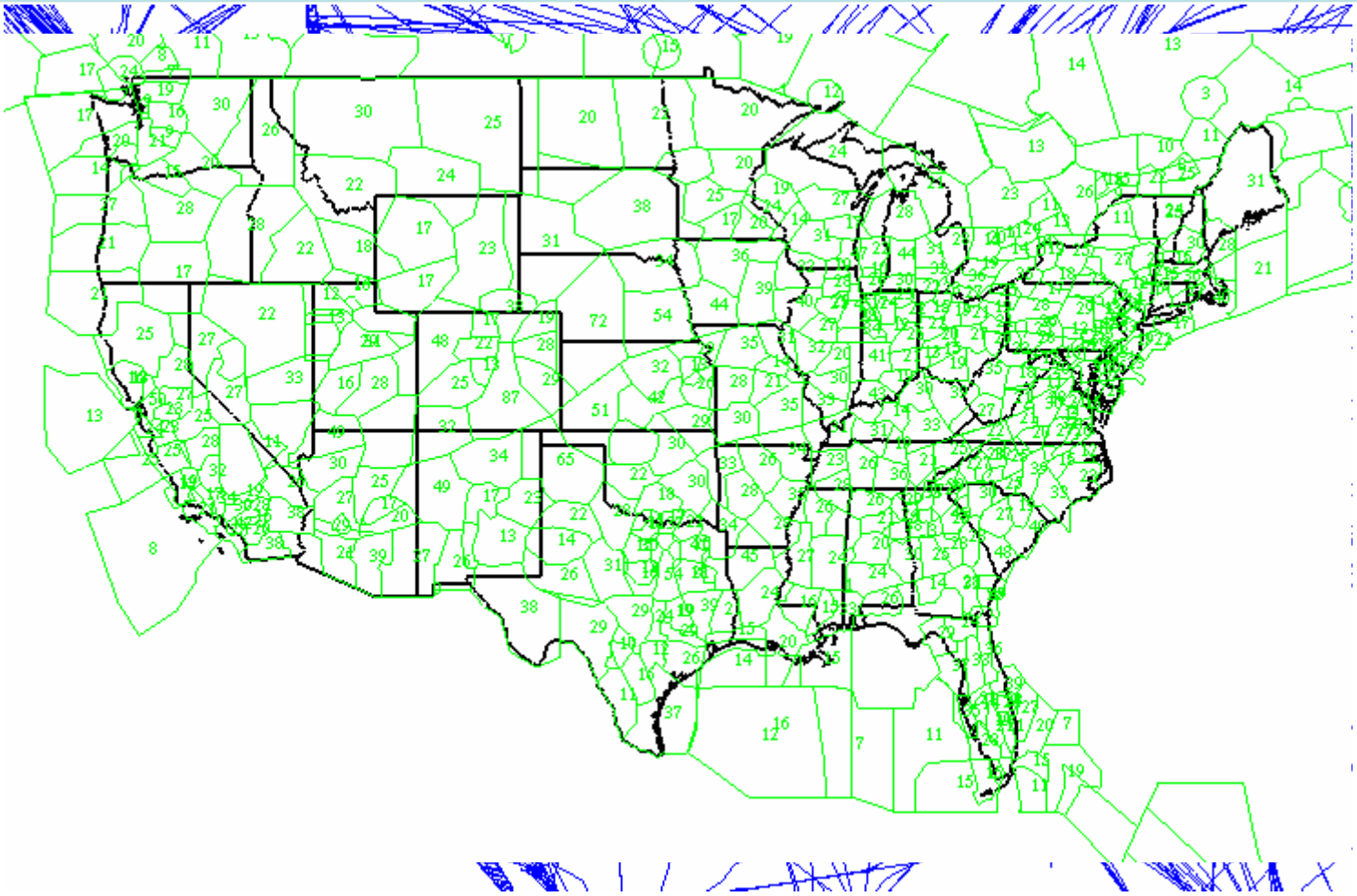
Computational Geometry for DAC: GeoSect Methodology

Joseph S. B. Mitchell and Girishkumar Sabhnani
Stony Brook University

STONY BROOK
STATE UNIVERSITY OF NEW YORK

Airspace Sectorization Problem



Joint work with A. Basu and G. Sabhnani (ALENEX'08)

Objectives

- Design and implement efficient algorithms to compute optimal (or nearly-optimal) airspace configurations
- Devise novel methods that may assist in maximizing safe utilization of airspace
- Explore future concepts of operations

"Provide flexibility where possible and structure where necessary."

Parimal Kopardekar (NASA Ames)

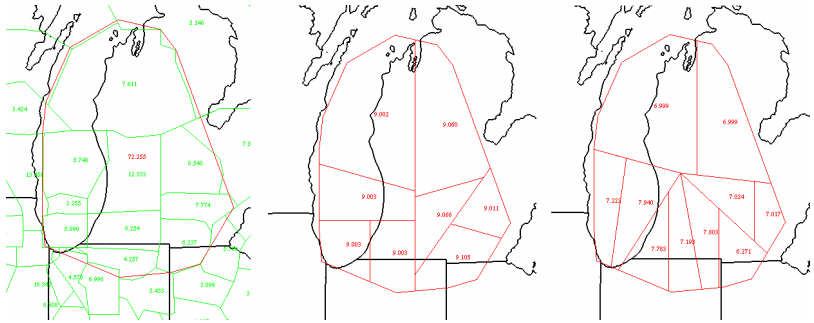
NASA: Dynamic Airspace Configuration (DAC) project

Design for Control

- Determine a mapping of controllers (or oversight processes) to flights.
- Approaches:
  - Partition airspace into sectors, other structural elements
  - Partition aircraft (e.g., into “gaggles”)

Airspace Sectorization Problem

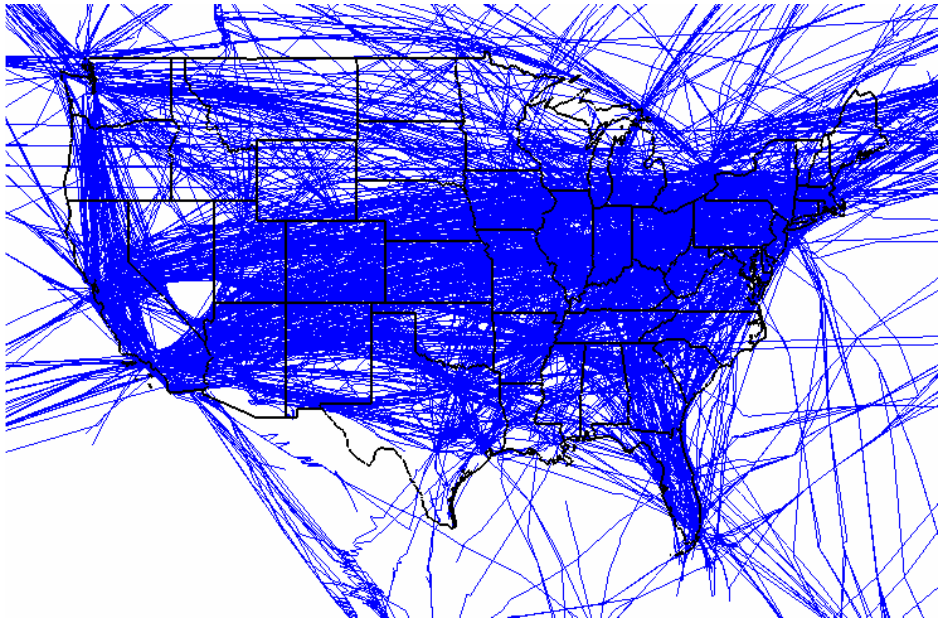
- k – no. of sectors
- b_i – Workload(WL) of sector i *"Workload" – tricky to model*
- Given the air-traffic pattern (trajectories), decompose the domain of airspace into k sectors, "optimally"
  - Min-Max WL
  - Min-Sum WL



