Towards Simplified Optimal Sector Splitting

Leonid Sedov and Valentin Polishchuk
Communications and Transport Systems
ITN, Linköping University, Sweden
firstname.lastname@liu.se

Billy Josefsson
LFV
firstname.lastname@lfv.se

***Abstract*—Merging and splitting control sectors is a certified way to address demand-capacity imbalances in an airspace: during high-traffic hours a sector is split into two or more smaller sectors, while in low-traffic hours the parts are merged back. In this paper we explore ways of splitting a sector with simple straightline cut so as to balance, as well as possible, the number of controlled aircraft in the obtained smaller sectors. The approach is verified by computational experiments with enroute traffic over Swedish airspace.**

I. Introduction and Related work

Demand-capacity imbalances are a major challenge in airspace management, and initiatives like dCDB (dynamic Demand Capacity Balancing) within SESAR and DAC (Dynamic Airspace Configuration) in NextGen are specifically addressing it. The issue is that the traffic density is changing over time, so static control sectors may become under- and overloaded during a day. To alleviate the pressure on the air traffic controllers (ATCOs), sectors are being split during high traffic hours; to save the operational costs, the sectors are merged back when the traffic density decreases.

Finding optimal ways to split and merge sectors is a fundamental research problem in ATM, which has attracted a lot of attention over the years. One of the most important distinction between algorithmic sectorization paradigms is whether the resulting sectors are built up by gluing together some "elementary bricks" (hexagons [1], square pixels [2], 3D voxels [3], etc.), or are obtained by directly cutting the airspace into the sectors [4], [5], [6] (see survey [7] for a full taxonomy of approaches to sectorization). Work of the former type employs an Integer Program (IP) to build up the sectors optimally – this synthesis approach provides a lot of means to fine-tune sector design, but gives little control over the sectors shape (the sectors are produced by the "blackbox" IP); therefore additional tricks and/or postprocessing are then required to even keep each sector (simply) connected, and further care is needed to ensure that the sectors boundaries are smooth. On the contrary, direct partitioning of the airspace into sectors allows one to influence the sectors geometries within the algorithm (producing, in particular, convex sectors), but gives little flexibility in taking care of complex constraints and objectives.

The research area remains active: a new form of sectorization IP was developed in [8], [9], and novel tools, like simulated annealing [10], are also being explored.

Our contribution

We consider a sectorization approach falling into the second (geometric decomposition algorithms) category from the two paradigms outlined above. In a nutshell, the major difference between this paper and earlier research is that previous work aimed at producing sectors of potentially complicated shapes (including 3d) taking into account the whole variety of sectorization quality measures; naturally, due to the complexity of the considered model, it was impossible to require that the obtained sectors are truly optimal according to some objective. Contrary to that, we consider only two KPIs (which, moreover, are based on a single input – the number of flights in the sector), but aim at attaining a perfect sector split into two parts in terms of the objectives. We show that, surprisingly, it is possible to find sectors which simultaneously balance the maximum and the average aircraft count, using the simplest straightline cut through the airspace. As a by-product, we obtain sectors of simple (convex) shapes and simple (straightline) boundary between the sectors.

The rest of the paper is organized as follows: The next section outlines the model. Section III presents problem formulation and our solution to it, culminating in our main result: simple straightline cut suffices to perfectly balance traffic between the sectors (Theorem 4). Section IV illustrates the use of our method on a synthetic example and reports on computational experiments with real-world flight data. The last section discusses possible extensions.

II. Modeling

This section describes our abstraction of the sectorization problem.

A. Complexity based on density

Prior research has singled out several desirable properties that a good sectorization should possess, see e.g., [7], [11]. The foremost requirement is the balance of control between the sectors – no sector should be overloaded with traffic while other sectors see only few airplanes. We therefore aim at balancing two measures of traffic complexity between the sectors:

1) The *maximum* number of aircraft in the sector over time
2) The *average* number of aircraft in the sector over time

The former is a standard measure of sector load, used in many prior works; it corresponds to the *Monitor Alert Parameter (MAP)* – the number of aircraft in the sector that should not

be exceeded. The latter measure is proportional to the *average dwell time* of aircraft in the sector [12]. While the maximum aircraft count represents the *peak* complexity of the sector, the average count represents the *total* complexity over the day. Balancing the two objectives tends to give a fair distribution of the traffic between the sectors.

We acknowledge that looking only at the number of aircraft (and not at flight directions) is a delimitation of our model, as it ignores the control complications caused by converging aircraft trajectories (the celebrated Dynamic Density [13], [14], [15]). We believe that our setup nevertheless provides a good starting point for more elaborated models. This was confirmed with operational experts from Nordic Unified Air traffic Control (NUAC) and Swedish ANSP Luftfartsverket (LFV) who manage enroute traffic over Sweden – the subject of our experiments (Section IV): the need for complex deconfliction maneuvers is rare in the uncongested Nordic airspace. We remark that aircraft count and, in particular, its maximum over time, was, in fact, in focus also in most of the earlier work on geometric airspace partitioning.

