

# Geometric Algorithms for Optimal Airspace Design and Air Traffic Controller Workload Balancing

Amitabh Basu

Joseph S. B. Mitchell

Girishkumar Sabhnani

## 1 Introduction

The sectorization problem is to determine a decomposition of a given domain  $\mathcal{D}$  of airspace into a set of  $k$  sectors,  $\sigma_1, \dots, \sigma_k$ , in an “optimal” manner. Optimality is defined in terms of the *workloads*,  $w(\sigma_i)$ , of the sectors, where  $w(\sigma_i)$  is a numerical value indicating the amount of “effort” required to manage and control traffic in sector  $\sigma_i$ . The objective may be to minimize the maximum workload (min-max) or to minimize the average workload (min-avg) across sectors, subject to an upper bound,  $k$ , on the number of sectors. Alternatively minimize  $k$  subject to a bound on the maximum or average workload across sectors.

We model the problem using a geometric and easily quantified approach to defining sector workload. The historical track data is assumed to be given. What makes our sectorization problem novel compared with most geometric load balancing problems previously studied is that the input data consists of *trajectories of moving points*.

## 2 The Sectorization Problem

### 2.1 One Dimension

Consider an airspace domain that is 1-dimensional, consisting of an interval, without loss of generality  $\mathcal{D} = [0, 1]$ , on the  $x$ -axis. Flights can take off at some point (“airport”) of  $\mathcal{D}$  and land at another point (“airport”).

The input data consists of a set  $S$  of flight trajectories, each represented by a sequence of “waypoints”,  $(x_i, t_i)$ , where  $t_i$  is the timestamp when the flight is recorded to be at location  $x_i \in [0, 1]$ . We consider there to be a finite time horizon,  $[0, T]$ , containing all of the timestamps  $t_i$ . We generally assume that the flight speed between waypoints is constant; thus, a trajectory can be thought of as a  $t$ -monotone polygonal chain in the  $(x, t)$ -plane. For now, we consider the input  $S = \{s_1, \dots, s_n\}$  to be a set of line segments in the  $(x, t)$ -plane, all of which lie within the 1-by- $T$  rectangle,  $[0, 1] \times [0, T]$ .

The sectorization problem asks us to partition  $[0, 1]$  into a set of  $k$  sectors,  $\sigma_1, \sigma_2, \dots, \sigma_k$ ; i.e., we desire partition points,  $x_0 = 0 < x_1 < x_2 < \dots < x_{k-1} < x_k = 1$ , which define the sector intervals  $\sigma_i = (x_{i-1}, x_i)$ .

The *max-workload*,  $w(\sigma_i)$ , of a sector  $\sigma_i = (x_{i-1}, x_i)$  is defined to be the maximum number of flights ever simultaneously in sector  $\sigma_i$ : this is given geometrically by the maximum number of segments of  $S$  intersected by a horizontal segment,

$\overline{(x_{i-1}, t)(x_i, t)}$ , for  $t \in [0, T]$ . The *avg-workload*,  $\bar{w}(\sigma_i)$ , of a sector  $\sigma_i = (x_{i-1}, x_i)$  is defined to be the *time-average* number of flights in the sector  $\sigma_i$ : this is given geometrically by the sum of the lengths of the  $t$ -projections of segments  $S$  clipped to the rectangle  $[x_{i-1}, x_i] \times [0, T]$ . If we let  $\xi_i(t)$  denote the number of segments of  $S$  crossed by the horizontal segment  $\overline{(x_{i-1}, t)(x_i, t)}$ , then  $w(\sigma_i) = \max_{t \in [0, T]} \xi_i(t)$  and  $\bar{w}(\sigma_i) = \frac{1}{T} \int_0^T \xi_i(t) dt$ .

The *min- $k$  sectorization problem* is to determine a set of partition points  $x_i$  (and corresponding sectors  $\sigma_i$ ) in order to minimize the number,  $k$ , of sectors in a partitioning of  $[0, 1]$ , subject to a specified *workload bound*,  $B$ . The workload bound  $B$  stipulates that  $w(\sigma_i) \leq B$ , or that  $\bar{w}(\sigma_i) \leq B$ , for all  $i = 1, \dots, k$ , in the max-workload or the avg-workload case, respectively.

The *min- $B$  sectorization problem* is to determine a set of partition points  $x_i$  (and corresponding sectors  $\sigma_i$ ) in order to minimize the upper bound,  $B$ , on the workloads of the sectors, subject to their being at most (and therefore exactly)  $k$  sectors, where  $k$  is specified as part of the input. In other words, we want to determine the  $x_i$ 's,  $i = 1, \dots, k$ , subject to  $w(\sigma_i) \leq B$ , or  $\bar{w}(\sigma_i) \leq B$ , for all  $i = 1, \dots, k$ , in the max-workload or the avg-workload case, respectively.

**Theorem 1** *The one-dimensional min- $k$ , max-workload, sectorization problem can be solved exactly in time  $O(kn \log^2 n)$ , where  $k$  is the output optimal number of sectors.*

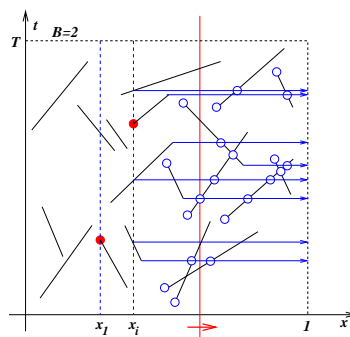


Figure 1: An instance of 1-d problem.