- OR given a max Workload B, minimize k

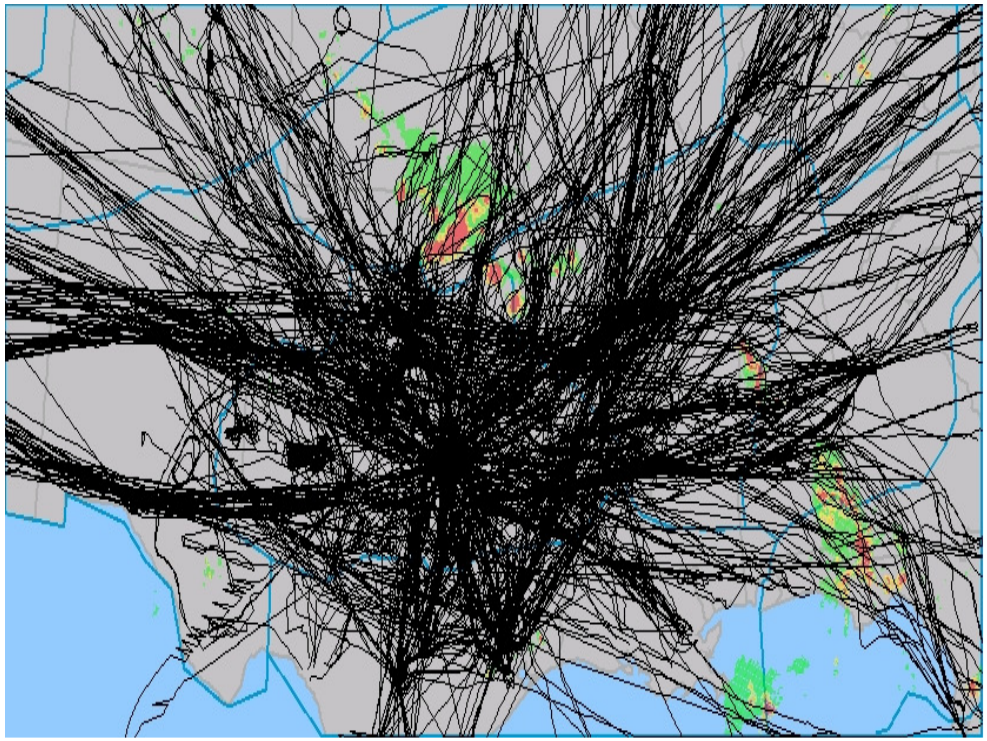
We model as precise optimal geometric partitioning problem, for which we give provable results and heuristics to approximate

Input: Demand



"Demand" given as a set of trajectories (flights) in space-time:

τ : (x_1,y_1,z_1,t_1), (x_2,y_2,z_2,t_2), (x_3,y_3,z_3,t_3), …

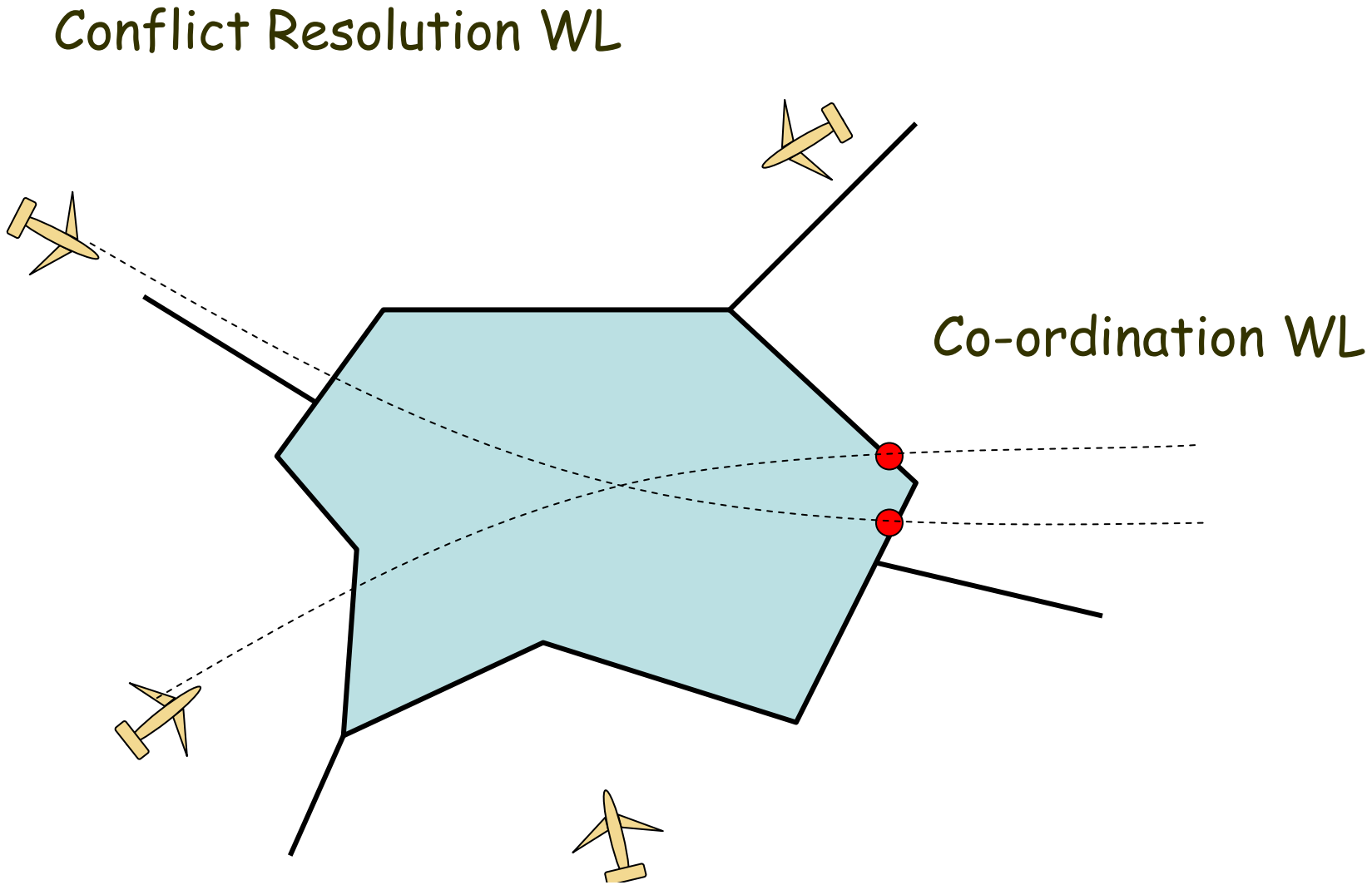


Metrics, Objective Functions

How to quantify b_i = Workload(WL) of sector i?

- Max # aircraft at any time instant (instantaneous count), or % of MAP values
- Average (over time) count
- Coordination workload: # Boundary crossings (hand-offs)
- Dynamic density
- Flight efficiency (fuel, time)
- DAC: frequency of change, magnitude of change, robustness to forecast inaccuracies