B. Binary split

In a generic sectorization problem, the input are flight trajectories through an airspace, and the goal is to partition the airspace into some number K of sectors, while redirecting a set of constraints. We consider the simplest case of splitting into only $K = 2$ sectors. While in most practical cases, one would be looking for a larger number of sectors, our split into 2 parts may be used as the basic ingredient for a more complex sectorization: since the binary split increases the number of parts by 1, it can be applied recursively to obtain an arbitrary number $K > 2$ of sectors (however, if K is not a power of 2, some parts may not be perfectly balanced). The recursive splitting of high-traffic sectors may be motivated also by the subsequent recursive merging: when the sectors are to be merged during low-traffic hours, it is important that the merged sectors also have balanced workload and bounded maximum complexity.

C. Flight segments

We assume that aircraft trajectories are given as a set of time-stamped segments – the standard data format in EUROCONTROL's demand data repository (DDR2) which we use in our experiments (Section IV).

D. Straight boundary

It is commonly acknowledged that the existing sectors have suboptimal geometries, often following state boundaries (which were optimized for anything but ATC); in fact, SESAR Joint Undertaking puts a lot of effort into removing country boundaries from consideration when designing the sectors – one important initiative is establishing FABs (functional airspace blocks) over the harmonised European skies. A good sector should have "nice" shape and "simple" boundary – subjective criteria which are often formalized by requesting that the sectors are *convex* [7], [11], [9] (convexity is a desirable property also from the operational point of view because, by definition of convexity, any straightline-segment flight intersects a convex region at most once, thus never re-entering the sector). Note that if the boundary between two sectors has a vertex (i.e., is *not* a single straightline segment), then one of the sectors is not convex (Fig. 1, left). Thus, to ensure convexity, we require that the sector boundary is a straightline segment, or equivalently that the sectorization is obtained by a single straightline *chord* joining two points on the boundary of the airspace (Fig. 1, right).

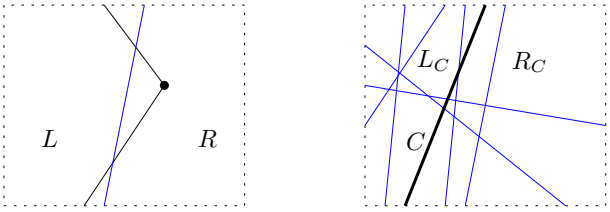

Figure 1. Left: A rectangular airspace (dotted) partitioned into sectors L and R; a "kink" (black dot) in the boundary between the sectors makes sector R non-convex – a flight segment (blue) may exit the sector and re-enter it again. Right: Chord C (bold) is the boundary between sectors L_C and R_C; blue segments are flights

As an additional constraint we require that the chord does not go through a flight segment, i.e., no flight segment should belong to the chord. This restriction makes sense from the operational perspective – the control over the plane should not be shared between the sectors (the only time when two ATCOs have to attend to the flight is when it crosses between the sectors). In fact, even stronger forms of this restriction may be imposed – that any flight segment lies somewhat "deep" inside within the sector (the distance between the flight and the sector boundary is larger than a given tolerance parameter), that the angle at which the flight segment crosses the boundary is not too small, etc.

III. Problem formulation and Solution

Formally, the input to our problem consists of:

- The region of interest P (the airspace) which is to be split into sectors. We assume that P is convex (for otherwise, decomposing P into convex sectors may not be possible).
- A set S of n straightline segments, where each segment $s \in S$ is a 6-tuple $(a_s, b_s, c_s, d_s, e_s, f_s)$ where (a_s, b_s), (c_s, d_s) are the coordinates of the segment start and end points resp., and e_s, f_s are the times when the aircraft enters and leaves s resp. For simplicity we assume that the segments are traversed by the planes with the same speed (the assumption is not essential and can be easily lifted). Equivalently, each element of S is a segment in the (x, y, t)-space (Fig. 2, left). We will also make the General Positioning Assumption (GPA) that S is non-degenerate in the sense that no 3 segments go through a common point, no 3 segment endpoints are collinear, etc. (we will include other assumptions into GPA as needed) – one can make the GPA hold by perturbing the input. We will often identify a segment with the flight following

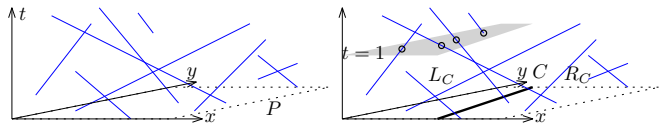

Figure 2. Left: The rectangular region of interest P (dotted) and the input segments in the (x, y, t)-space. Right: P split into two sectors by the chord C (bold). $L_C(1) = 4$ is number of segments intersecting the left sector at time $t = 1$ (the intersection points are the circles)

it and speak e.g., about the time when a segment enters a region, and such.

Our output is a chord C – a (directed) straightline segment with endpoints on the boundary of P. The chord splits P into two parts – the sector L_C to the left of the chord and R_C to the right of C.

