boundary Depiction Method. The most important factor that affects the applicability of the automatic boundary depiction method is the conciseness of the depicted boundaries. If the boundaries are composed of too much segments or even jagged, the method is undesirable. We randomly select 10 optimal configuration links (that correspond to 100 configurations and 8500 sectors in total) and draw the boundaries using the proposed boundary depiction method. 59284 segments are generated in total and the boundary of single sector is made up by 6.97 segments in average. Among the 59284 segments, only 34 are produced by the *OABD*. The *OABD* is called 17 times, which means that one segment in the concave hull is almost replaced by a two-part polygonal line in the *OABD*. This phenomenon may attribute to the advanced graph structure proposed in this paper that involves edges representing "Vonoroi neighborhood." With these newly added edges, the distance between two adjacent vertices on the concave hull is limited so that the line segment connecting them seldom walks through too much "obstacle circles." In all, the limited average number of segments consisting of the boundary of single sector and the fact that no jagged boundaries are produced by the *OABD* show the applicability of our automatic boundary depiction method.

6.6. Running Time. Because the sectorization problem is not a real-time decision problem and almost all solving methods need iterative optimization, few previous studies have reported the running time of their methods. However, in our mind, the running time is also important because short running time is helpful in utilizing latest traffic information.

6.6.1. Time for Data Preparation. In Section 3.3, we propose multiple data forms to store the traffic information. In our simulation, time spent to obtain traffic information for single time period ranges from 169 seconds to 324 seconds. The heavier the traffic, the more the time needed to calculate the metrics.

6.6.2. Time for Optimization. The values in the last column of Table 1 show that the evolution algorithm-based optimization is around 15 minutes for any single time period.

6.6.3. Time for Boundary Depiction. The time spent to draw the sector boundaries for single configuration is about 1.5 minutes. The most time-consuming part is judging whether segments on boundary polygons intersect with inner-edges of sectors or whether they intrude obstacle circles. Besides, as we do not need to depict the boundaries during the optimization process and the depiction is only needed once after the evolution, the 1.5 min running time is acceptable and it is expected to have advantages over methods with manual operations.

In all, in our simulation, it takes about 20 minutes from the data preparation to the sector boundary depiction for single configuration during a single time period. This implies if we want to get a configuration for next 2 hours, we can start just about 20 minutes before the starting moment of the period, which will benefit our approach with latest air traffic information so that to minimize the impact of uncertainty. In addition, as the approaches are all executed on a PC with

Core-i3 CPU, we can expect much shorter running time if we use a computer with more computing power.

6.7. Results of the Nondominated Configuration Links. We randomly select 20 nondominated solutions (configurations) for each time period; that is, the number of links of configurations in the forward network is about 20^{10}. Among them, we get 5284 nondominated links. One of the links with the largest gross similarity is $C_1^{13} \rightarrow C_2^{22} \rightarrow C_3^{7} \rightarrow C_4^{97} \rightarrow C_5^{56} \rightarrow C_6^{143} \rightarrow C_7^{215} \rightarrow C_8^{97} \rightarrow C_9^{112} \rightarrow C_{10}^{9}$. In fact, we have demonstrated the results of C_5^{56}, C_7^{215}, C_9^{112}, and C_{10}^{9} in Figure 12.

We must point out that these planned links can serve as candidates and they however may not be employed definitely. Air traffic management officers can choose one such link at the beginning of a day in order to start some necessary preparations for subsequent periods, such as determining the schedule of controllers and previewing the sectors. Although exact configurations applied may change because of uncertainties in the air traffic system, such changes should not be too serious if no emergency occurs.

7. Conclusion and Future Work

This study solves the 2D ERSP with approaches based on weighted undirected graph cuts. We comprehensively considered the multiple objectives and constraints of the 2D ERSP. We employed memetic local search-embedded CNSGA-II to evolve the initial clusters and proposed several methods to produce initial generation. We also proposed a concave hull-based method to draw exact sector boundaries. In addition to the sectorization problem, we also solved the CTP, which generated a series of nondominated links of configurations for several consecutive time periods.

The simulation results showed significant improvement from the configuration applied in practice in almost all important aspects of the sectorization problem. Along with the results in depicting boundaries and in the CTP, we can claim that the proposed approaches are promising practical decision support tools for DAC considering their comprehensiveness, practicality, automatization, and efficiency.