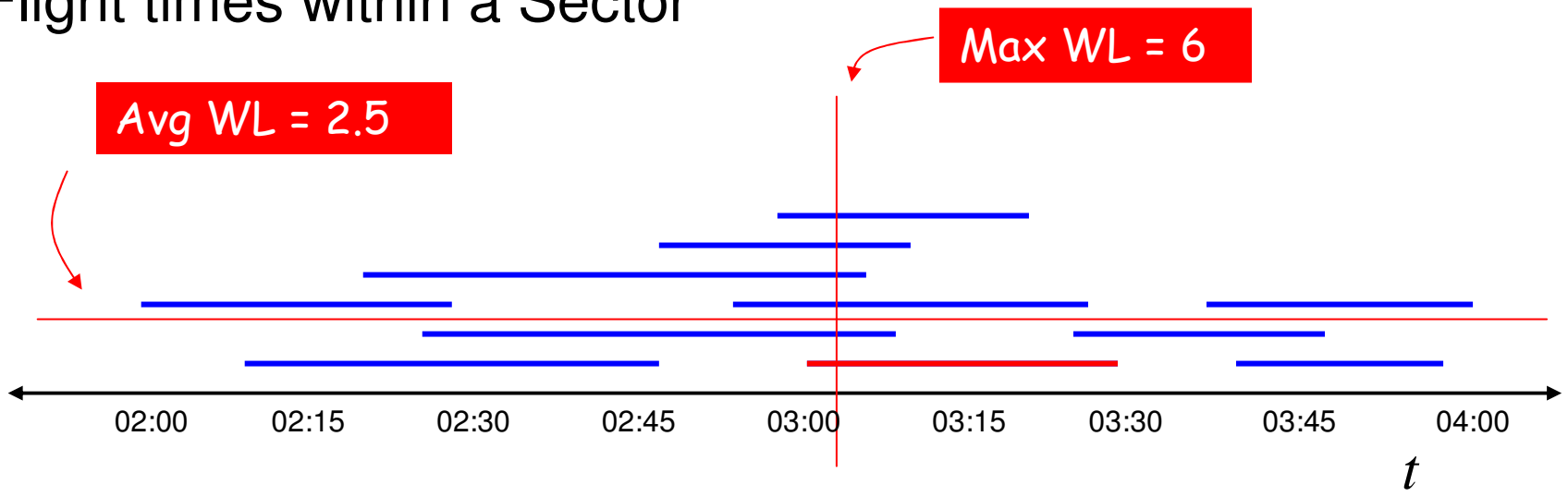
Workload of a Sector



Workload Modeling

- Worstcase WL : Max no. of planes in sector at any time (Max–WL)
- Avg WL : Avg no. of planes in sector over time
- Coordination Workload

Flight times within a Sector



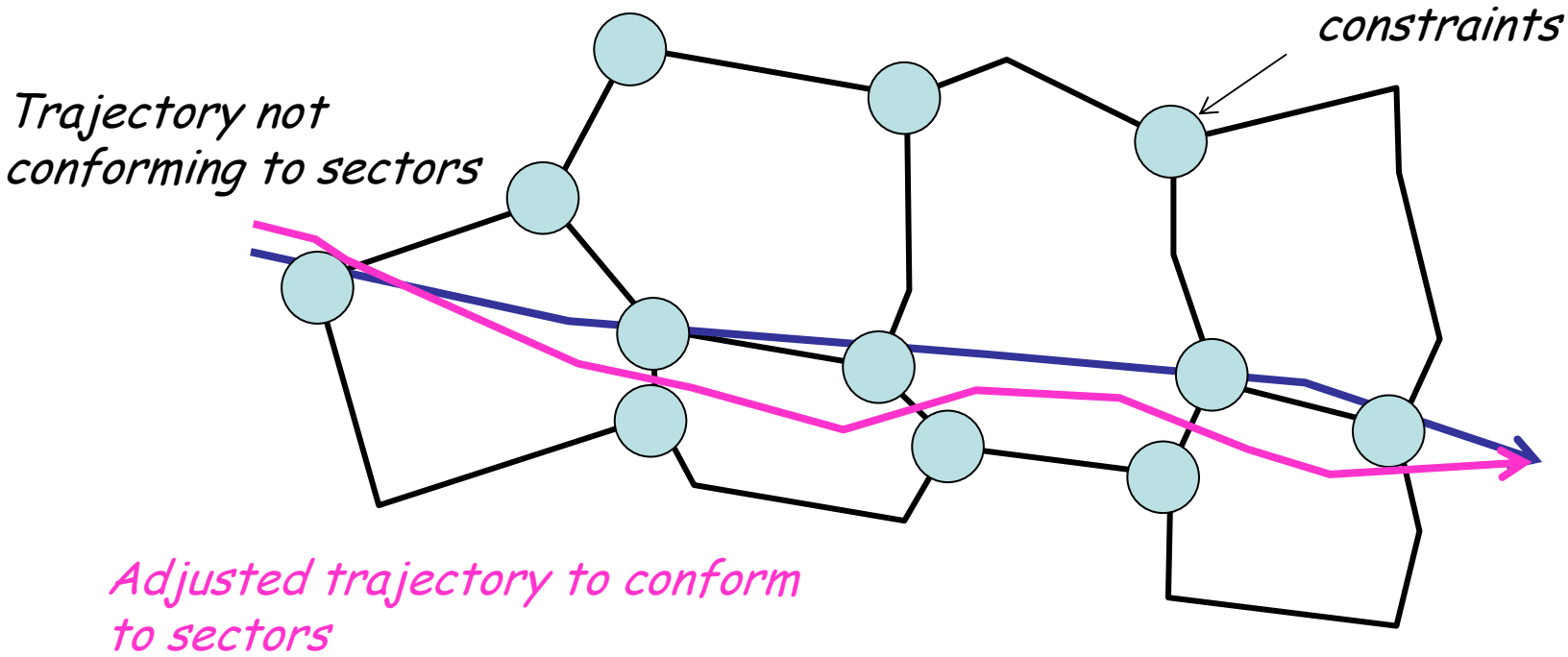
Good Sectors and Good Sector-Conforming Routes

- Find sectors *and* set of trajectories to
  - Minimize
    - # sectors
    - var(workload across sectors)
    - increase in flight length/fuel (over opt or user-preferred routes) to make conformal with sector design
  - Subject to:
    - Load balance (each workload < W)
- Multicriteria optimization

Iterative Process

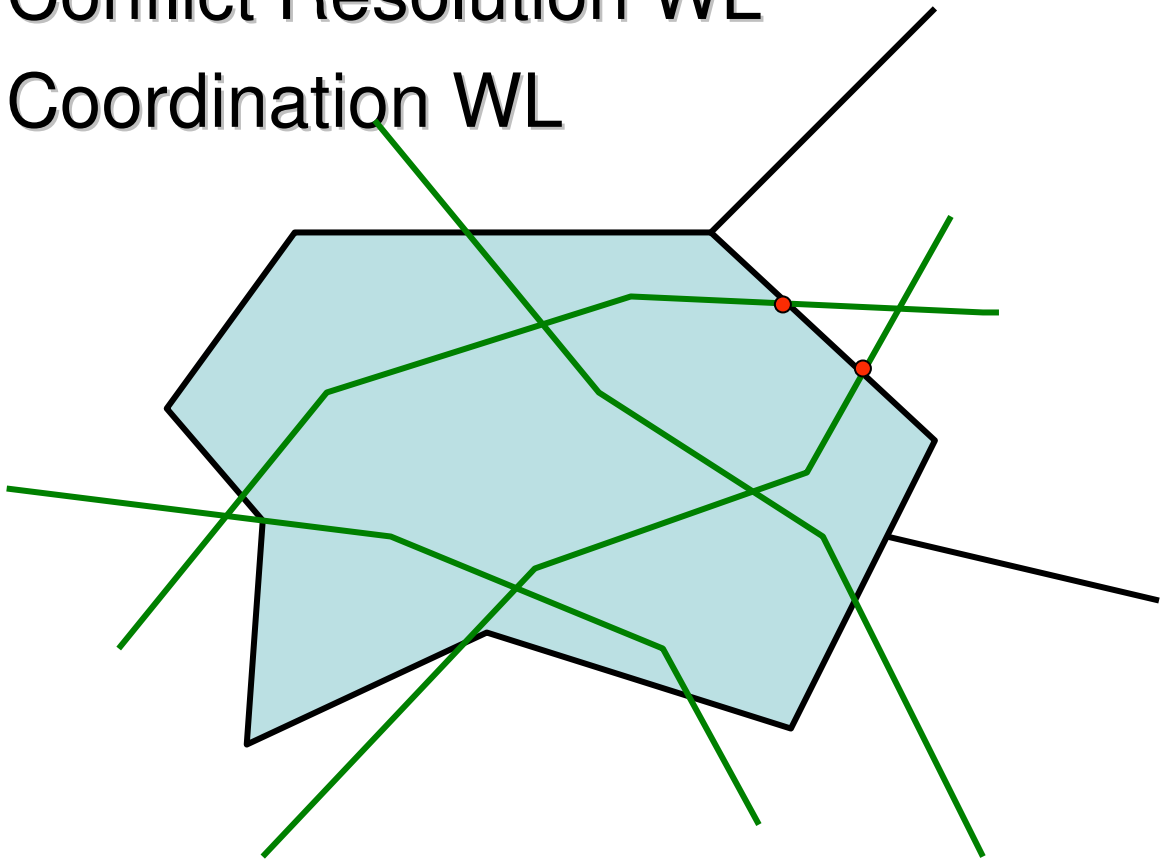
- Feedback loop: Iterative adjustment of routes to sectors and sectors to routes

