

# Flow Conforming Operational Airspace Sector Design

Girishkumar R. Sabhnani\*, Arash Yousefi†

*Metron Aviation Inc., Dulles VA, 20166*

Joseph S. B. Mitchell‡

*State University of New York, Stony Brook, NY, 11794*

This is a follow-up work to Sabhnani et al.<sup>2,7</sup>. We present new algorithms for airspace sector design by extending the Binary Space Partition (BSP) method in the Computational Geometry Toolkit for Sectorization (GeoSect).<sup>2</sup> New flow conforming cuts allow sector boundaries to conform to the standard flows. We also incorporate the Air Traffic Controller (ATC) operational constraints related to sector shapes, critical flow intersection points and the Special Use Airspace (SUA) directly in the model. A simple way to sectorize in 3D using partitions in the altitude dimension is presented. Finally, we give a method for local flow conforming cuts that can be used to minimally post-process a sector design so that it satisfies the ATC operational constraints.

## Nomenclature

$\delta$	Distance of Critical Points from the Sector Boundary.
$\phi$	Angle of Intersection of Standard Flows with the Sector Boundary.
GEOSECT1.0	GEOSECT: Convex Static Sectorization.
GEOSECT2.0	GEOSECT: Flow Conforming Operational Static Sectorization.
GEOSECT3.0	GEOSECT: Shape Preserving Operational Dynamic Sectorization.
GEOSECT	Computational Geometry Toolkit for Sectorization.
ACES	Airspace Concept Evaluation System.
ATC	Air Traffic Controller.
B	Boundary Nodes.
BSP	Binary Space Partition.
c	Allowable Orientations.
E	Edges.
I	Internal Nodes.
MIP	Mixed Integer Programming.
N	Nodes.
NAS	National Airspace System.
SUA	Special Use Airspace.
ZDC	Washington ARTCC.
ZFW	Fort Worth ARTCC.
ZKC	Kansas City ARTCC.

---

\*Senior Analyst, Advanced Research & Engineering, 45300 Catalina Ct.

†Principal Analyst, Advanced Research & Engineering, 45300 Catalina Ct., AIAA Senior member.

‡Professor, Applied Mathematics and Statistics Department.

## I. Introduction

MUCH work<sup>2,4,5,10,11</sup> has been done in coming up with workload balancing sector designs (for various workload metrics). While workload balance is an important goal of re-sectorization, the validity of a sector design is also evaluated<sup>3,7</sup> based on the sector shapes and how the sector boundaries interact with the structural elements in demand like the standard flows and the critical points.

In Sabhnani et al.<sup>7</sup>, controllers pointed out the limitations in sector designs produced by GEOSECT: Convex Static Sectorization (GEOSECT1.0). While GEOSECT1.0 performed very well in balancing the workload, it did not explicitly model some of the operational constraints described by the ATC. We summarize these operational aspects below:

- Standard flows should cross the sector boundaries (almost) orthogonally. This ensures that the aircraft entering a sector along standard flows, do not graze along the boundary of the sector.
- The critical merge and conflict points must remain sufficiently inside the sector boundaries. This gives enough time to the ATC to manage the conflict (and merge) before it happens. This constraint must be a function of the average traffic velocity; the constraint will have to be greater, for example, if the crossing traffic is high altitude (faster) traffic, versus low altitude (slower) traffic.
- No more than three sectors should come together at the same point. This is because an aircraft flying near the intersection point could quickly pass between three sectors in minutes.
- The shape of the sectors must be more or less convex, with no sharp angles. This is important to avoid the re-entry of an aircraft in the same sector, once it has left that sector.

Apart from these shape and sector-demand interaction constraints, the ATC also pointed out certain important properties that the sector design should possess with respect to the SUA. A SUA is a space assigned for special purposes like military usage, *no-fly* zones around Manhattan etc. More often it is defined as a region (polygon) in 2D with a range of altitude, much like a sector definition, in which no traffic demand is allowed to enter. There are various permanent SUA's all over the National Airspace System (NAS) and sometimes SUA's are dynamically defined for a specific time window, eg. a severe weather impacted zone may be considered as a no-fly zone.

Ideally, the SUA should lie considerably inside the sector boundary. This gives sufficient time to the sector controller after a flight has entered his sector, to re-direct it around the SUA, if needed. Sometimes a sector boundary may cut across the SUA, splitting it into two regions as shown in Fig. 1. There, the requirement is that a considerable part of the SUA should lie in both sectors. Thus both controllers are aware of its existence (and approximate size), and can direct the traffic around it. In other words, the sector boundary should not clip a corner of the SUA.

Brinton et al.<sup>3</sup> also verify that algorithmic generation of sector boundaries will not provide the coordination of flight routes and sector boundaries unless metrics such as the proximity of the routes to sector boundaries and the distance from flight route crossing points to sector boundaries are explicitly considered in the sector design algorithm. In this paper, we present modifications to the heuristics in GEOSECT1.0<sup>2</sup> using flow conforming cuts that require sectors to conform to the standard flows. We incorporate the ATC operational constraints related to sector shapes, critical flow intersection points and the SUA, directly in the model. These constraints ensure that any resulting sector design is operational, from the ATC perspective. We incorporate these enhancements in GEOSECT: Flow Conforming Operational Static Sectorization (GEOSECT2.0) along with a way to partition in 3D to produce operation sectors in 3D.

The organization of paper is as follows: In Section II, we discuss the flow conforming method and the ways in which we incorporate various shape, flow and SUA constraints in the model. Section III presents the experiments for both, (a) setting the parameters and (b) creating sector designs with historical demand. In Section IV, we describe a simple extension of the BSP method to subdivide in the altitude dimension, resulting in 3D sectors. We also present a local method using flow conforming cuts to post-process sector designs produced by other sectorization methods (in particular the method of Yousefi et al.<sup>11</sup>), so as to

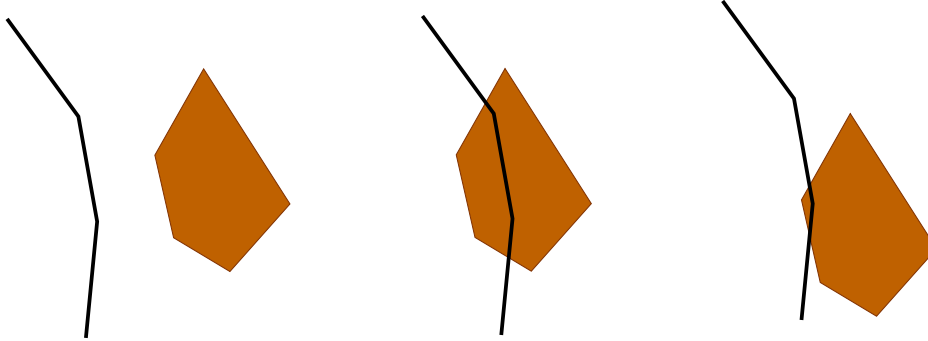


Figure 1. Desired interaction between the (brown) SUA and the (black) sector boundary. Left: SUA is considerably away from the sector boundary (desired); Middle: Sector boundary cuts through the SUA, with considerable part of SUA on each side of the cut (allowed); Right: Sector boundary clips a corner of SUA (disallowed).

minimally change the sector boundaries to make them satisfy the operational constraints, in Section V. Finally, the conclusion and future work is presented in Section VI.

## II. Flow Conforming Sector Design

In current practice, part of the demand trajectories are modified to conform to the existing sector boundaries. This will be true even after sector re-design, because no matter where the sector boundaries end up in the NAS, a fraction of the user preferred routes are bound to violate some constraints important for the controller operations. Thus, if the sector boundary is compliant with the underlying demand structure (standard flows), only a small set of aircraft will need to be re-routed.