However, there are still several important issues we should address in the future:

(1) Other Automatic Boundary-Depicting Algorithms. As we stated in Section 1, powerful boundary-depicting algorithms for undirected graph cut-based approaches are very limited. Searching for more efficient depicting algorithms is very important.

(2) Interaction between Traffic Flow Management (TFM) and DAC. Both TFM and DAC are key problems in air traffic management, and their outputs seem to act as inputs to each other. Can we find a logic to deal with them simultaneously? Research in this direction is already underway, but no general architecture or principle has been proposed.

(3) Local Adjustment of Airspace Configuration. We mentioned in Section 6.7 that we can plan a configuration link at the beginning of a day and follow the link if no emergency occurs. However, when an emergency occurs that sharply changes the traffic flow pattern, we must adjust the configuration temporarily. This local adjustment must be fulfilled very soon after the emergency; therefore, algorithms dealing with local adjustment must be very efficient.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research is supported in part by the National Natural Science Foundation of China Grant 61179052, in part by the 973 National Basic Research Program of China Grants 2010CB731805 and 2010CB731806, and in part by Aviation Science Fund of China (20128058005). The authors appreciate the corporation and precious data offered by Beijing Area Control Center of Civil Aviation Administration of China and Air China.

References

[1] D. Gianazza, J. M. Alliot, and G. Granger, "Optimal combinations of air traffic control sectors using classical and stochastic methods," in *Proceedings of the 2002 International Conference on Artificial Intelligence*, Las Vegas, Nev, USA, June 2002.

[2] R. Geng, *Research on the dynamic airspace management and collaborative air traffic flow optimization [Ph.D. thesis]*, Tsinghua University, Beijing, China, 2009.

[3] C. Verlhac and S. Manchon, "Optimization of opening schemes," in *Proceedings of the 4th USA/Europe Air Traffic Management R & D Seminar*, Santa Fe, NM, USA, December 2001.

[4] M. Bloem and P. Gupta, "Configuring airspace sectors with approximate dynamic programming," in *Proceedings of the 27th Congress of the International Council of the Aeronautical Sciences (ICAS '10)*, pp. 4085–4097, Nice, France, September 2010.

[5] P. Kopardekar, K. Bilimoria, and B. Sridhar, "Initial concepts for dynamic airspace configuration," in *Proceedings of the 7th AIAA Aviation Technology, Integration, and Operations Conference*, pp. 695–706, Belfast, Northern Ireland, September 2007.

[6] D. Delahaye, J. M. Alliot, M. Schoenauer, and J. L. Farges, "Genetic algorithms for partitioning air space," in *Proceedings of the 10th Conference on Artificial Intelligence for Applications*, pp. 291–297, San Antonia, Tex, USA, March 1994.

[7] K. Leiden, P. Steve, and S. Quesada, "Flight level-based dynamic airspace configuration," in *Proceedings of the 8th AIAA Aviation Technology, Integration and Operation Conference (ATIO '09)*, Hilton Head, SC, USA, September 2009.

[8] S. C. Han and M. Zhang, "The optimization method of the sector partition based on metamorphic voronoi polygon," *Chinese Journal of Aeronautics*, vol. 17, no. 1, pp. 7–12, 2004.

[9] M. Xue, "Airspace sector redesign based on Voronoi diagrams," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, no. 12, pp. 624–634, 2009.

[10] M. Xue, "Three-dimensional sector design with optimal number of sectors," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 609–618, 2012.

[11] J. J. Tang, S. Alam, C. Lokan, and H. A. Abbass, "A multi-objective approach for Dynamic Airspace Sectorization using agent based and geometric models," *Transportation Research Part C: Emerging Technologies*, vol. 21, no. 1, pp. 89–121, 2012.

[12] A. Basu, J. S. B. Mitchell, and G. K. Sabhnani, "Geometric algorithms for optimal airspace design and air traffic controller workload balancing," *Journal of Experimental Algorithmics*, vol. 14, article 3, 2009.

[13] R. Kicinger and A. Yousefi, "Heuristic method for 3D airspace partitioning genetic: algorithms and agent-based approach," in *Proceedings of the 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO '09)*, Hilton Head Island, SC, USA, September 2009.

[14] C. R. Brinton and S. Pledgie, "Airspace partitioning using flight clustering and computational geometry," in *Proceedings of the IEEE/AIAA 27th Digital Avionics Systems Conference (DASC '08)*, pp. B31–B310, St. Paul, Minn, USA, October 2008.