Optimal path problem: Modify trajectory to avoid corner clipping, obey crossing rules, etc



Contributions to Workload

- Workload
  - Conflict Resolution WL
  - Coordination WL



Motivation

- The existing sectors boundaries
  - determined by historical effects
  - have evolved over time
  - not the result of analysis of route structures and demand profiles
- Hence the sectors are not WL balanced

- Also of the 15,000 Air Traffic Controllers, 7,000 are retiring in next 9 years

- Novel Partitioning : Non-static (Steiner) points

Related Work

Airspace Sectorization Problem

- Integer programming; after discretizing the NAS - Yousefi and Donohue ‘04
- Genetic algorithms - Delahaye et al. ‘98
- Graph partitioning methods - Tran et al. ’03, Martinez et al.’07

Other Partitionings

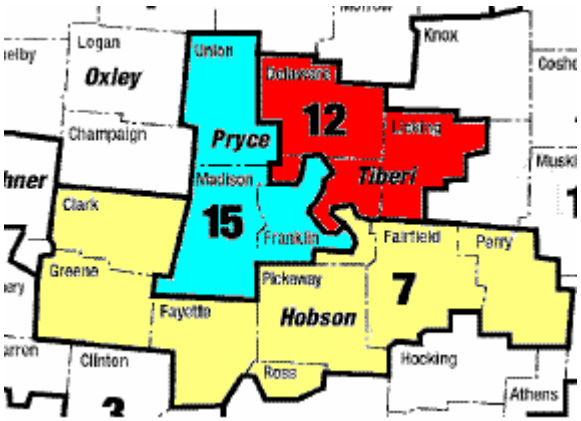
- Partitioning of rectangles and arrays for load balancing of processors - Khanna, Muthukrishnan, and Skiena ‘97
- Political districting – Altman ’97
- Minimum-cost load balancing in sensor networks – Carmi, Katz ‘05

Related: Optimal Load-Balancing Partitions

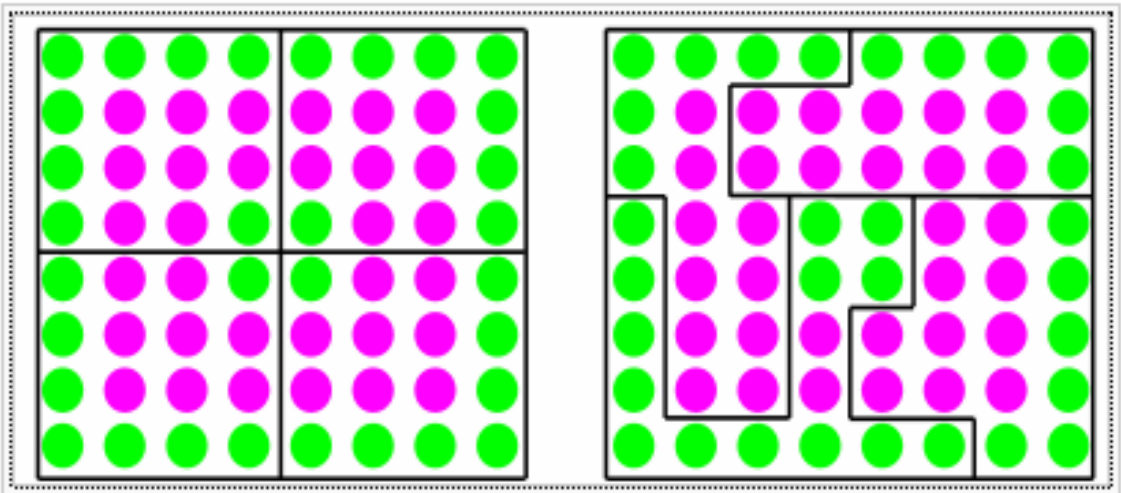
- Load-balancing:
  - Rectangular partition of *n* x *n* matrix A, into *k* rectangles in order to minimize the max weight of a rectangle is NP-hard – [Muthukrishnan et al '98]
  - The element values used in the reduction are constants ≤4

Forms basis of NP-hardness proof for sectorization

Related: Election Districting



An example of "cracking" style Gerrymandering; where the urban (and mostly liberal) concentration of Columbus, Ohio is split into thirds and then each segment outweighted by attachment to largely conservative suburbs.



Redrawing the balanced electoral districts in this example creates a guaranteed 3-to-1 advantage in representation for the magenta voters. Here, 14 green voters are *packed* into one district and the remaining 18 are *cracked* across the 3 other districts.

Source: Wikipedia

Gerrymandering

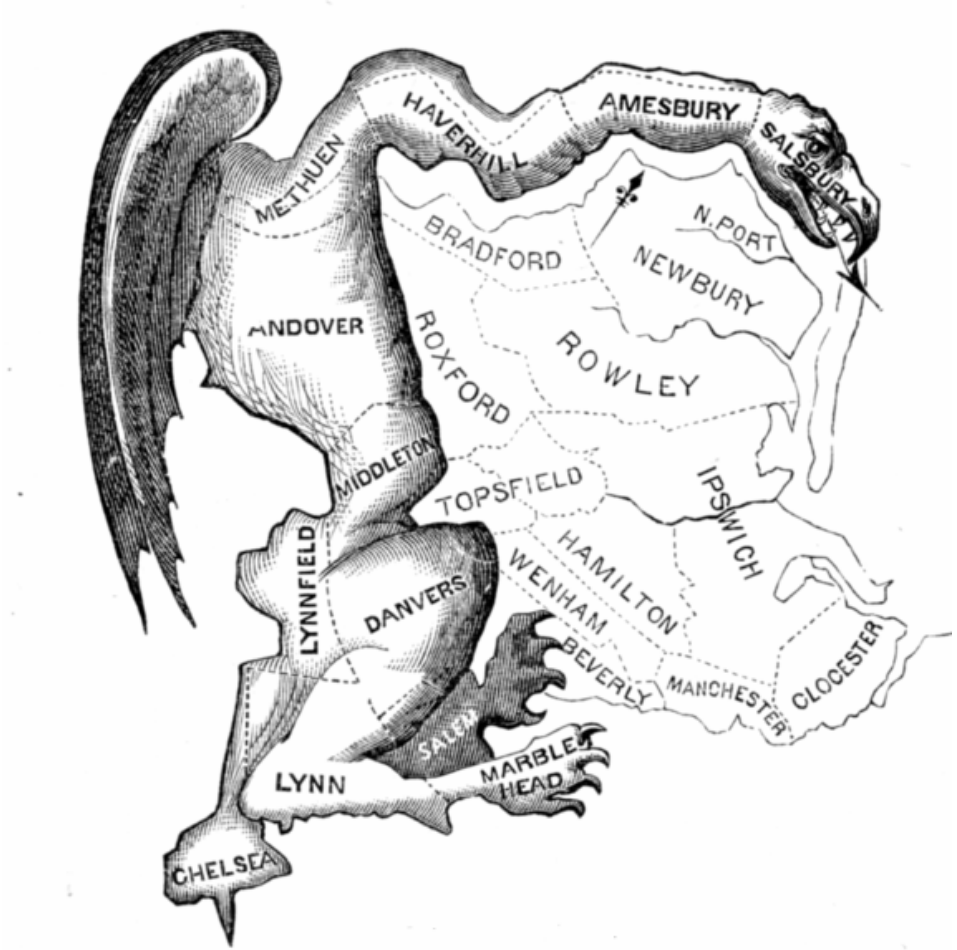
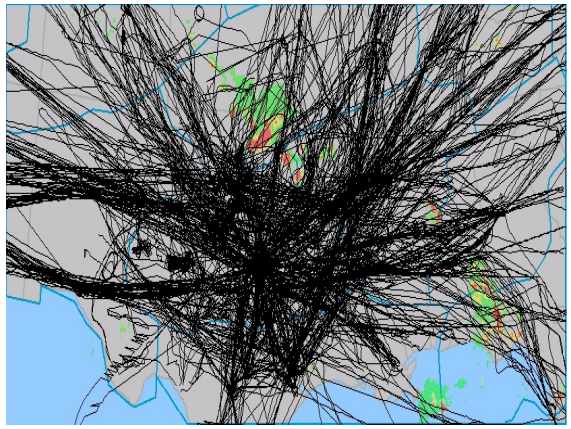