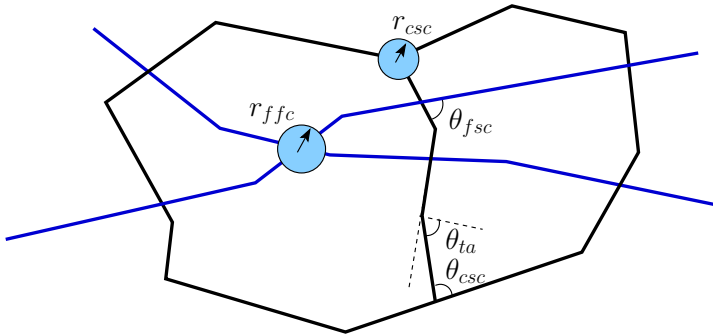
### A. Identifying Standard Flows and Critical Points

Standard flows are meant to abstract the structure in demand and highlight more traveled routes while the critical points resemble high conflict crossing and merge points. We use the greedy trajectory clustering method<sup>8</sup> to identify the standard flow and critical points. Note that one may use the existing jet routes or any other traffic abstraction method for defining standard flows in GEOSECT2.0.

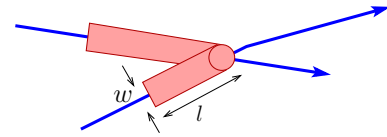
### B. Modeling Constraints

We extend the notion of *cut*, the partitioner at any level of recursion in GEOSECT1.0, from a straight line segment to more general polygonal path (with small number of bends) in GEOSECT2.0. This allows more flexibility in the cut, thus making desirable shape and flow interaction features possible. Note that this may introduce non-convexity in the sector shape, something that we control by adding suitable constraints. (In GEOSECT1.0, we started with convex hull of the region and straight line cuts at each step guaranteed convexity of the resulting subdivisions.) We model the feedback from controllers related to the various shape, flow interaction and the SUA constraints into GEOSECT2.0, as described below (see Fig. 2(a)).

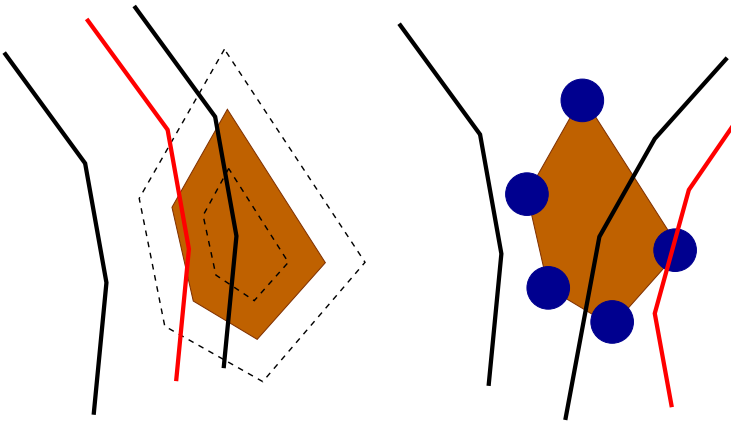
1. **Flow-Sector boundary crossing:** Standard flows should cross a sector boundary almost orthogonally. We enforce this constraint by restricting the polygonal cuts which, if intersect standard flows, have the (acute) intersection angle  $> \theta_{fsc}$ , ( $0 \leq \theta_{fsc} \leq 90$ ).
2. **Flow-Flow crossing:** In case of any crossing traffic, the width in the direction of the arriving traffic needs to be long enough (average dwell-time typically a minimum of 4-5 minutes) to resolve conflicts with other traffic. We model this requirement, by placing a rectangular obstacle of length  $l$  proportional



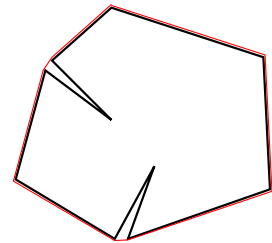
(a) Disc and angle constraints.



(b) Desired buffer from the sector boundary modeled as rectangular and disc obstacles (pink) for directional standard flows (blue).



(c) Modeling SUA Constraint: (black) sector boundaries are allowed while (red) boundaries are disallowed. Left: True modeling using inner and outer offset of the SUA boundary. Right: Simplified modeling using disc obstacles at the corners of SUA boundary.



(d) Bad case for convexity constraint: The ratio of area of (black) sector polygon and its (red) convex hull is high, even though the sector is not close to being convex.

**Figure 2. Constraints in GEOSECT2.0.**

to the average dwell-time along the arriving standard flow, as shown in Fig. 2(b). The width of this rectangle  $w$  is guided by the separation desired between flow and sector boundary. We also put a disc obstacle (of diameter  $w$ ) at the crossing point, so that it remains sufficiently inside the sector. For simplicity, the current implementation of GEOSECT2.0 only keeps a disc of radius  $r_{ffc}$  at the crossing point.

3. **Cut-Sector boundary intersections:** The angle made by a cut at the point where it meets the sector boundary should not be too acute, as the resulting sector will have sharp angle at that vertex. Like above, we force this angle to be  $> \theta_{csc}$ . Also, after we make a cut, we add a disc obstacle of radius  $r_{csc}$  at the point of intersection of cut and the region boundary. Thus no more than three sectors will meet at a vertex and no neighboring sectors share a boundary of length  $< r_{csc}$  (important for easy hand-offs).
4. **Turn-Angle at a vertex along the polygonal cut:** The turn-angle at an internal vertex of the polygonal cut should not be too acute, for same reason as in constraint 3. We add a constraint: turn-angle  $> \theta_{ta}$ , ( $0 \leq \theta_{ta} \leq 90$ ).
5. **SUA-Cut interactions:** It is easy ensure that the SUA considerably inside the sector boundary; by offsetting the sector boundary externally with the desired separation margin. But, so as to allow a sector boundary to cut across the SUA (splitting the SUA, with considerable portions on both sides of the cut), we use an inner offset of the SUA as shown in Fig. 2(c). The final constraint is, either the cut must completely avoid the outer offset polygon, or if it intersects the SUA polygon then it must also intersect the inner offset. For simplicity in the current implementation, instead of using the offsets we put disc obstacles of radius  $r_{sua}$  at all vertices of SUA boundary (or after boundary simplification, in order to have small number of vertices defining the sector boundary). This serves well for most cases.
6. **Convexity:** This constraint is intended to keep the sector shape close to convex. We constrain that the ratio of area of sector polygon to the area of its convex hull should be  $> \gamma$ , ( $0 \leq \gamma \leq 1$ ). With reasonable values for above angle constraints, this suffices to measure the convexity of sector i.e. a case like shown in Fig. 2(d) is unlikely to happen.

Cuts that satisfy the above angle constraints and avoid any rectangular/disc obstacles, would result in sector designs that address the controller concerns. The key question now is how to ensure workload balance across sectors while satisfying all these constraints.

## C. Method

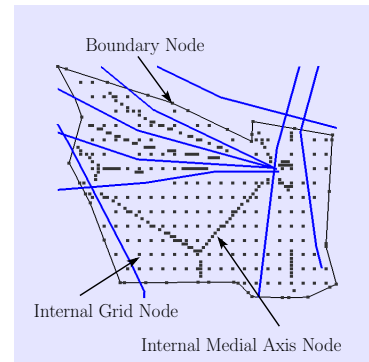
Since GEOSECT1.0 was very good at balancing workload, we try to re-use its top down sectorization methodology. The BSP heuristic from GEOSECT1.0 can be extended as follows. At each step, find a polygonal workload balancing cut that satisfies all the constraints, and recurse until desired number of sectors are obtained (or the workload of each sector decreases below some threshold). Since sectorization problem for balancing ATC workload is known to be NP-hard,<sup>6</sup> we resort to finding *good* workload balancing cuts heuristically, over a discrete search graph.

### 1. Discretizing the Search Space

We first discretize the region boundary, by placing few (parameter dependent) points uniformly spaced along the boundary. Initialize the set of Boundary Nodes (B) with these points, and add to B the original vertices of the region. Only cuts that connect a pair of these boundary nodes will be considered at any BSP recursion step. Next, we discretize the interior of the region using a discrete uniform square grid and initialize the set of Internal Nodes (I) with these points. We augment, both B and I, with points that approximate the *medial axis* of the standard flows. Start with a finer resolution (uniform) grid; the approximate medial axis points are the ones that are simultaneously close (minimum distance is same) to two or more standard flows. An instance of the discretization explained above is shown in Fig. 3. Each of the internal nodes can potentially

act as a way point on the cut, giving the cut greater flexibility for good workload balance, while satisfying the model constraints.