Let $L_C(t)$ be the number of flights inside the left sector at time t, or equivalently the number of segments intersecting L_C lifted to height t in the (x, y, t)-space (Fig. 2, right); define $R(t)$ analogously. Let $ML(C) = \max_{t\in[0,T]} L_C(t)$ denote the maximum aircraft count in L_C, where T is the given time horizon. The maximum count in R_C is defined analogously: $RL(C) = \max_{t\in[0,T]} R_C(t)$.

The average number of aircraft in L_C is

$$AL'(C) = \frac{1}{T}\int_{t\in[0,T]} L_C(t)\,\mathrm{d}t \tag{1}$$

By ergodicity, this average number is proportional to the total length of the segments from S inside L_C. Indeed, for a segment $s \in S$, let $1(s, t)$ be the indicator function of the presence of s in the sector over time: $1(s, t) = 1$ if s is in L_C at time t, and $1(s, t) = 0$ otherwise. Then $\int 1(s, t)\,\mathrm{d}t$ is the time that the aircraft, following s, spends in L_C:

$$\int 1(s,t)\,\mathrm{d}t = |s \cap L_C|/v \tag{2}$$

where $|s\cap L_C|$ is the length of s inside the sector and v is the speed of the aircraft. Counting separately the contribution of each aircraft, we can write $L_C(t) = \sum_{s\in S} 1(s, t)$ and rewrite (1) as

$$AL'(C) = \frac{1}{T}\int \sum_s 1(s,t)\,\mathrm{d}t \tag{3}$$

Interchanging the integration with the summation and using (2), we obtain

$$AL'(C) = \frac{1}{Tv}\sum_{s\in S} |s \cap L_C|$$

Since v and T are independent of C, we use

$$AL(C) = \sum_{s\in S} |s \cap L_C|$$

i.e., the total length of the segments in L_C, as the measure of the average aircraft count in the sector. We define

$$AR(C) = \sum_{s\in S} |s \cap R_C|$$

analogously.

The objectives

As the measures of how good a split a chord makes, we use the differences in the maximum and average numbers of aircraft in the left and right sectors. Specifically, for a chord C, the *max-imbalance* or *M-imbalance* is defined as $M(C) = |ML(C) - MR(C)|$; similarly, the *avg-imbalance* or *A-imbalance* is $A(C) = |AL(C) - AR(C)|$. With the above defined notation, our problem can be formally stated as

> *Given the airspace P and the set S of flight segments, find the chord C balancing the maximum and average aircraft count between the sectors L_C and R_C, i.e., the chord minimizing the M- and A-imbalances $M(C)$ and $A(C)$.*

Since $A(C)$ and $A'(C)$ differ by a factor of vT, minimizing $A(C)$ is equivalent to minimizing $A'(C)$.

A. Listing M-imbalances

We prove that even though there are uncountably many chords, the number of possible max-imbalances is only quatric in the number of segments, because many of the chords lead to the same M-imbalance:

Theorem 1. *The number of distinct max-imbalances is $O(n^4)$.*

Proof. Let us look closer at the interaction between chords and segments in S. In this proof we will work in the (x, y, t)-space, so by a chord uv we will actually mean the vertical plane in the space, going through the points u and v on the boundary of P in the (x, y)-plane.

Let H be the set of the horizontal planes $t = e_s, s \in S$ through the starting and ending points of all segments in S. We call the points in $H \cap S$ *critical*. That is, critical points are intersections of the planes with the segments (e.g., the hollow circles in Fig. 2, right are critical points).

Define the *combinatorial type* of a chord to be the subset of the critical points to the left of the chord and say that a chord is *canonical* if it passes through two critical points (Fig. 3). It is easy to see that canonical chords define combinatorial types: consider any chord and transfer it parallel to itself until it hits a critical point, and then rotate the chord around the hit point until it hits another critical point (i.e., until the chord becomes canonical); during the transfer and the rotation, the combinatorial type of the chord does not change (since changing the type requires moving over a critical point). Moreover, chords of the same combinatorial type have the same M-imbalance. To show this, use the same transfer-and-rotation to move any chord C onto the canonical chord C^* of the same type. During the motion, the sector boundary does not intersect any segment, hence L_C and L_{C^*} contain the same subset of the aircraft at *any* time t, meaning that also the maximum number of aircraft in the sectors are the same $ML(C) = ML(C^*)$. By the same argument, $MR(C) = MR(C^*)$, and so the max-imbalances of all chords having the same type as C^* is the same.