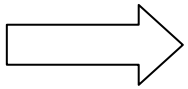
Image:The Gerry-Mander.png

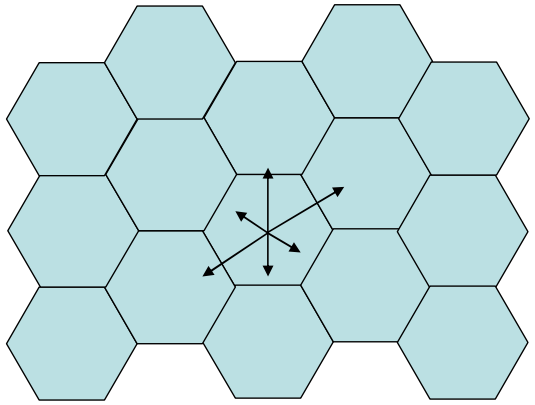
A gerrymandered Congressional District, the 11th CD of CA (now occupied by Democrat Jerry McNerney), drawn to favor Republican Richard Pombo. While the Danville area is a traditional Republican stronghold, Morgan Hill is not, and that largely Democratic district was added to obtain the proper population numbers for the 11th after Livermore was assigned to the 10th at the behest of the incumbent Democrat (Ellen Tauscher), since it contains the Lawrence Livermore National Laboratory (located near the "580" shield) and she sits in the House Energy Committee. The 10th CD is immediately north of the 11th in Contra Costa and Solano Counties. See the California 11th congressional district election, 2006 for an unexpected result that overcame this gerrymander.

Mixed Integer Program (MIP)



1. Demand input (flight tracks)





2. Tile the airspace; compute workload or complexity

$$\text{minimize} \quad \sum_{i,j} \frac{f_{i,j}}{c_{i,j}} \tag{1}$$

subject to :

$$\sum_{s} y_s = open \tag{2}$$

$$\sum_{il \text{ is a ToNode}} f_{i,j} - \sum_{jl \text{ is a FromNode}} f_{i,j} = S_i - D_{il \text{ is a sink node}} \tag{3}$$

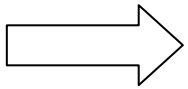
$$D_s \geq L \times \mu \times y_s \qquad L = (0.0,1.0] \tag{4}$$

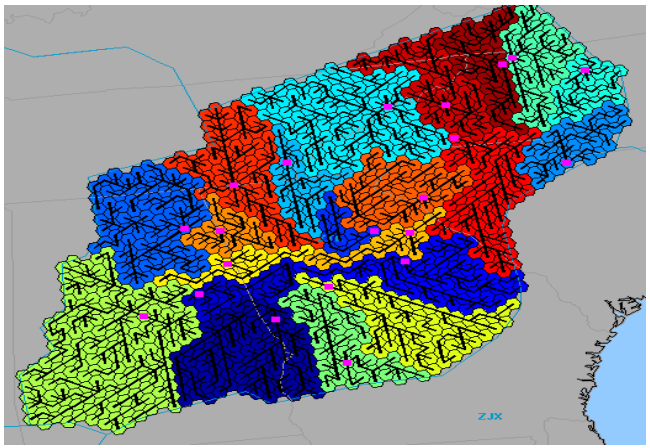
$$D_s \leq U \times \mu \times y_s \qquad U = [1.0,2.0) \tag{5}$$

$$\frac{f_{i,j}}{N} \leq u_{i,j} + y_{s=i} \tag{6}$$

$$\sum_{j} u_{i,j} = 1 \tag{7}$$

3. Mixed Integer Program





4. Sectorization output

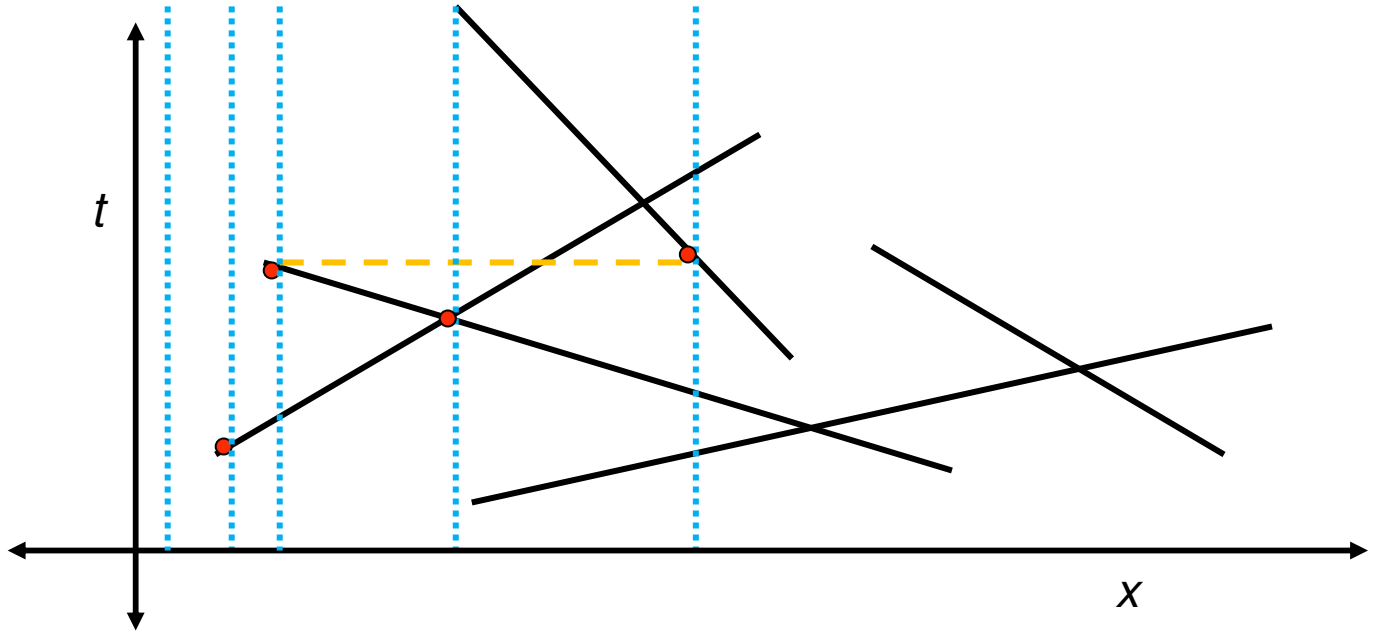
[Yousefi and Donohue '04]

Approach

- Devise mathematically precise problem statements for the optimization
- Utilize tools from computational geometry to design algorithms for efficient solution and to analyze these algorithms theoretically
- Use implementation and experimentation to verify proof of concept and to compare results to alternate approaches