We build a complete Graph( $G(N,E)$ ), where Nodes ( $N$ ) =  $B \cup I$ . From  $N$ , we remove all the nodes that lie in the vicinity (parameter dependent) of a standard flow to avoid a cut from bending near a flow. Also, from the set of Edges ( $E$ ) we remove all edges that violate any disc or angle constraint. The turn-angle constraint depends on the previous edge selected in a cut, hence can only be checked up while searching for the cut. Likewise the convexity constraint depends on the final partitioned polygons, hence can only be enforced after computing the partitioned polygons. The idea is to compute multiple cuts at each level of recursion and discard those that did not satisfy the convexity constraint.



**Figure 3.** Discrete search space nodes (points) in GEOSECT2.0 (standard flows are shown in blue).

## 2. Searching for Workload Balancing Cut

After discretizing the search space, we wish to find a workload balancing cut from one boundary node to other. We start with a discrete set of Allowable Orientations ( $c$ ) for our cut. For each orientation do the following. Consider all pairs of nodes, from  $B$ , such that the straight line joining them is parallel to the current orientation (within some allowable threshold). For each pair of nodes  $i, j \in B$ , find a *min-turn-angle* depth first search path (refer Algorithm 1) satisfying the turn angle constraint, starting from  $i$  to  $j$ . We use min-turn-angle path to keep the cuts as straight as possible, thus keeping the shape of sectors more or less convex. Note that, the mta\_dfs path from  $i$  to  $j$  may differ from the path from  $j$  to  $i$ . Further, we explicitly check if the two polygons defined by this cut satisfy the convexity constraint. Finally, among all the constraint satisfying cuts, we pick the one that balances the workload best, over all orientations.

---

### Algorithm 1 mta\_dfs

---

```

Input: Start node:  $i$ , End node:  $j$ ,  $G(N,E)$ ,  $B$ .
Output: Minimum turn angle depth first search path from  $i$  to  $j$ .
 $T \leftarrow \{\}$  (Stack of nodes to visit)
 $V \leftarrow B \setminus \{i, j\}$  (Set of visited nodes)
push  $T, i$ 
while  $T$  is not empty do
   $u \leftarrow \text{pop } T$ 
   $V \leftarrow V \cup \{u\}$ 
  if  $u = j$  then
    return computed_path( $i, j$ )
  end if
   $C \leftarrow \{\}$  (List of feasible current neighbors)
  for all nodes  $v \in N$  do
    if  $v \notin V$  and  $e(u, v) \in E$  and  $e(u, v)$  satisfies  $\theta_{ta}$  then
      push  $C, v$ 
    end if
  end for
  for all nodes  $w \in C$ , in decreasing order of  $\mathcal{L}_{wuj}$  do
    push  $T, w$ 
  end for
end while
return null

```

---

In Algorithm 1, the details of *computed\_path*( $i, j$ ) are skipped, which returns the depth first search path

by using the necessary data structures. Other heuristics like Pie-Cut (and Wheel-Cut) in GEOSECT1.0 can also be extended in a similar way. Instead of finding polygonal cuts connecting two boundary nodes, a central internal node may act as the pie-center, and polygonal cuts would connect this node to boundary nodes. We have not yet implemented these heuristics in GEOSECT2.0.

### III. Experiments

The code for GEOSECT2.0 is written in C++ and uses OpenGL (glut and glui) libraries for visualization. The experiments described below were run on a machine with Intel(R) Core(TM)2 Quad CPU (Q9300 @ 2.50GHz) and 8GB ram). In GEOSECT2.0, all distances are measured as Euclidean distance directly considering (latitude,longitude) for the coordinates of points i.e. we use *flat* earth assumption for all computations. Even though an approximation, this should not have much affect on the validation of method.

Three data sets were used for most of the experiments. **Set1** (and **Set2**) consist of regions spanning 4 high-altitude sectors (currently operational in the NAS), from Kansas City ARTCC (ZKC) (and Fort Worth ARTCC (ZFW)) center. Region for **Set3** comprises of the whole of Washington ARTCC (ZDC) center. The 452 tracks for **Set1** come from simulated data, and are meant to represent 90 minutes of high traffic time window. The 1327 tracks for **Set2** (6380 for **Set3**) are actual flown trajectories for a 24 hour time period. All sets have track data for altitude range  $\geq 24k$ . See Fig. 4, for the screenshots of the data sets. The standard flows for all the sets are obtained by clustering the input trajectories using greedy trajectory clustering method.<sup>8</sup>

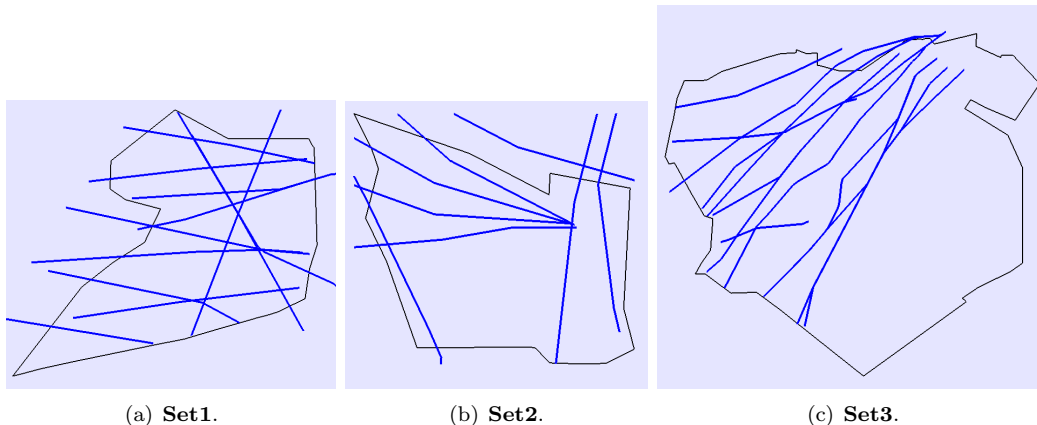


Figure 4. Data sets along with auto extracted (blue) standard flows.

We compute statistics for two new *flow-related* metrics, which quantify the *goodness* of sector boundary interactions with the standard flows. The new methods for sectorization are bound to do good with respect to these metrics, since they have been encoded as specific constraints in the model.

1. Angle of Intersection of Standard Flows with the Sector Boundary ( $\phi$ ).
2. Distance of Critical Points from the Sector Boundary ( $\delta$ ).

Note that while Brinton et al.<sup>3</sup> compute these metrics for all tracks, we restrict ourselves to evaluating these only with respect to the standard flows.

The first set of experiments include the sensitivity analysis of the workload balance to various parameters and constraints in the new model (mostly using **Set1** and **Set2**). After making intelligent choice for the parameters, experiments are conducted on **Set3** for different workload (objective) functions. For all experi-

ments, at each step of BSP recursion, the region with maximum peak aircraft count was picked for further sub-division.

### A. Sensitivity Analysis

For all sensitivity analysis experiments, the number of sectors desired for both **Set1** and **Set2** is 4. Also, the objective function is to balance the time average workload.

- **Search Space:** Size of the search space is dependent on the the number of nodes and edges in the search graph. There is a parameter  $gs$  which guides this size in `GEOSECT2.0`. In particular, points are uniformly spaced along the boundary at a distance  $total\_boundary\_length / (C \cdot 2^{gs})$  for some constant  $C$ . Similarly, the spacing between the internal (uniform square) grid points is  $[0.5 \cdot (x_{span} + y_{span})] / (C \cdot 2^{gs})$  where  $x_{span}(y_{span})$  is the x-width (y-width) of the region's bounding box in 2D. Thus, the number of search nodes at any level of recursion is proportional to  $(C \cdot 2^{gs})^2$  i.e. with every increment in  $gs$  by 1 the number of search nodes becomes (approximately) 4 times. Parameter  $gs$  is kept in the exponent to ensure that for any specific value of  $gs$ , all search nodes coming from parameter values  $gs' < gs$  are preserved.

Note that, few of the uniform search nodes may be missing from the search space, either because they lie outside the region or near a constraint. Similarly, the edges violating any constraint may also be missing. Hence, increasing  $gs$  only guarantees an approximate (multiplicative) increase in the search space.