Finally, since $|S| = n$, there are $O(n^2)$ critical points and $O(n^4)$ canonical chords. Hence there are also $O(n^4)$ different possible values for $M(C)$ – one per chord combinatorial type;

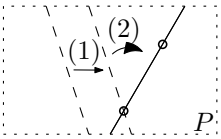

Figure 3. C (dashed) is moved (1) and rotated (2) until becoming canonical (bold). Circles are critical points

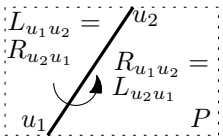

Figure 4. As the A-balanced split rotates, the sectors swap places

moreover, they all can be listed algorithmically by computing the canonical chords. □

It follows from the above proof that the M-imbalance is a "continuous" function of C:

Corollary 2. *As the chord C moves infinitesimally, $M(C)$ changes by at most 1.*

Proof. Changing the maximum count $ML(C)$ or $MR(C)$ by 2 would require the chord to jump over 2 critical points simultaneously, which is not possible by the General Positioning Assumption. Moreover, as C moves over a single critical point, only one of $ML(C), MR(C)$ changes by 1; thus also the difference $M(C) = |ML(C) - MR(C)|$ changes by 1. □

We use the corollary in the next subsection to prove our main result (Theorem 4).

B. Minimizing both M- and A-imbalance

Previous subsection showed how to find all possible max-imbalances. However, it did not show how to balance the maximum counts, and did not consider avg-imbalance at all. The reminder of this section fills both gaps: we "scroll through" all A-imbalances and show that the maximum and the average may be balanced *simultaneously*.

The crucial fact is that for any position of one endpoint of the chord, there exists a unique position of the other endpoint that vanishes the A-imbalance:

Lemma 3. $\forall u \in \partial P \, \exists! v : A(uv) = 0$

Proof. If v is immediately clockwise from u along the boundary ∂P of the airspace, then L_{uv}, and hence $AL(uv)$ are very small (essentially $AL(uv) = 0$); on the contrary, if v is immediately counterclockwise from u, then L_{uv} is (almost) the entire P, and hence $AL(uv)$ is large (essentially equal to the total length of all segments). Since $AL(uv)$ changes continuously and monotonically as v moves along the boundary of the airspace, by the Intermediate Value Theorem, there exists a unique point v where $AL(uv)$ passes over half of the length of all the segments. □

In fact, the explicit dependence $v(u)$ of v on u may be derived using the Implicit Function Theorem: One may (re)define the combinatorial type of a chord as the set of segments endpoints that are to the left of the chord (here, unlike in there previous subsection, we work in 2d); similarly to the previous section, it can be shown that there is only a quadratic number of the types, and that they are defined by the set $\mathcal{L}$ of $O(n^2)$ lines passing through all pairs of segments' endpoints. One may scroll then through all combinatorial types. Chords from the same type have their endpoints on two fixed arcs of ∂P (the arcs are defined by lines from $\mathcal{L}$), and intersect the same set of segments from S (since the set of intersected segments is uniquely determined by the endpoints to the left of the chord). Thus, for a fixed combinatorial type, and hence fixed arcs to which u and v belong, the total length $AL(uv)$ of segments in L_{uv} is given by the same formula which may be explicitly written. If u and v are such that $AL(uv) = AR(uv)$, then $AL(uv) = const$ (the common value of $AL(uv)$ and $AR(uv)$ is equal to the half of the total length of all segments). Setting the full differential to 0

$$\frac{\partial AR(u,v)}{\partial u}\mathrm{d}u + \frac{\partial AR(u,v)}{\partial v}\mathrm{d}v = 0$$

we obtain, in accordance with the Implicit Function Theorem,

$$\frac{\mathrm{d}v}{\mathrm{d}u} = -\frac{\partial AR(u,v)/\partial u}{\partial AR(u,v)/\partial v} \tag{4}$$

Equation (4) may be solved for $v(u)$ (analytically or numerically).

We are now ready to prove our main result – existence of the "doubly-balanced" split:

Theorem 4. $\exists C : M(C) = A(C) = 0$

Proof. For a point $u \in \partial P$ on the airspace boundary, let $C(u) = C(uv(u))$ be the chord that vanishes the A-imbalance: $A(C(u)) = 0$ (existence and uniqueness of $C(u)$ is guaranteed by Lemma 3). Let $M(u) = M(C(u))$ be the M-imbalance of the chord.

Take an arbitrary point $u_1 \in \partial P$, and let $u_2 = v(u_1)$ be such that $A(u_1u_2) = 0$. By symmetry, $u_1 = v(u_2)$, i.e., $C(u_2)$ is the same chord as $C(u_1)$, just going in the opposite direction (Fig. 4). In particular, the left and right sectors of the chords swap places ($L(u_1u_2) = R(u_2u_1), R(u_1u_2) = L(u_2u_1)$), implying that

$$ML(u_1u_2) - MR(u_1u_2) = MR(u_2u_1) - ML(u_2u_1) \tag{5}$$

Now, suppose that $M(u_1) \neq 0$, e.g., that $ML(C(u_1)) > MR(C(u_1))$. Move u_1 along ∂P. By Corollary 2, during the motion $M(u_1)$ changes continuously "in integers", i.e., changes in increments/decrements of 1. By equation (5), when u_1 gets to u_2, the M-imbalance changes the sign. Thus, somewhere between u_1 and u_2 there exists a point u where $M(u)$ crosses 0. □

One may find the doubly-balanced split using equation (4) to "watch" how $C(u)$ changes as u moves along the boundary of P, and using Theorem 1 to keep track of $M(u)$.