1D Problem

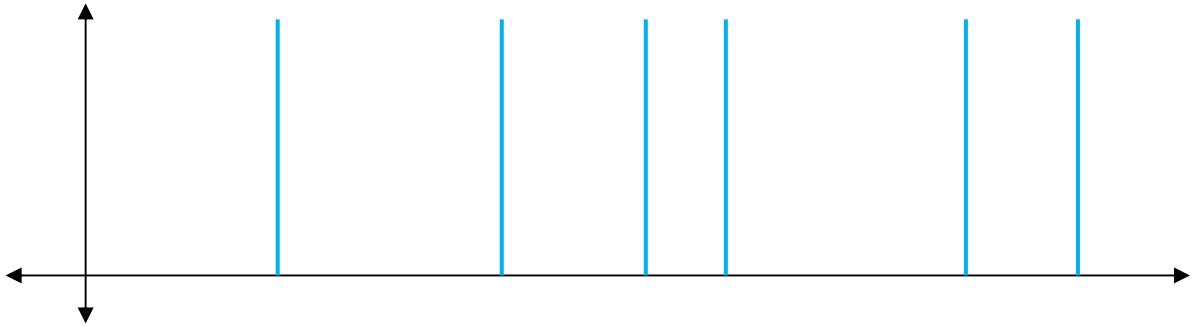
Given a budget B on max WL, minimize # of sectors (intervals)



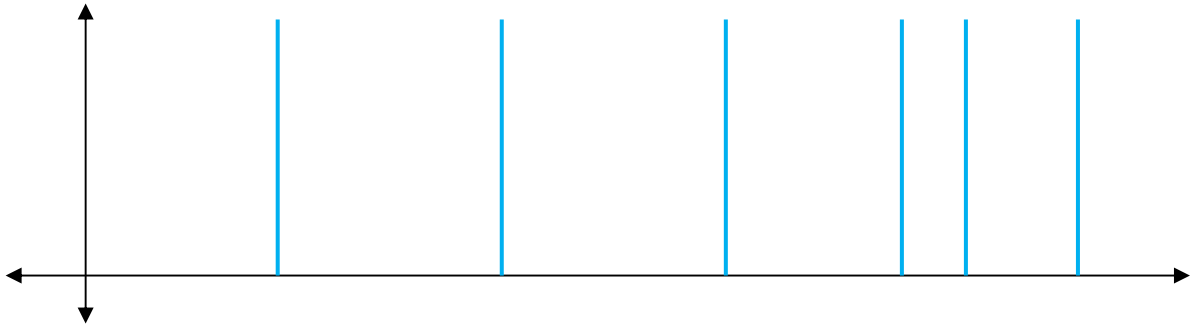
Greedy Algorithm

1D problem

Greedy = Optimal



Greedy



Optimal

1D problem

Running Time

- O(nlogn) Median 'x' Coordinate of the Vertices in Arrangement R. Cole, J. Salowe, W. Steiger, and E. Szemeredi SIAM J. Comput. '89
- O(nlogn) : Compute the Max-WL after the interval is decided
- O(log(n^2)) : Binary Search O(n^2) critical points
- Total running time O(nlogn * log(n^2)) = O(nlog^2(n)) * k

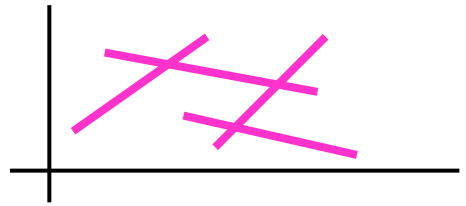
1D problem : Running Times

Worstcase WL: Given budget B, minimize # sectors k

- O(n logn) : Median x-coordinate of the Vertices in Arrangement [R. Cole, J. Salowe, W. Steiger, and E. Szemeredi SIAM J. Comput. '89]
- O(n logn) : Compute the Max-WL for a given interval
- O(log(n^2)) : Binary Search on O(n^2) critical points

- Total running time O(n logn * log(n^2)) = O(n $\log^2$ (n))
- B-level in arrangement of lines, Randomized Algorithm O(n logn) by T. Chan '99

- Total for k sectors: O(kn $\log^2$(n)) or O(kn log n)



Avg WL : Easier, O(k+n logn)

WL is a piecewise linear and continuous function with breaks at end points (O(n))

1D problem

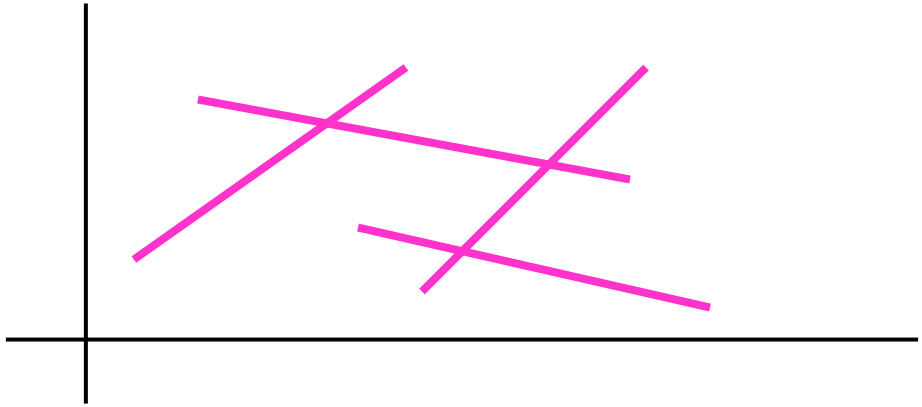
Time Avg WL

- Avg number of planes in a sector at any time

Easier to sectorize in 1D

- O((k+n)logn) for given budget B

- Because the WL is a piecewise linear and continuous function with breaks at end points (O(n))



2D problem

Hardness of special case

- Form small bundles of straight trajectories associated with each element of matrix, laid out in rectangular grid
- Sectorization in the form of rectangles will solve the NP-hard problem

2D Problem: Theory

x, y and t

Hardness of a special case

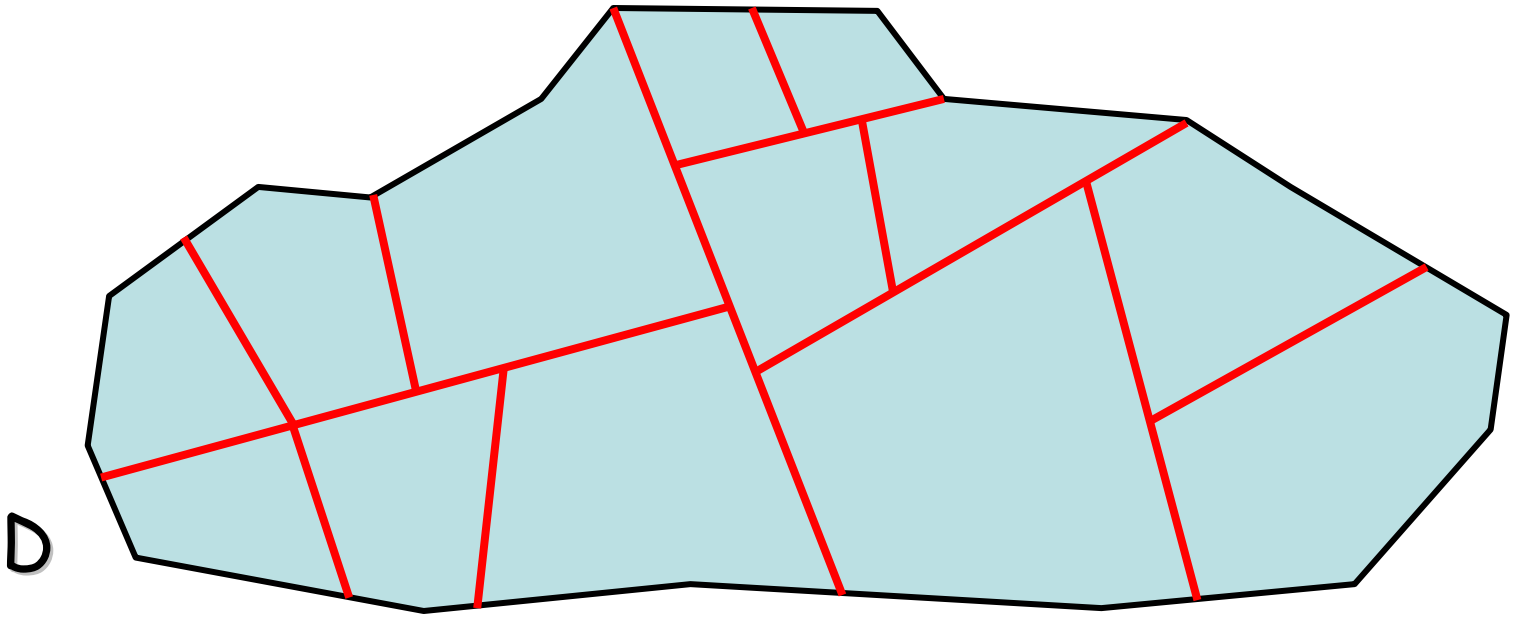
- Sectors - axis aligned rectangles
- Problem used for reduction :
  - Rectangular partition of n x n matrix A, into k rectangles in order to minimize the max weight of a rectangle is NP-hard – [Muthukrishnan et al '98]
- The element values used in the r
  are constants ≤4