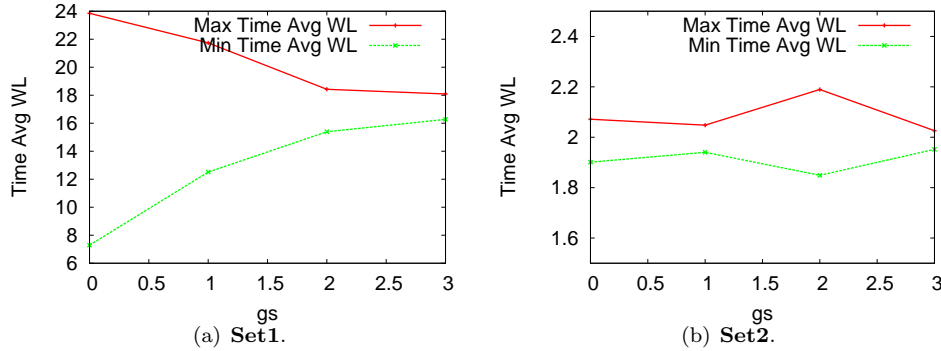


Figure 5. Search space ( $gs$ ) v/s time-avg. workload.

In Fig. 5, we show the effect of increasing  $gs$  on workload balance. Not surprisingly, the workload balance improves by increasing the size of search space, as number of choices for the cuts increases. For  $gs \geq 3$  the workload balance almost becomes perfect. Also for **Set2**, good workload balance is already achieved for  $gs$  as less as 1. This shows that one may get lucky to find good search space for small values of  $gs$ . As discussed before, a higher value for  $gs$  is unlikely to hurt the workload balance, though it does have a huge impact on the running time. For **Set2**, as  $gs$  was increased from 0 to 3, the average number of search nodes per iteration (of recursive BSP) increased from 56 to 634, the average number of feasible cuts for all orientations (8 here) increased from 28 to 367 and the running time of the experiment increased from 0.033 minutes to 2.86 minutes.

- **Discrete Orientations of Cuts:** The graph in Fig. 6 also verifies the intuition. The workload balance improves as the number of allowed orientations (hence the number of feasible cuts) increases. Note that beyond a point, when all pairs of boundary nodes are checked to find a cut, the increase in the number orientations would not increase the number of feasible cuts.

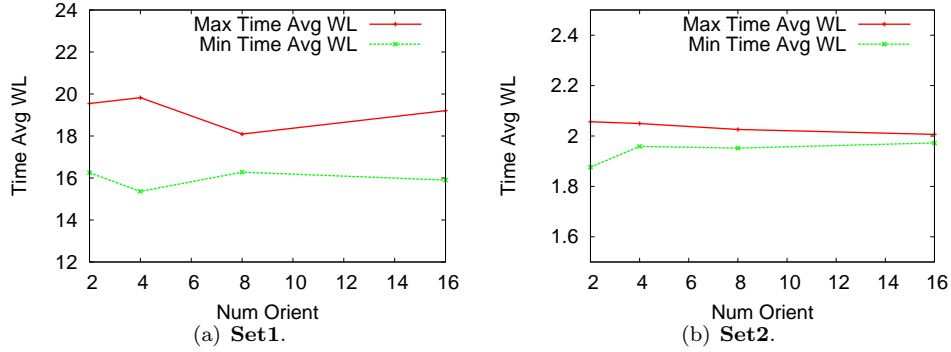


Figure 6. No. of orientations ( $c$ ) v/s time-avg. workload.

- Model Constraints:** Various constraints in the model (as described in Section IIB) can be broadly classified as angle constraints and disc constraints. The angle constraints  $\theta_{fsc}$ ,  $\theta_{csc}$  and  $\theta_{ta}$  all should ideally be close to orthogonal ( $90^\circ$ ). In Fig. 7, we show the sensitivity of workload balance to the angle constraints, as they increase from  $0^\circ$  (no constraint) to  $90^\circ$  (max constraint). All the angle constraints are increased simultaneously as it is difficult (and may be unnecessary) to classify one as more important than the other. The radius for all disc constraints were set to be 0 (no constraint). Very high values of angle constraint ( $75^\circ$  and  $85^\circ$ ) have a significant impact on the workload balance. In Fig. 8, we show the sensitivity of average(over 4 sectors) minimum  $\phi$  to the angle constraints. The average minimum  $\phi$  consistently increases with the increase in the angle constraint.

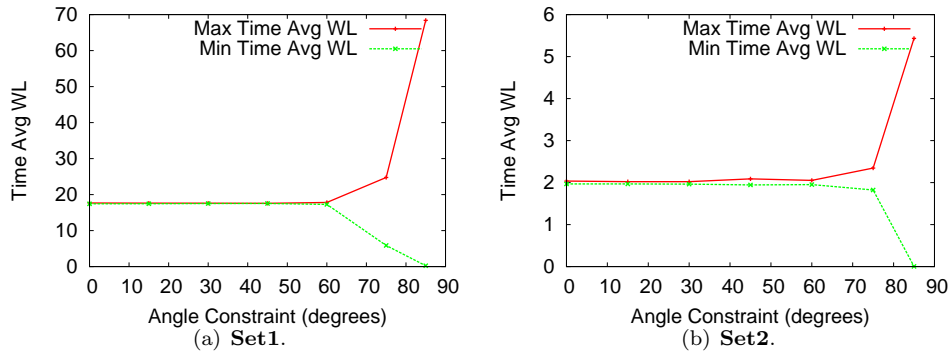


Figure 7. Angle constraint v/s time avg. workload.

The radius of two disc constraints: flow-flow intersection  $r_{ffc}$  and the cut-boundary intersection  $r_{csc}$  are also increased simultaneously and the effect on the workload balance is shown in Fig. 9. The angle constraints were all set to be  $0^\circ$  (no constraint). Even for disc constraint as high as 0.20 the workload balance is good. Note that, no SUA boundaries were used for any experiments; hence though the concept exists (and is implemented in GEOSECT2.0), sensitivity of workload balance to the  $r_{sua}$  constraint is left as future work. In Fig. 10, we show the sensitivity of average (over 4 sectors) minimum  $\delta$  to the disc constraint. Similar to angle constraint, the average minimum  $\delta$  consistently increases with the increase in disc constraint.

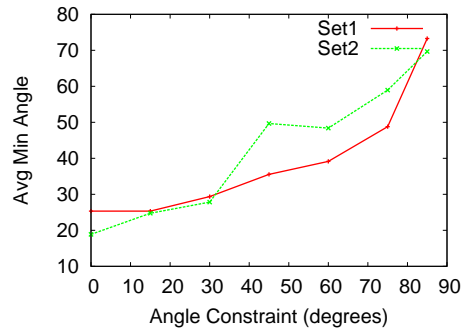


Figure 8. Angle constraint v/s avg. min  $\phi$ .

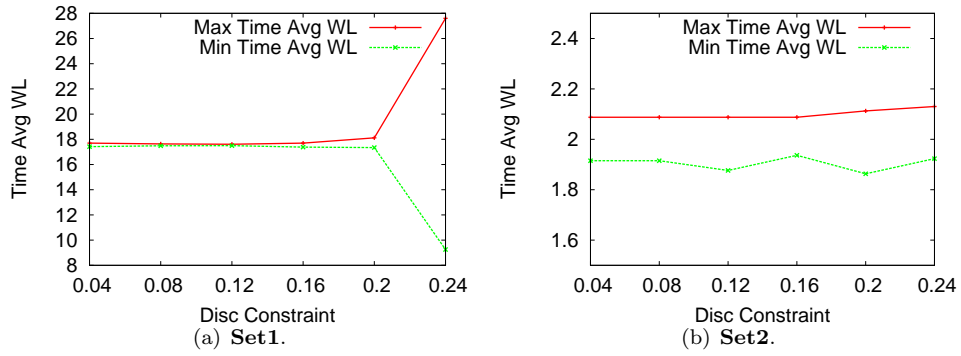


Figure 9. Disc constraint v/s time avg. workload

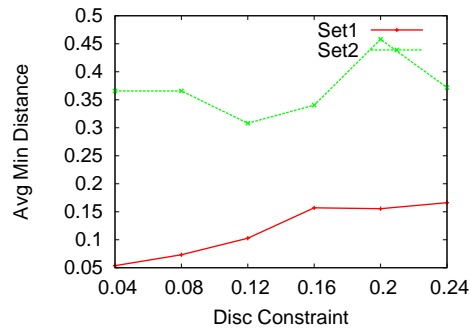


Figure 10. Disc constraint v/s avg. min  $\delta$ .