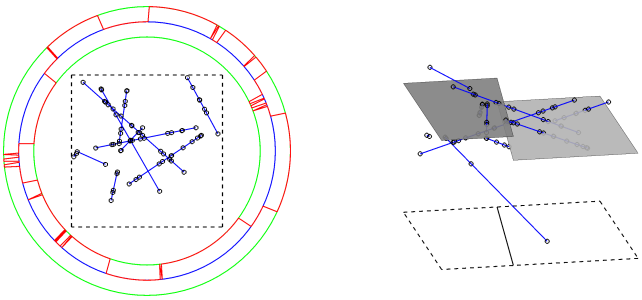

Figure 5. Left: Black circles are the critical points; the M-imbalance changes in increments or decrements of 1 (the red stepwise function). Right: The sectors shifted up to the times at maximum aircraft counts

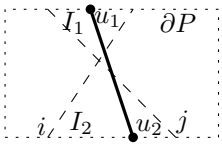

Figure 6. Two canonical chords (dashed) and points u_1, u_2 in the intervals I_1 and $I_2 = (i, j)$ on ∂P. If $AL(u_1u_2) > AR(u_1u_2)$, the next candidate location for $u_2 = v(u_1)$ is tried clockwise of the current u_2; otherwise, the search for u_2, delivering $A(u_1u_2) = 0$, continues counterclockwise

IV. Experiments

Figure 5 shows a small synthetic example. On the left, the red circular stepwise curve is the graph of $ML(uv(u)) - MR(uv(u))$ as the function of u. For visualization clarity, the location of u is represented by points on a circle surrounding the rectangular airspace (instead of points on the boundary of the airspace itself). That is, the "u-axis" of the graph is the blue circle around the airspace, and the value of the function (the red curve) is given by the distance from the center of the circle. In fact, the function takes only three values (-1,0,1) – the circles represent the levels of the function. It can be seen that the difference $ML(uv(u)) - MR(uv(u))$ indeed changes by 1 at certain points (and, in fact, stays equal to 0 at quite many angle ranges), and that the graph is centrally symmetric about the circle center, in accordance with the theory developed in the previous section. On the right, the times of maximum aircraft counts are shown.

For a real-world example, we took airspace over Sweden. The Swedish Flight Information Region (FIR) was enclosed into a rectangle P, to avoid dealing with the complicated outer boundary, which mostly follows the state boundary. A busy day (T=24hrs) of flight data was downloaded from Eurocontrol's DDR2 repository. We restricted attention to the FIR overflights: our focus is on partitioning the upper airspace, and those aircraft that landed and/or took off within the region would mostly be handled in the transition airspaces. All the flights were clipped off at the points where they crossed P (each of our flights had a unique entry and a unique exit point from P). We replaced every fight trajectory with the single segment between the crossing points, thus emulating free routing through the airspace (which is in place in the Danish–Swedish FAB for several years). Even though some of the flightplans did contain waypoints inside P, using the direct segments allowed us to exclude the turning points as workload hotspots (attention attractors) for ATCOs, providing yet another justification for using only aircraft count as the complexity indicators. (Alternatively, our algorithms could take as the input the flightplans together with the turning points.)

We then used Theorem 1 to zoom in on cuts with 0 max-imbalance: we drew canonical chords through pairs of critical points, which split ∂P into intervals (Fig. 6). For every pair of intervals I_1, I_2 such that for any $u_1 \in I_1, u_2 \in I_2$ the combinatorial type of the chord u_1u_2 is the same, we evaluated $M(u_1u_2)$ and kept only pairs with no M-imbalance (their existence is guaranteed by Theorem 4).

Finally, we searched for avg-balanced splits of the airspace into two parts as per Theorem 4 (since we handled traffic over the whole FIR, our parts rather represent control centers than sectors). To avoid going around the full boundary of the airspace, we looked for the A-balanced chords $A(u_1u_2)$ only for u_1, u_2 lying within the pairs of intervals I_1, I_2 that have $M(u_1u_2) = 0$ for all points $u_1 \in I_1, u_2 \in I_2$ (i.e., the intervals identified as described in the previous paragraph).

Solving the differential equation (4) may be a daunting task in practice; instead, we used monotonicity of $AL(u_1, u_2)$ as the function of u_2 and performed a binary search for the A-balanced chord $A(u_1u_2)$. Specifically, for each pair of intervals I_1, I_2, we took a large set $U \subset I_1$ of points laid out regularly in the first interval (essentially U is a dense 1d grid within I_1). For each point $u_1 \in U$ we evaluated $AL(u_1i), AR(u_1i), AL(u_1j)$ and $AR(u_1j)$ where i and j are the endpoints of I_2. If the differences $AL(u_1i) - AR(u_1i)$ and $AL(u_1j) - AR(u_1j)$ had the same sign (both positive or both negative), we discarded u_1; otherwise we knew that there exists a point $u_2 \in I_2$ such that $A(u_1u_2) = 0$. We searched for u_2 with the binary search: given a candidate location for u_2, we computed the length of segments inside $L_{u_1u_2}$, and directed the search clockwise or counterclockwise around the boundary depending on whether $AL(u_1u_2)$ was smaller or larger than the total length of all segments (see Fig. 6).