**Theorem 2** *The one-dimensional min- $B$ , max-workload, sectorization problem can be solved exactly in time  $O(kn \log^3 n)$ .*

### 2.2 Two Dimensions

The input data consists of a set  $S$  of trajectories, which correspond to  $t$ -monotone polygonal chains in  $(x, y, t)$ -space. We let  $n$  denote the number of trajectories, and  $N$  the total number

of waypoints (vertices) in the full set of  $n$  trajectories. Given a domain of interest,  $\mathcal{D} \subset \mathbb{R}^2$ , we are to partition it into a small number of sectors, each of which has a small workload.

The max-workload for a sector  $\sigma \subset \mathcal{D}$  is the maximum number of trajectories intersected by a “horizontal” (in  $t$ ) polygon of shape  $\sigma$ , sweeping vertically through time,  $t \in [0, T]$ .

### 2.2.1 Hardness

**Theorem 3** *The optimal sectorization problem (min-k or min-B) for rectangular partitions into sectors in two dimensions is NP-hard.*

## 3 Heuristics for 2D Sectorization

### 3.1 BSP Heuristic

BSP heuristic is based on computing a *most balanced cut* at each stage, which is defined as follows. Given a node of the current BSP subdivision, with associated sector  $\sigma$ , our algorithm finds a straight cut (from among a set of fixed orientations) to partition the convex polygon  $\sigma$  into two subpolygons, in order to minimize the maximum workload (either max-workload or avg-workload) of the two subpolygons. In order to find the most balanced cut, we use a discrete set of  $\ell$  allowable orientations for our cut. For each orientation, we find the most balanced cut with that orientation as follows. We project the line segments that make up the trajectories onto a plane perpendicular to the cut orientation, resulting in the 1D problem.

The balanced BSP heuristic can clearly produce very skinny sectors, even while producing sectors with well-balanced workloads. Skinny sectors can be undesirable because air traffic passing through the sector may be in the sector for radically different time periods, depending on the orientation of the trajectory with the diameter of the sector. We propose ways to avoid bad aspect ratios by selecting only from the cuts which respect some aspect ratio thresholds.

### 3.2 Pie Cutting

In addition to the BSP cuts, we consider another cutting operation to allow for more flexibility during sectorisation. This is the so-called “pie-cut”. For this we fix a point within the region (called the *center*) and an orientation. We now wish to make a pie-cut which comprises three rays originating from the *center*. One of the rays is along the designated orientation. The other two are such that the resulting 3 pieces are all convex and as well-balanced as possible. We accomplish this pie-cut in two steps, obtaining one cut in each step. The line segments are first transformed to their polar coordinates, in the following sense. Consider any point  $p$  with polar coordinates  $(r, \theta, z)$  with respect to the *center* and the given orientation ( $r$  is the distance from the *center*,  $\theta$  is the angle which the line through  $p$  and the *center* makes with the given orientation. This point is transformed to  $(\theta, z)$ , resulting again in an instance of the 1-D sectorization problem. Then a cut is found which divides the workload in 1 : 2 ratio. Then the second cut is chosen with range restrictions (so that the resulting regions are convex) to balance the workload in the  $\frac{2}{3}$ -sized region.

## 3.3 The Final Heuristic

We formulate a method using both the operations of BSPs and Pie-cuts. The final heuristic first attempts to make a possible pie-cut. If the pie-cut is unable to find a partition respecting the  $\beta$  threshold, we use a BSP cut. The new regions are then inserted into a priority-queue according to the workloads. We recurse on the heaviest region until all the regions have a workload less than  $B$ .

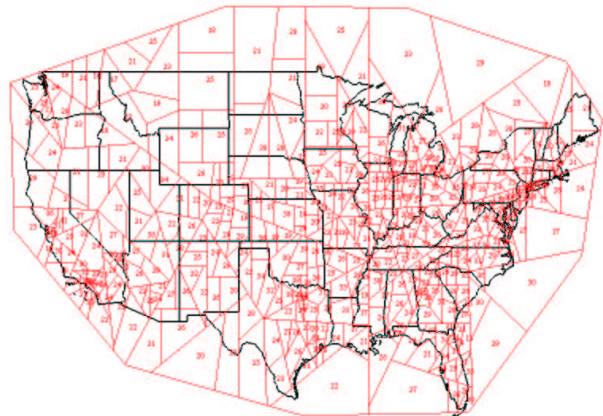
## 4 Experimental Setup and Results

Implementation of our system GEOSECT is done in C++ (Microsoft Visual Studio 6.0), using the OpenGL Graphics library for all visualizations. All experiments were run on machines with 3.2 GHz Pentium 4 processors, 1GB RAM.

Data is provided to us by Metron Aviation. The historical track data corresponds to approx 24hr period from 04:00, June 27 to 05:00, June 28 2002 with 74588 flight tracks and the average complexity of each track (number of bends) being 59.26. We use existing sector data to compare our results.

We built a competitive heuristic, by running experiments and tuning the different parameters like number of discrete orientations for the bsp cuts and the thresholds for avoiding skinny sectors. Finally we compared our results with the sectors currently used in NAS. In particular we ran experiments to sectorize a large convex polygonal region,  $U$  containing all of the continental USA. While taking the statistics of the original sectors, we consider only the sectors which are completely inside the selected region of interest for partition. Also, the partitioning looks to balance the time-average workload.

Summarizing the results, both the final heuristic and the BSP give nicely balanced sectors with good non-skinny shapes. The standard deviation and max values of worstcase and time-average workloads improve by a factor of 2-3, while using essentially the same number of sectors. One of the screenshots of the result can be seen in figure 2.



**Figure 2:** Final Heuristic results sectorizing the entire US region