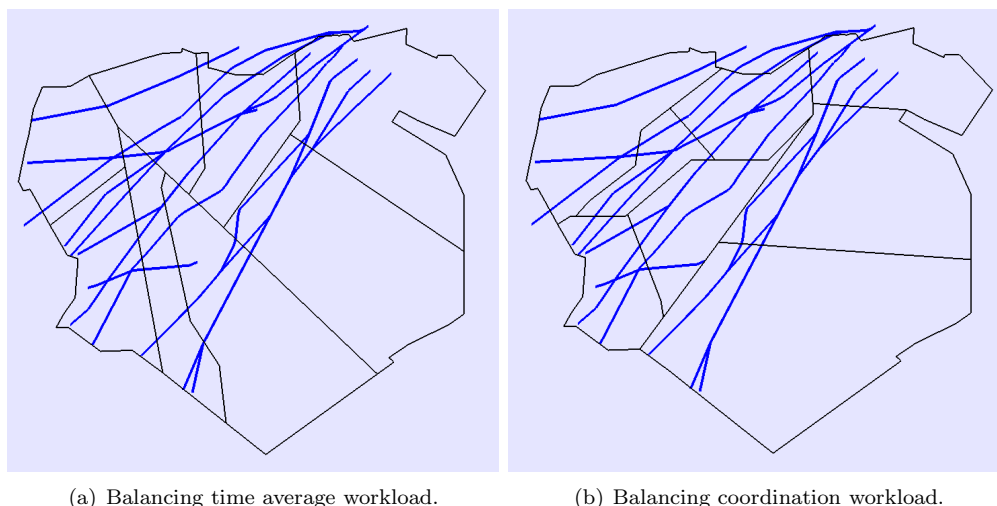
## B. Comparing different objective functions

Using sensitivity analysis results, we pick reasonable values for each constraint, and experiment with **Set3** for different workload functions. In particular, we start by looking at two extreme cases, first experiment where we balance (minimize the maximum over all sectors) time average workload, and second where we balance coordination workload. A sector with low coordination workload has less intersections (and thus more alignment) with flows. Thus, balancing coordination workload favorably increases the average dwell time of air craft within the sector.

For this set of experiments,  $gs = 1$  (this already gave sufficient graph size), the number of sectors was fixed to 8, all angle constraints were set to  $45^\circ$  and all disc constraints were set to 0.16.  $c = 16$  orientations were used, so as to allow more flexibility to find cuts that align with flows. See Table 2, for the comparison of time average workload and average dwell time for the two experiments. As desired, first sector design gave excellent workload balance, but the average dwell time is considerably less. On the other hand, the second sector design successfully increased the average dwell time, though at the cost of time average workload balance. Both the designs performed very well on the average (over 8 sectors) minimum  $\phi$  and the average minimum  $\delta$ . Refer to Fig. 11 for screenshots of the sector designs.

Sector Design	Time Avg WL		Avg. Dwell Time		Avg. Min. $\phi$	Avg. Min. $\delta$
	<i>Max</i>	<i>Std.Dev.</i>	<i>Avg.</i>	<i>Min</i>		
Time Avg WL	13.88	0.09	8.81	5.17	65.85	0.39
Coord. WL	19.18	1.39	11.35	6.33	63.37	0.38

**Table 2. Comparing workload and flow conforming metrics for two sector designs.**



**Figure 11. Results for Set3.**

Next, we try to combine the objective functions from previous two experiments so as to get good workload balance, and at the same time try to align sector boundaries with the standard flows. At each step of BSP recursion, among all feasible cuts, get the one with best coordination workload balance. Consider all the feasible cuts whose coordination workload balance is within  $x\%$  of the best coordination workload. Among these, pick the one which gives the best time average workload balance. For  $x = 0$ , it would pick the best coordination workload balancing cut every time (like in second experiment above). The graph in Fig. 12 shows the effect of increasing  $x$  on the average dwell time and the time average workload balance. It is interesting to see that as we relax the constraint (by increasing  $x$ ) on requiring the cut to balance the

coordination workload, we get improvement in workload balance. But, the monotonous decrease in the average dwell time is surprising, as there is no obvious reason for the two objective functions to compete each other. See Fig. 13 for the screenshots of sector designs for the combined objective functions.

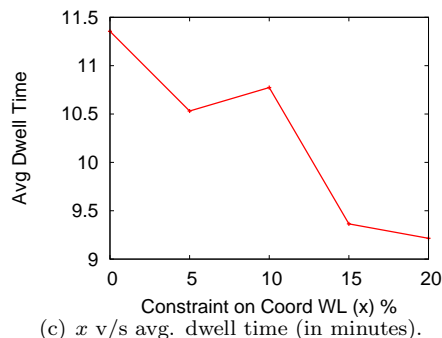
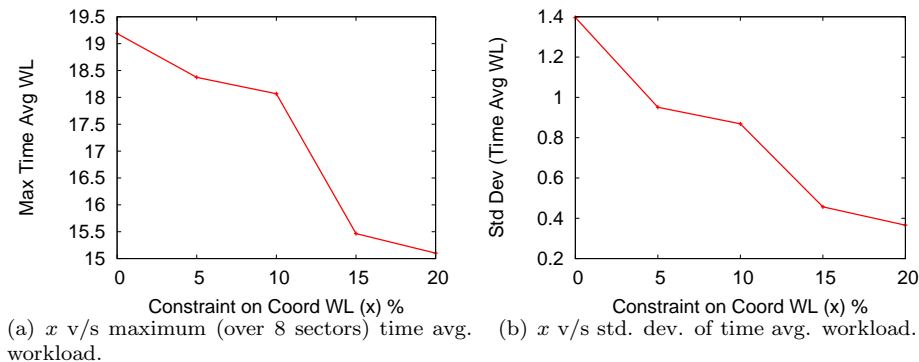


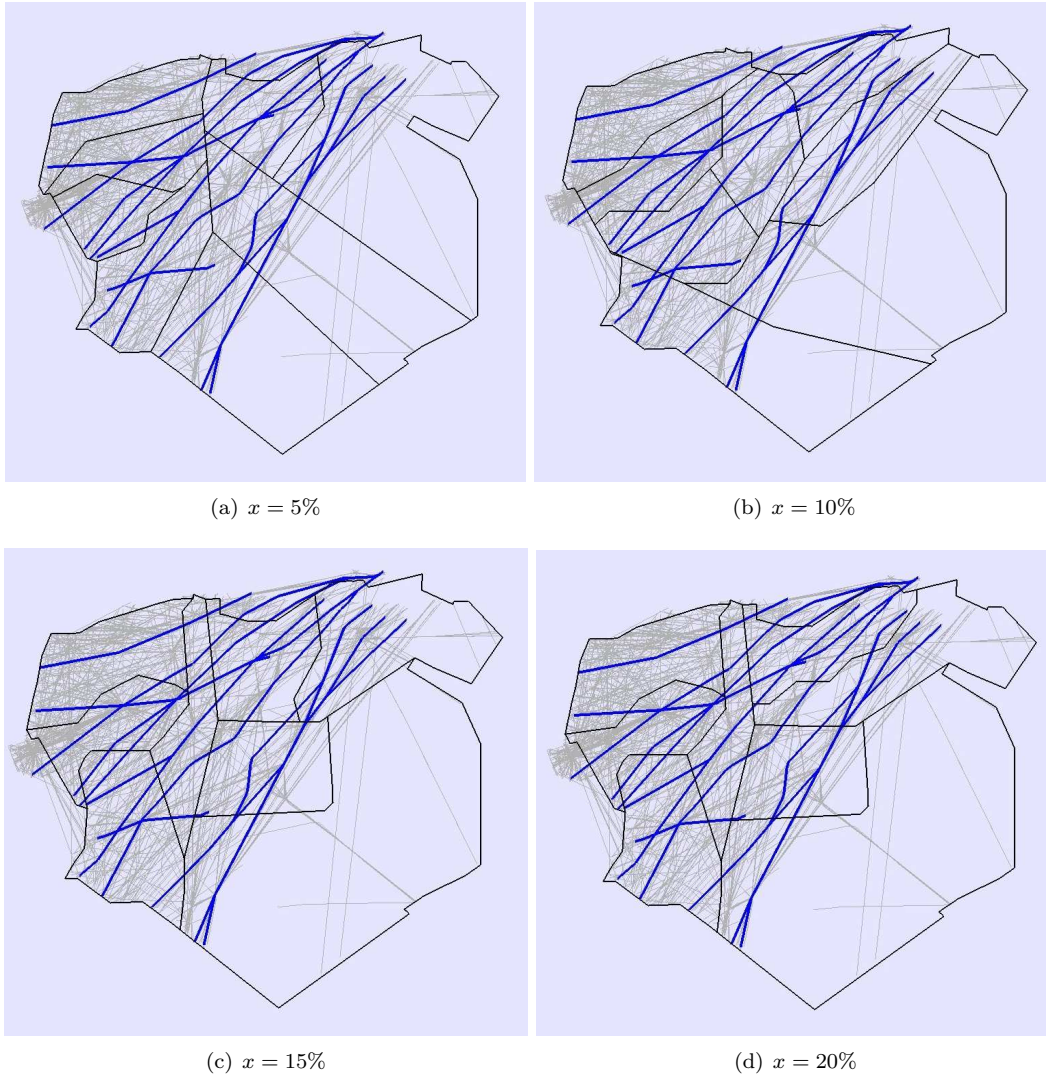
Figure 12. Effect of constraint (on the coordination workload balance)  $x$ .