The pseudocode for finding the balanced cut is presented in Algorithm 1. The computational complexity of the algorithm is $O(n^5)$.

Figure 7, left shows the resulting split of the airspace. The chord cuts out Sweden off the congested area southeast, suggesting that there is essentially as much control work to do in Sweden as there is in the adjacent European airspace. It is worth noting that this is obtained fully automatically with our algorithms, without any human looking over the map.

Of course, Swedish aviation authorities are less excited about separating Sweden from Europe; the interest is in partitioning the Swedish airspace itself. To accomplish that, we shifted the right boundary of the rectangle R to the left, so that it runs along the longitude of 18 degrees, and recomputed the split. (Our techniques work for partitioning non-rectangular

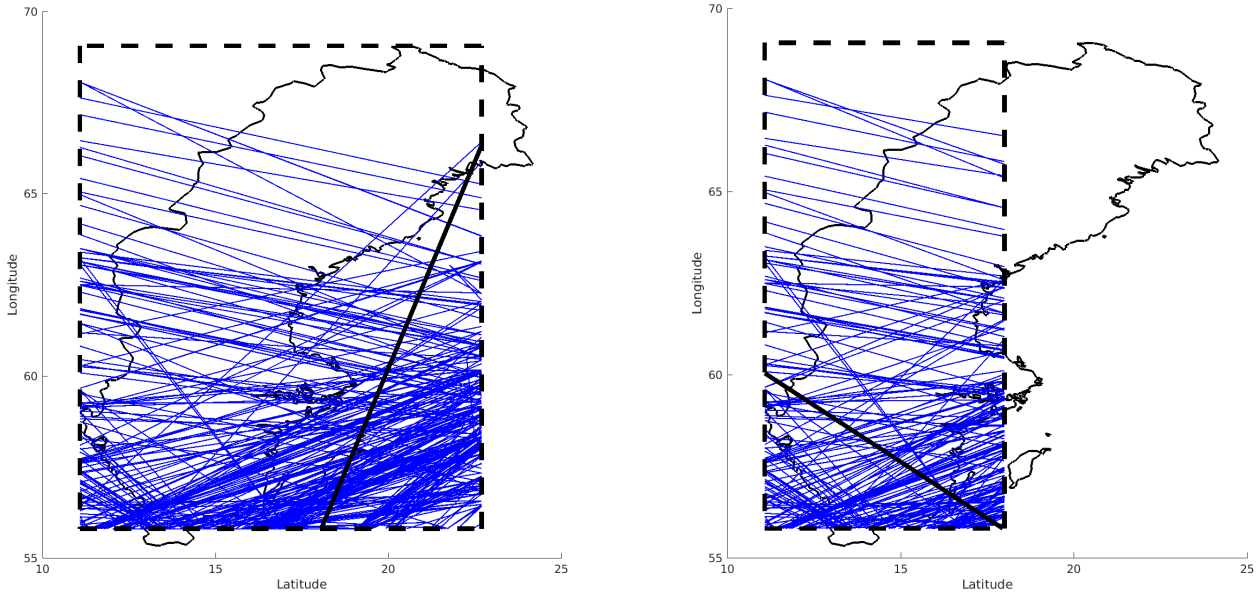

Figure 7. The M- and A-balanced split of P. Left: The rectangle is the bounding box of Swedish FIR. Right: A smaller rectangle, more tightly focusing on the flights over Sweden

airspaces just as well, but for this experiment we kept P a rectangle – shifting the boundary to the left made a northern part of Sweden fall out to the right of the rectangle, but the fallen out contained almost no traffic anyway.) Figure 7, right shows the result. Again, without any human oversight, our algorithm separated the truly more congested southern part of the Swedish FIR from the north – a reasonable division.

As mentioned in Section II-B, one natural way to proceed with our method is to do recursive balanced partitioning: split a large airspace into two and continue splitting the parts. In terms of the considered example, this would mean that *both* of our chords could be used *simultaneously*: first a SW–NE chord (like in Fig. 7, left) cut out Sweden from Europe, and then a SE–NW chord (ala Fig. 7, right) partition Swedish airspace itself. Figure 8, left shows the two chords combined: it can be seen that the combination results in a high-degree vertex of the sectorization (at the lower side of P, around the longitude of 18 degrees) where several sector boundaries meet – an undesirable artifact. To remedy this, we note that our technique produces *all* perfectly balanced cuts – and there may be more than one (see e.g., our synthetic example in Fig. 5). Figure 8, right shows all doubly-balanced cuts of the airspace over Sweden (the part on the left of the cut in Fig. 7, left), and Figure 9 shows two of the cuts used to divide the Swedish part. The cuts were picked manually as the ones making the largest angle with the primary cut from Fig. 7, left (i.e., so that the chord at the next level of the recursion is as perpendicular as possible to the previous-level chord) – designing algorithms for automated choice of the perpendicular cuts is outside the scope of this paper.