Form small bundles of straight trajectories associated with each element of matrix, laid out in rectangular grid

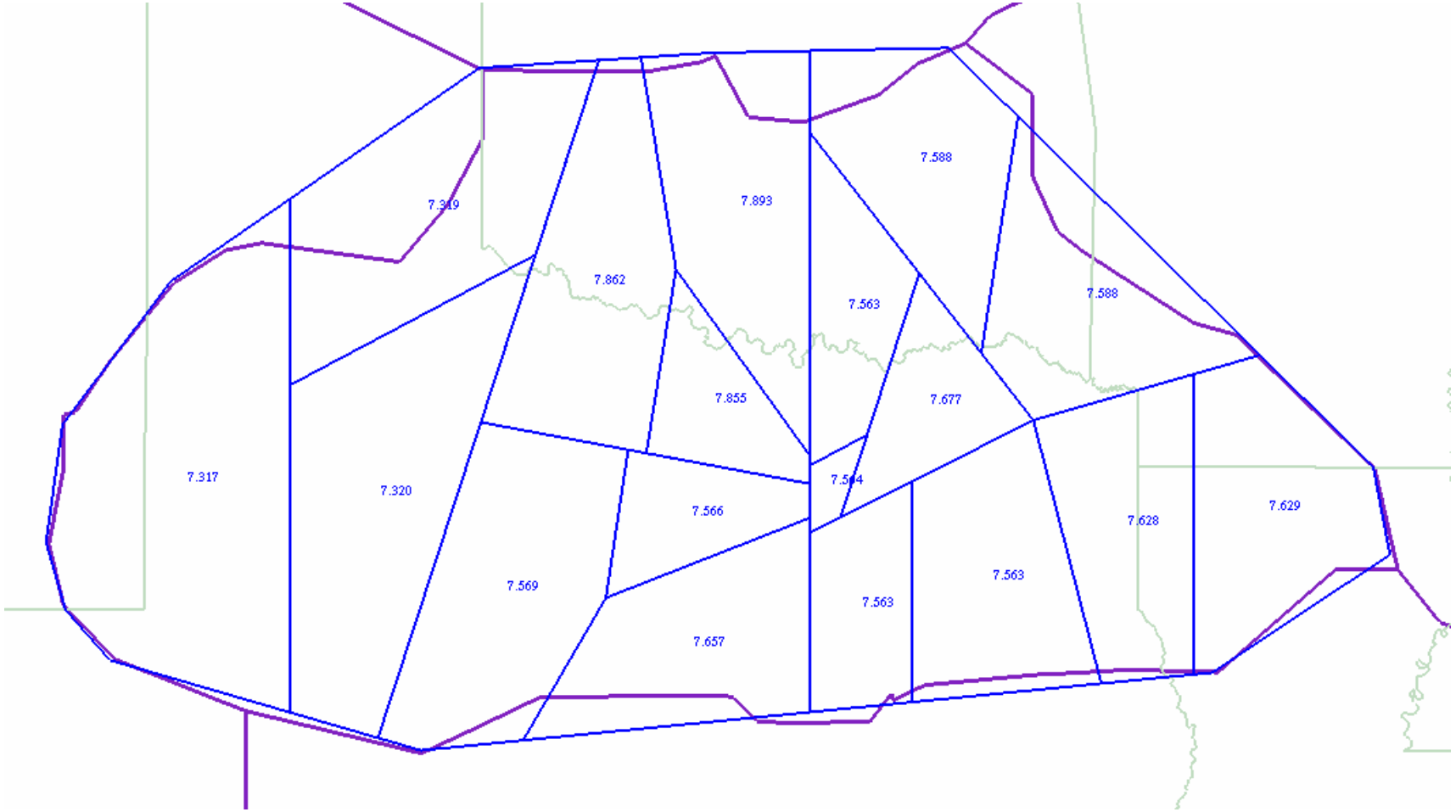
CG Concept: Binary Space Partition (BSP)

BSP: Special case of a Convex Partition, P, of a domain D

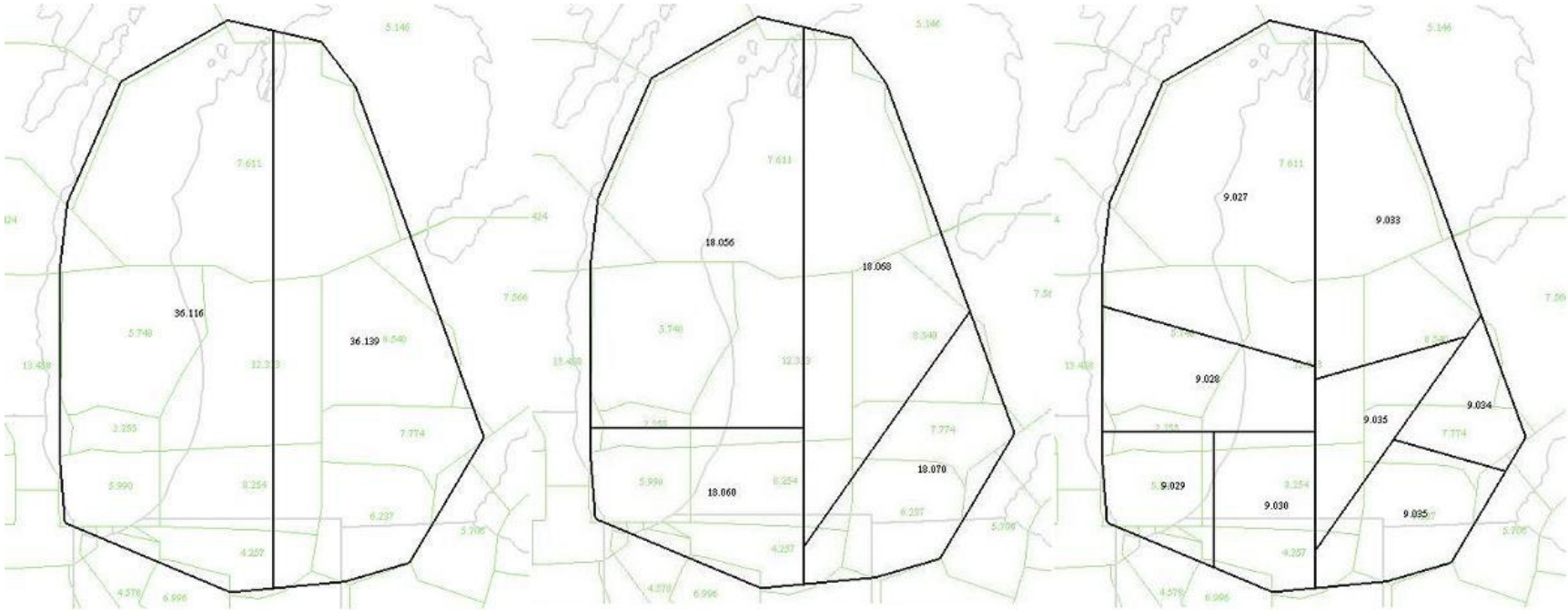
Defining property: Recursively obtained by cutting a face of P into two subfaces by a line/plane/hyperplane, cutting “all the way through” at each step