We also ran experiments for the entire NAS, considering each center separately for sectorization. Air traffic for a 24 hour time period was considered, and sector designs were computed for two different altitude layers:  $24k - 35k$  and  $35k - 60k$  feet. Hand extracted standard flows were used for all the centers. See Fig. 14, for the screenshots. This experiment was run with an intermediate implementation of GEOSECT2.0; hence there were a couple of constraints missing, leading to some visual artifacts in the sector boundaries.

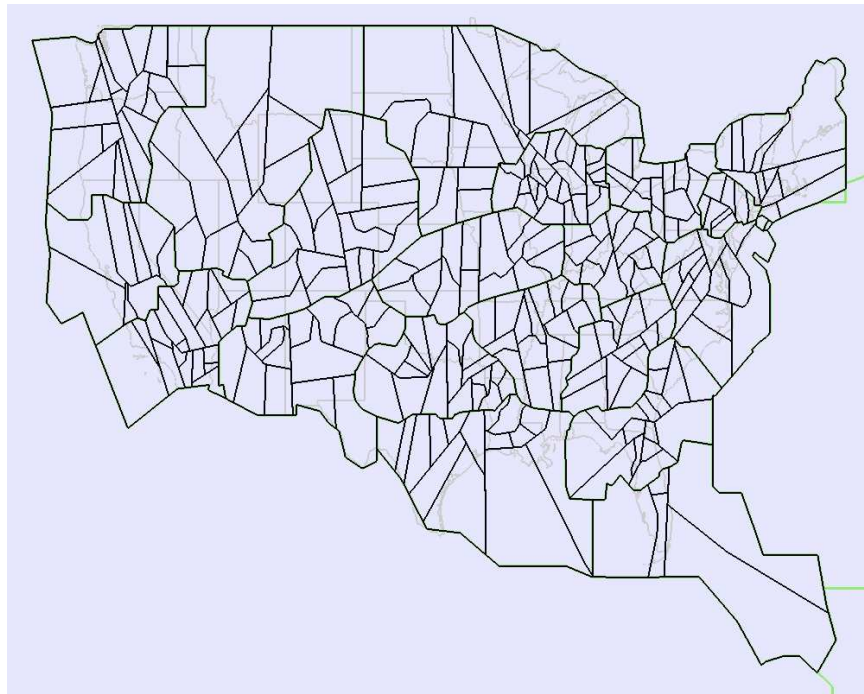
#### IV. 3D Sectorization

Today's NAS is divided into low-altitude, high-altitude and super-high altitude sectors. Each existing sector has an altitude range (min and max) which filters the air-traffic handled by the controller in that sector. Intuitively, each sector has a flat *floor* and a flat *ceiling* defining its boundary in the  $z$  (altitude) dimension. Keeping the floor and ceiling flat is important. This is partly because of the limitations of the Operational Display Systems used by the controllers, which still gives them 2D visualization of the airspace. Hence, from a controller perspective it is easy to work with a specific range of altitude assigned to a 2D sector region.

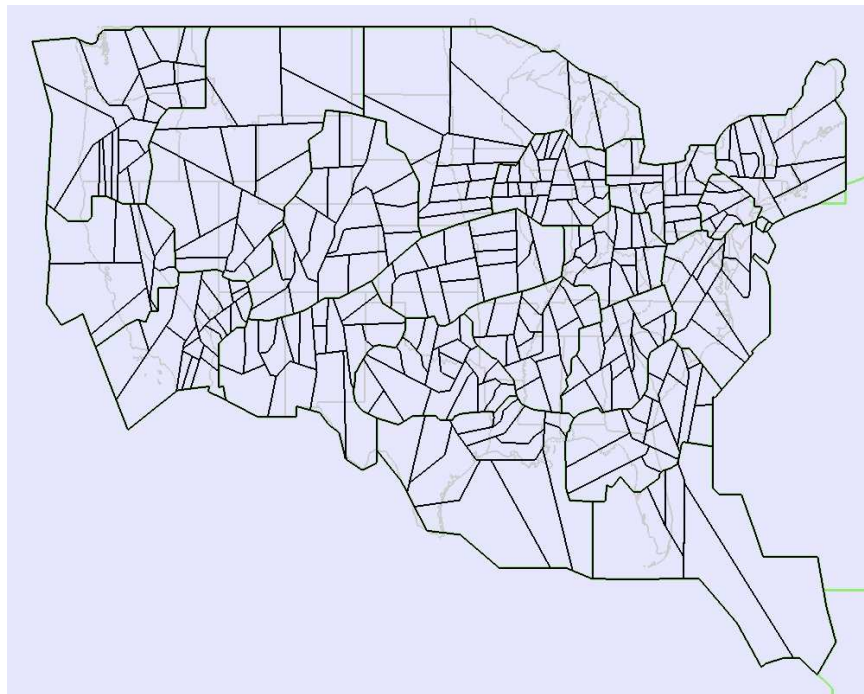
The current demand also has a lot of altitude variation. Flows moving in opposite direction are always altitude separated. Depending on the aircraft size, it may or may not have the ability to attain certain altitudes. Most important of all, flight ascends/descends (during take off/landing) add to the traffics altitude variation in the TRACON region.



**Figure 13. Screenshots of sector designs for the combined objective function.**



(a) Alt. range 24k-35k feet.



(b) Alt. range 35k-60k feet.

**Figure 14. Results for the entire NAS, one center at a time.**

## A. Method

Designing sectors in 2D is restrictive in the sense that the altitude variation in traffic is not taken into account. We describe a simple extension of the BSP method to subdivide in the altitude dimension, resulting in 3D sectors. The new heuristic for 3D sectorization is: At each level of BSP recursion, find the best  $z=\text{const.}$  cut (projecting the tracks in  $z-t$  plane as shown in Fig. 15). Compare the workload balance resulting from this cut to that resulting from a 2D BSP cut and pick the better of two. Figure 15 gives an idea, how partitioning a sector at an altitude level might help segregate the demand. Effectively it splits high altitude traffic from low altitude, thereby assigning workload of different altitudes to different controllers.

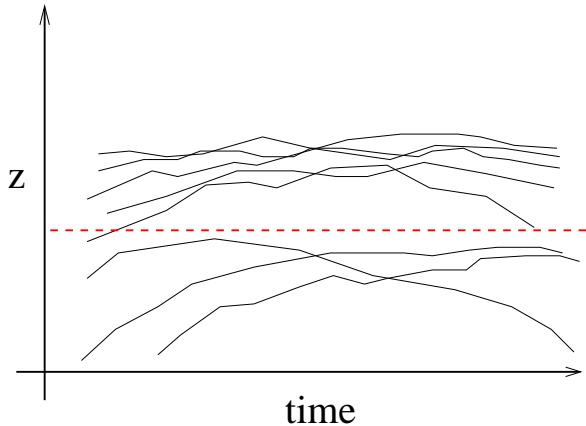


Figure 15.  $z=\text{const.}$  cut splitting high alt. traffic from low alt. traffic

## B. Experimental Results

We experiment with **Set3** (for 8 sectors balancing the peak aircraft count), and compare the results for 2D and 3D sector designs. We do not use any standard flows for this experiment. This is because, the current capability in GEOSCT2.0 has flows defined only in 2D, and not all flows are relevant at all altitude levels. See Fig. 16, for the screenshot of sectors produced by the 3D method. In particular, one intermediate region was divided at altitude level  $34.5k$ .