Algorithm 1: Balanced cut

Input : A convex region P and a set S of segments
Output: A chord $\mathcal{C}$ with endpoints on ∂P

$I \leftarrow \emptyset$ ▷ Set of points on ∂P;
$C \leftarrow$ critical points of ;
foreach *pair of points* $(c_1, c_2) \in C$ **do**
  $(i, j) \leftarrow \partial P \cap$ line through c_1, c_2;
  I.append(i, j);
end
Sort I in counterclockwise order;
foreach *pair of intervals* $(I_1, I_2) \in I$: *combinatorial type of* u_1u_2 *is the same* $\forall u_1 \in I_1, u_2 \in I_2$ **do**
  $M \leftarrow$ M-imbalance(u_1, u_2);
  if $M = 0$ **then**
    if $\exists$ *A-balanced cut* u_1u_2 **then**
      return $\mathcal{C} \leftarrow u_1u_2$;
    end
  end
end

V. Conclusion

We showed how to split a sector into two parts while balancing the traffic density between the obtained sectors. Our sectorization uses only single segment as the boundary between two sectors, which provides for sectors of non-complicated (in fact, convex) shape – a useful property [7], [9], [11] (if applied recursively, in order to obtain an arbitrary number $K > 2$ of sectors, the simple 1-chord split is remi-

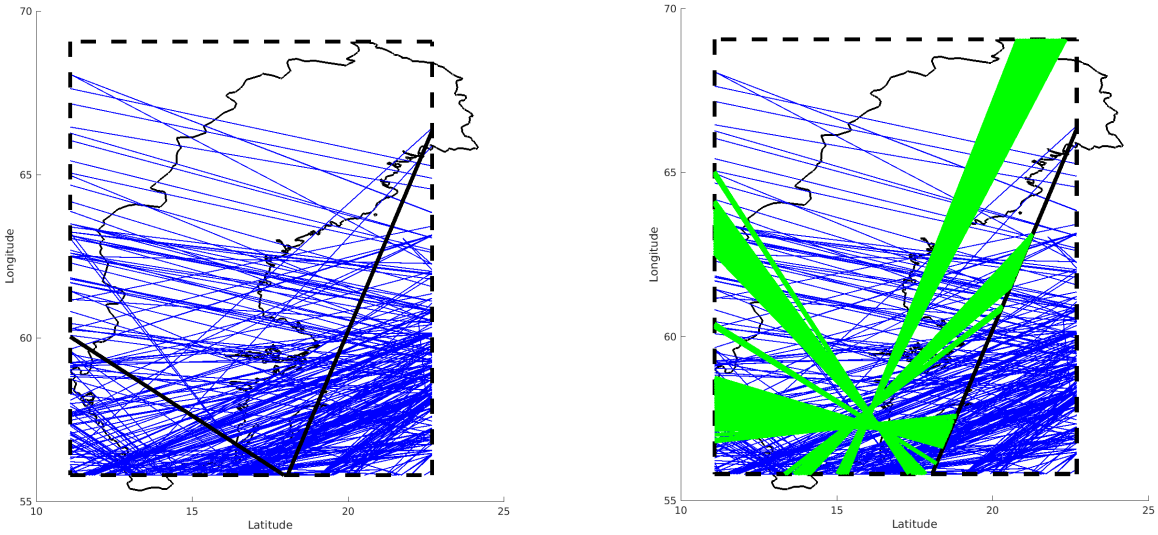

Figure 8. Left: The two cuts meet in a high-degree vertex. Right: Any cut within a green area is doubly-balanced