[15] G. Yang, M. H. Hu, and Y. J. Wang, "Airspace sector structure optimization design based on nonlinear programming," *Journal of Transportation Engineering and Information*, vol. 6, no. 4, pp. 82–86, 2008.

[16] A. Yousefi and L. G. Donohue, "Temporal and spatial distribution of airspace complexity for air traffic controller workload-based sectorization," in *Proceedings of the AIAA 4th Aviation Technology, Integration and Operations Forum (ATIO '04)*, Chicago, Ill, USA, September 2004.

[17] A. Yousefi, *Optimal airspace design with air traffic controller workload-based partitioning [Ph.D. thesis]*, George Mason University, Fairfax, Va, USA, 2005.

[18] M. Drew, "Analysis of an optimal sector design method," in *Proceedings of the IEEE/AIAA 27th Digital Avionics Systems Conference (DASC '08)*, pp. 3.B.4-1–3.B.4-10, IEEE, St. Paul, Minn, USA, October 2008.

[19] A. Klein, "An efficient method for airspace analysis and partitioning based on Equalized Traffic Mass," in *Proceedings of the 6th USA/Europe Air Traffic Management Research and Development Seminar (ATM '05)*, pp. 2–11, Baltimore, Md, USA, June 2005.

[20] P. Jagare, P. Flener, and J. Pearson, "Airspace sectorisation using constraint-based local search," in *Proceedings of the 10th USA/Europe Air Traffic Management Research and Development Seminar (ATM '13)*, Chicago, Ill, USA, June 2013.

[21] J. H. Li, T. Wang, M. Savai, and I. Hwang, "Graph-based algorithm for dynamic airspace configuration," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 4, pp. 1082–1094, 2010.

[22] S. A. Martinez, G. B. Chatterji, S. Dengfeng, and A. M. Bayen, "A weighted-graph approach for dynamic airspace configuration," in *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, pp. 1476–1491, Hilton Head Island, SC, USA, August 2007.

[23] Y. Chen and D. Zhang, "Dynamic airspace configuration method based on a weighted graph model," *Chinese Journal of Aeronautics*, vol. 27, no. 4, pp. 903–912, 2014.

[24] H. Trandac, P. Baptiste, and V. Duong, "Airspace sectorization with constraints," *RAIRO Operations Research*, vol. 39, no. 2, pp. 105–122, 2005.

[25] S. Zelinski, "A comparison of algorithm generated sectorizations," *Air Traffic Control Quarterly*, vol. 18, no. 3, pp. 279–289, 2010.

[26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[27] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," Tech. Rep. 826,C, Caltech Concurrent Computation Program, Pasadena, Calif, USA, 1989.

[28] M. P. Savai, H. Li, T. Wang, and I. Hwang, "An algorithm for adaptable dynamic airspace configuration," in *Proceedings of the 10th AIAA Aviation Technology, Integration, and Operations Conference (ATIO '10)*, Fort Worth, Tex, USA, September 2010.

[29] R. K. Ahuja, T. L. Magnanti, and B. J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

[30] F. Netjasov, M. Janić, and V. Tošić, "Developing a generic metric of terminal airspace traffic complexity," *Transportmetrica*, vol. 7, no. 5, pp. 369–394, 2011.

[31] P. Kopardekar and S. Magyarits, "Measurement and prediction of dynamic density," Tech. Rep., NASA Ames Research Center, Moffett Field, Calif, USA, 2003.

[32] D. Gianazza, "Evaluation of air traffic complexity metrics using neural networks and sector status," in *Proceedings of the 2nd International Conference on Research in Air Transportation*, Belgrade, Serbia, 2006.

[33] F. Chung, *Lectures on Spectral Graph Theory*, American Mathematical Society, Washington, DC, USA, 1997.

[34] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, pp. 298–305, 1973.

[35] A. Cook and G. Tanner, "European airline delay cost reference values," Tech. Rep., Eurocontrol Performance Unit, Department of Transport Studies, University of Westminster, London, UK, 2011.

[36] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," in *Advances in Neural Information Processing Systems*, T. Dietterich, T. Becker, and Z. Ghahramani, Eds., vol. 14, pp. 849–856, MIT Press, Cambridge, UK, 2002.

[37] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[38] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 1983.

[39] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, Athena Scientific, Nashua, NH, USA, 3rd edition, 2005.