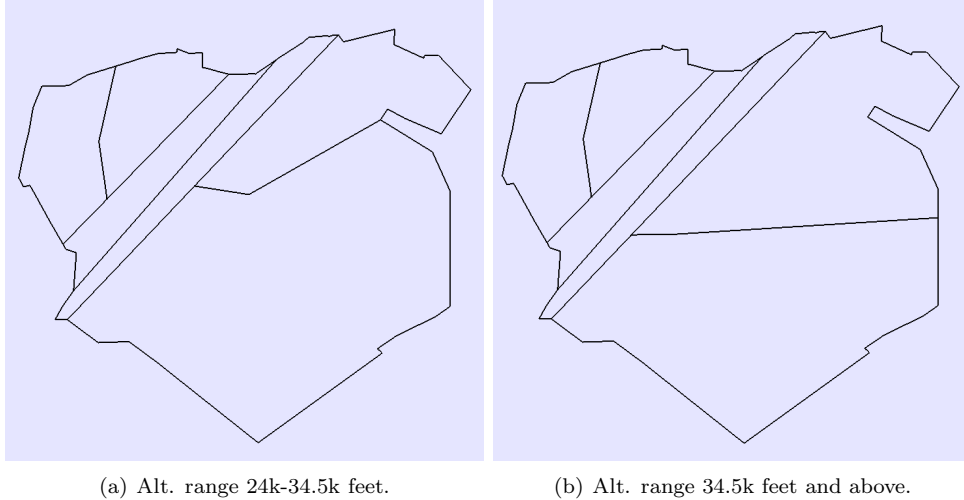
Refer to Table 3, for comparison of workloads for 2D and 3D sector designs. Even though the 3D method marginally improved the max peak aircraft count, there is not much difference between the workload balance of the two methods. This may be because of the en-route airspace that we were trying to sectorize. The 3D capability may be useful, when we model experiments for TRACON region, where there is a lot of ascending and descending traffic. More experiments validating the 3D sectorization method are underway.

Sectorization	Peak Aircraft Count			Time Avg WL		
	<i>Max</i>	<i>Min</i>	<i>Std.Dev.</i>	<i>Max</i>	<i>Min</i>	<i>Std.Dev.</i>
2D	37	33	0.42	14.30	12.36	0.22
3D	36	31	0.62	14.85	11.69	0.37

Table 3. Comparing results of 2D and 3D sector designs.

## V. Local Post-processing Sector Designs

In this section, we describe a local post-processing method, that can take input an existing sector design and the demand and modify the sector design minimally to ensure that the operational constraints



**Figure 16. Results for 3D sectorization of Set3.**

are satisfied. In particular, we work with the Mixed Integer Programming (MIP) sectorization method of Yousefi<sup>11</sup>. The MIP sectorization aims at clustering a hex cell partitioning of airspace for minimizing sector boundary crossings and balancing the overall workload across sectors. It is very powerful in optimizing the objective function, however it does not consider geometric requirements of sector polygons. Also since the resulting sector polygons are clusters of hex-cells the sector boundary is not very simple and hence not operational. The flow conforming cuts in GEOSECT2.0 are primarily good in maintaining these shape and other operational aspects of sector design.

### A. Method

The main idea is to use the flow conforming cut as a local re-optimization step for re-sectorizing a set of neighbouring MIP sectors. The current implementation uses 2-to-2 re-optimization technique of GeoSect. In particular, this means erasing the boundary between two adjacent MIP sectors and re-computing it using flow-conforming-cut search routine from GeoSect. The advantage of this approach is that we incorporate the sector shape and flow interaction constraints directly into the model and hence resulting sector design is more operationally feasible.

The description of 2-to-2 re-optimization method is as follows (refer to Fig. 17) For all adjacent MIP sector pairs do:

- Erase the boundary between two neighbouring sectors.
- Generate discrete search nodes near (defined by parameter) the original MIP sector boundary. This restricts the new GeoSect cut to stay near the original boundary.
- Find multiple GeoSect cuts using discrete search nodes, respecting sector shape and flow constraints.
- Select a cut that is closest to the original boundary (in order to use the optimization result) and replace the original boundary with this cut. Alternately, we can select a cut that optimizes some workload metric.

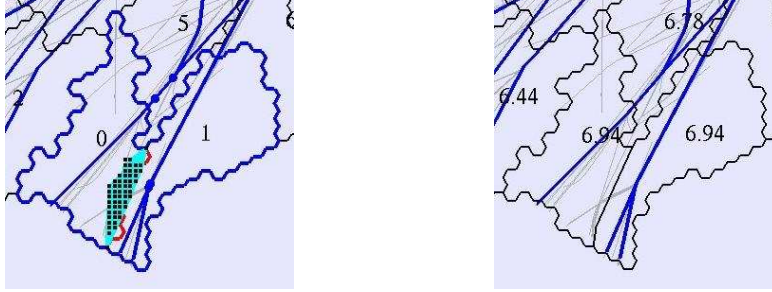


Figure 17. Left: The common region (blue) after merging two neighboring sectors. The discrete search graph (black nodes and magenta edges) for GeoSect to find a new cut. Right: The new GeoSect cut replacing the original MIP boundary.

## B. Experimental Results

We present the preliminary results of GeoSect post-processing of MIP solution for 16 sectors in ZDC. Track data for 24 hours and altitude range 24K feet and above is used and comes from Airspace Concept Evaluation System (ACES) simulation for sector comparisons in Zelinski et al.<sup>12</sup>. All angle constraints ( $\theta_{fsc}, \theta_{csc}, \theta_{ta}$ ) in GeoSect were 45 degrees, and the radius of all disk constraints ( $r_{ffc}, r_{csc}$ ) was 0.05 degrees (approx. 2.5 nautical miles). The convexity constraint was set to be 0.6. As seen in Fig. 18, for most part, the post-processing seems to take care of the sector shape and flow conforming artifacts in the MIP solution.

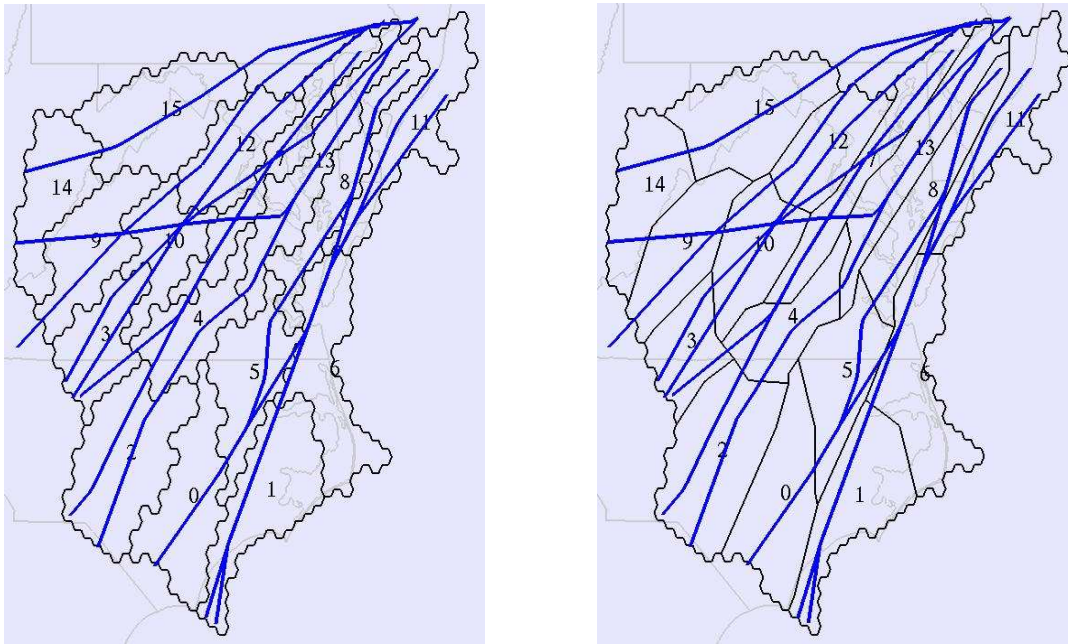


Figure 18. Left: MIP sectors for ZDC. Right: GeoSect2.0 post-processing results. (Standard flows are show in blue.)