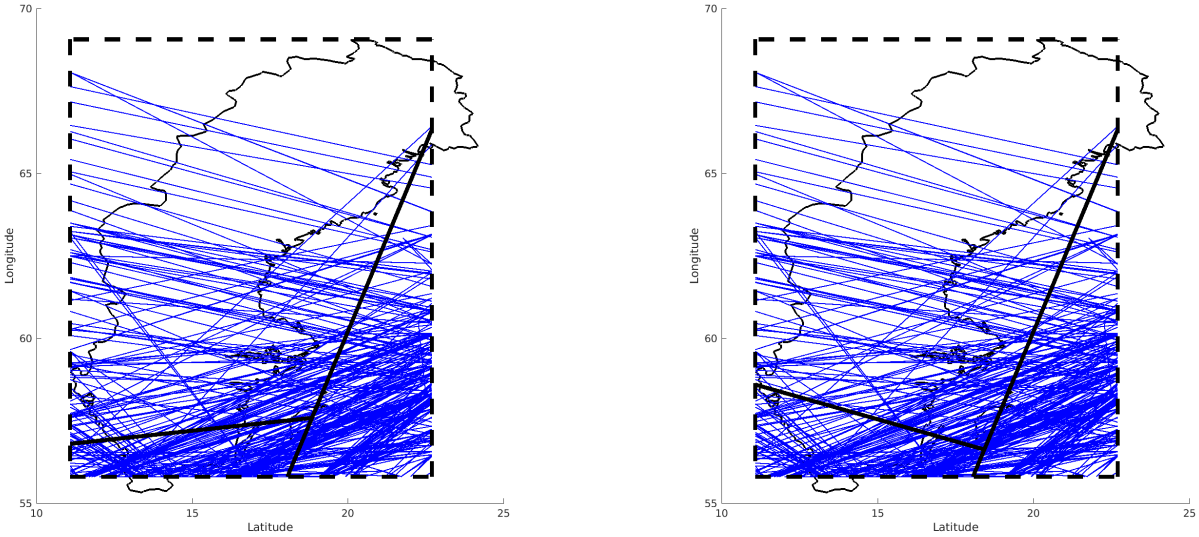

Figure 9. Possible splits

niscent of the binary space partition sectorization [6]). Simple splits are advantageous for subsequent dynamic sectorization, when the sectors are split and merged on a regular basis: we believe it will make it easier for the ATCOs to qualify to handle sectors with the simple boundary (i.e., to be certified both for split and for merged sectors).

Future work may extend our solution in many ways: (recursively) splitting into multiple sectors, using more complicated boundary between the sectors, caring about directions of the flights (we looked only at the aircraft counts), taking into account interaction between the fights and the sectors boundaries (e.g., minimizing the number of sector changes), etc. In some cases there may exist more than one sectorization that simultaneously balances the maximum and the average number of flights; in such situations it would be of interest to choose the best sectorization based on some other KPI (in fact, as our method produces *all* balanced binary splits, the subsequent choice of the best sectors may be left to the human designer). We hope that despite the relative simplicity of our approach, it can be embedded as a basic unit into more sophisticated sectorization tools.

Acknowledgements

We thank Anders Nyberg (NUAC) for discussions on the topic of the paper. This work is part of the Swedish En-route Airspace Optimization (SWEAO) project supported by the Swedish Transport Administration (Trafikverket) via the Swedish Air Navigation Service provider LFV (Luftfartsverket).

REFERENCES

[1] A. Yousefi, "Optimum airspace design with air traffic controller workload-based partitioning," Ph.D. dissertation, George Mason University, 2005.

[2] C. R. Brinton, K. Leiden, and J. Hinkey, "Airspace sectorization by dynamic density," in *Proceedings of the 9th AIAA Aviation Technology, Integration and Operations (ATIO) Forum. American Institute of Aeronautics and Astronautics*, 2009.

[3] P. Jägare, P. Flener, and J. Pearson, "Airspace sectorisation using constraint-based local search," in *ATM*, 2013.

[4] D. Delahaye, M. Schoenauer, and J. M. Alliot, "Airspace sectoring by evolutionary computation," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, May 1998, pp. 218–223.

[5] I. Gerdes, A. Temme, and M. Schultz, "Dynamic airspace sectorisation using controller task load," in *Proceedings of the SESAR Innovation Days*. EUROCONTROL, 2016.

[6] 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 (JEA)*, vol. 14, p. 3, 2009.

[7] P. Flener and J. Pearson, "Automatic airspace sectorisation: A survey," *CoRR*, vol. abs/1311.0653, 2013. [Online]. Available: http://arxiv.org/abs/1311.0653

[8] T. A. Granberg, T. Polishchuk, V. Polishchuk, and C. Schmidt, "A novel MIP-based airspace sectorization for TMAs," in *USA/Europe Air Traffic Management Research and Development Seminar (ATMSeminar)*, 2017.

[9] C. Schmidt, T. A. Granberg, T. Polishchuk, and V. Polishchuk, "Convex sectorization–A novel integer programming approach," in *Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE, 2017, pp. 1–12, 3rd Best Paper Award.

[10] M. Sergeeva, D. Delahaye, C. Mancel, L. Zerrouki, and N. Schede, "3d sectors design by genetic algorithm towards automated sectorisation," in *SESAR Innovation Days*, 2015.

[11] M. Prandini, L. Piroddi, S. Puechmorel, and S. L. Brázdilová, "Toward air traffic complexity assessment in new generation air traffic management systems," *IEEE transactions on intelligent transportation systems*, vol. 12, no. 3, pp. 809–818, 2011.

[12] I. Kostitsyna and J. S. B. Mitchell, "Local redesigning of airspace sectors," *CoRR*, vol. abs/1302.1089, 2013. [Online]. Available: http://arxiv.org/abs/1302.1089

[13] W. Hughes, "Dynamic density–a review of proposed variables," *NASA Advanced Concepts Branch*, pp. 1–12, 2000.

[14] P. Kopardekar and S. Magyarits, "Dynamic density: measuring and predicting sector complexity [atc]," in *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, vol. 1. IEEE, 2002, pp. 2C4–2C4.

[15] P. H. Kopardekar, A. Schwartz, S. Magyarits, and J. Rhodes, "Airspace complexity measurement: An air traffic control simulation analysis," *International Journal of Industrial Engineering: Theory, Applications and Practice*, vol. 16, no. 1, pp. 61–70, 2009.