In Fig. 19, we compare the effect of post-processing on workload balance (toatal tz-hit count), average dwell time and convexity of each sector. While the workload balance worsened a little (23.4% increase in the tz-hit count for sector 10), the average dwell time interestingly improved for many sectors. The improvement

in convexity of each sector is of course notable. Table 4 gives the summary of comparison of these metrics. We notice that the average over all sectors of average dwell-time increased and so did the convexity, while the standard deviation of the workload balance worsened slightly.

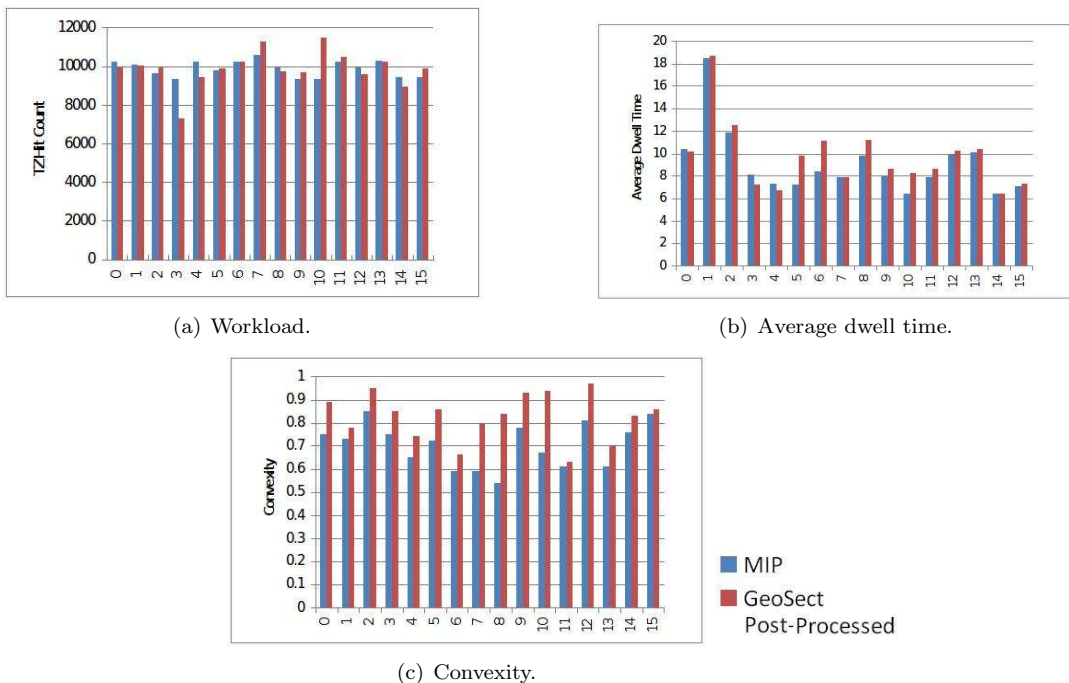


Figure 19. One-to-one comparison of sectors before and after processing.

Sectorization	Workload (tz-hit count)			Avg. Dwell Time			Convexity	
	<i>Min</i>	<i>Max</i>	<i>Std.Dev.</i>	<i>Min</i>	<i>Max</i>	<i>Avg</i>	<i>Min</i>	<i>Avg</i>
MIP	9304	10613	419	6.38	18.48	9.07	0.54	0.70
GEOSECT Post-Processing	7297	11487	937	6.40	18.67	9.69	0.63	0.83

Table 4. Summary of metric comparison before and after post-processing.

## VI. Conclusion and Future Work

We presented a new sectorization method using flow conforming cuts that ensure certain shape, flow and SUA constraints that are important for the operational validity of a sector design. Experimental results indicate that it is possible to balance the workload while satisfying the operational constraints. A method for 3D sectorization using altitude cuts was also presented. 3D method did not have significant improvement over 2D method in the en-route airspace, but it might be useful for sectorization in the TRACON region. The results of local method for post-processing MIP sectorization look promising, as the desired shape and flow constraints are satisfied at a reasonably small cost of slight workload imbalance. It should be clear that this method of post-processing applies to any sectorization (either existing or generated by any other method).

Note that, a GEOSECT2.0 cut that satisfies all the defined constraints may not exist. At that point, the options are to either insert more search nodes (for more flexibility) or relax few constraints (in an order defined by the importance of each constraint). In future, we plan to do more experiments in 2D so as

to develop an automatic way of relaxing some of these constraints. Then the method would automatically select good parameters depending on the input demand data. As another objective function, we would like to directly maximize sector capacities (either measured as 5/3 avg. dwell time according to the FAA standard<sup>1</sup> or by using the method of Welch<sup>9</sup>). One way to achieve this is to minimize the max difference between the sector capacity and the peak aircraft count. The intuition for this objective function is to provide maximum capacity in places where it is needed by the input demand.

For the 3D method, we intend to use the flow definitions in 3D along with 3D definition of the model constraints. We also intend to extend the local method by merging and re-sectorizing local regions of airspace with workload imbalance (due to the dynamically changing demand) for dynamic sectorization (in GEOSCT: Shape Preserving Operational Dynamic Sectorization (GEOSCT3.0)).

## Acknowledgments

This work was supported under NextGen-Airspace Project of NASA's Airspace Systems Program (Order No. NNA07BB33C). We thank Gregory L. Wong at NASA Ames Research Center and Dr. Robert Hoffman at Metron Aviation Inc. for helpful discussions.

## References

- <sup>1</sup>Federal Aviation Administration, Section 8. Monitor Alert Parameter, Facility Operation and Administration, US DOT Order JO 7210.3V, Ch. 17, [http://www.faa.gov/air\\_traffic/publications/ATpubs/FAC/Ch17/s1708.html](http://www.faa.gov/air_traffic/publications/ATpubs/FAC/Ch17/s1708.html), 2008.
- <sup>2</sup>A. Basu, J. S. B. Mitchell, and G. R. Sabhnani. Geometric algorithms for optimal airspace design and air traffic controller workload balancing. *J. Exp. Algorithmics*, 14:2.3–2.28, 2009.
- <sup>3</sup>C. R. Brinton and L. S. Cook. Analysis of current airspace operations and implications for dynamic airspace configuration. In *AIAA Guidance, Navigation, and Control Conference*, Aug 2008.
- <sup>4</sup>C. R. Brinton and S. Pledgie. Airspace partitioning using flight clustering and computational geometry. In *27th Digital Avionics System Conference*, 2008.
- <sup>5</sup>M. Drew. Analysis of an optimal sector design method. In *27th Digital Avionics System Conference*, 2008.
- <sup>6</sup>A. H. Farrahi and Z. Wood. Computational complexity of the airspace sectorization problem. Personal Communication, Feb 2009.
- <sup>7</sup>J. S. B. Mitchell, G. Sabhnani, J. Krozel, B. Hoffman, and A. Yousefi. Dynamic airspace configuration management based on computational geometry techniques. In *AIAA Guidance, Navigation, and Control Conference*, Aug 2008.
- <sup>8</sup>G. R. Sabhnani, A. Yousefi, D. Kierstead, I. Kostitsyna, J. S. B. Mitchell, and V. Polishchuk. Algorithmic traffic abstraction and its application to nextgen generic airspace. Draft submitted, 2010.
- <sup>9</sup>J. Welch. Macroscopic workload model for estimating en route sector capacity.
- <sup>10</sup>M. Xue. Airspace sector redesign based on voronoi diagrams. In *AIAA Guidance, Navigation, and Control Conference*, Aug 2008.
- <sup>11</sup>A. Yousefi. *Optimum Airspace Design with Air Traffic Controller Workload-Based Partitioning*. PhD thesis, George Mason University, 2005.
- <sup>12</sup>S. Zelinski. A comparison of algorithm generated sectorizations. In *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)*, June-July 2009.