Example: ZFW Divided into 18 Sectors

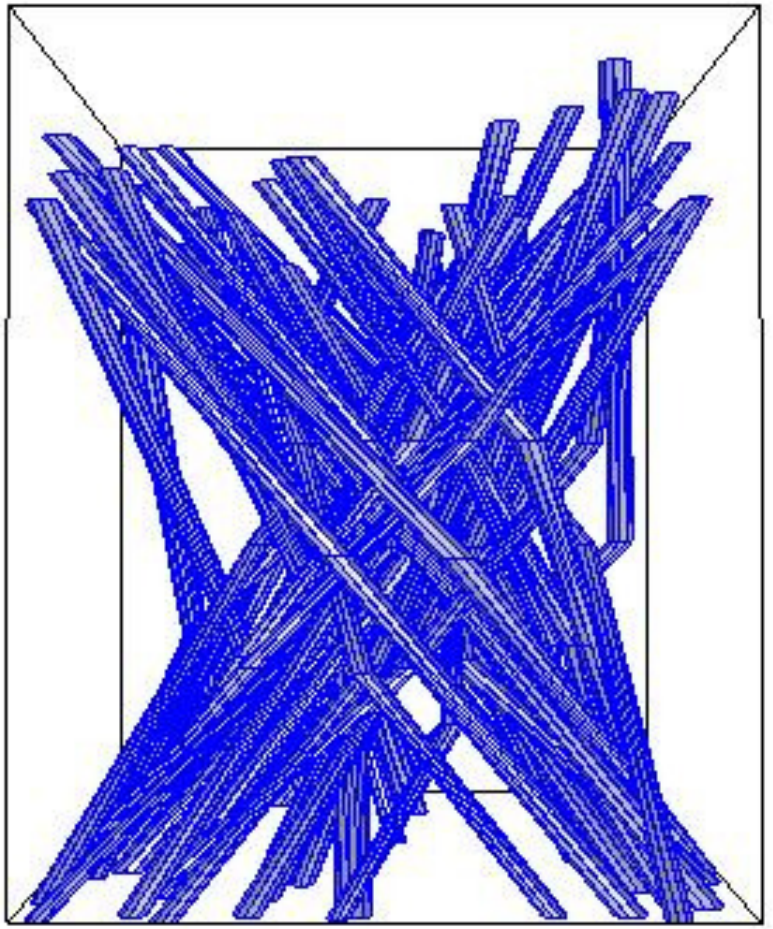


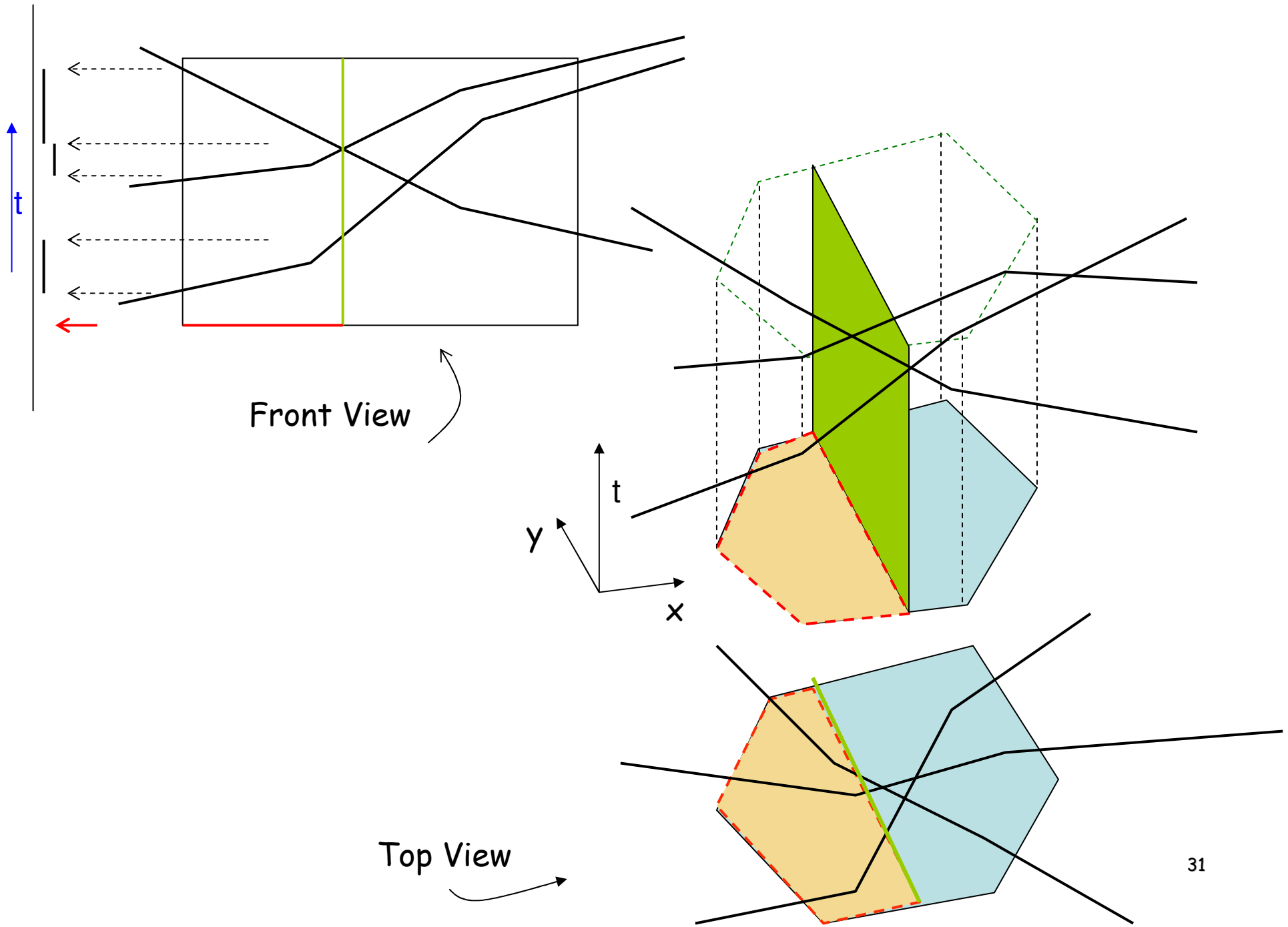
Example: 2, 4, 8 Sectors



View of Trajectories in Space-Time

View in space-time
(x,y,t) or (x,y,z,t)




t
Front View
t
y
x
Top View

Optimizing a Binary Space Partition (BSP)

Dynamic Programming: Optimize over all BSP's, exploiting the fact that BSP's are recursively defined.

Result: Provably optimal (among BSP partitions) method to partition an airspace into sectors, in a top-down fashion, for any specification/definition of "work load" in a sector

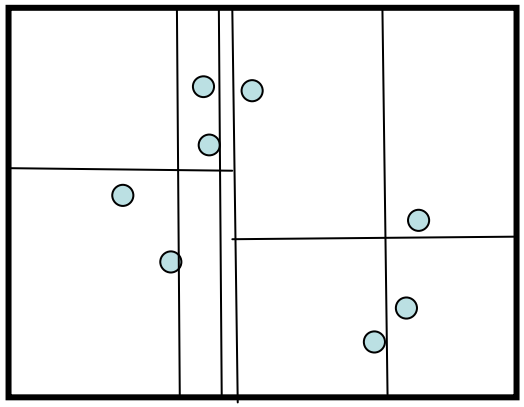
(minimize the maximum workload in a sector – most balanced sectorization)

Optimal Load-Balancing BSP Partition

- Dynamic Program
- Input: *n* points in a rectangle, *R*
- Objective: BSP-Partition *R* into *m* rectangles, each with exactly *k* points, while maximizing the minimum aspect ratio ("niceness")

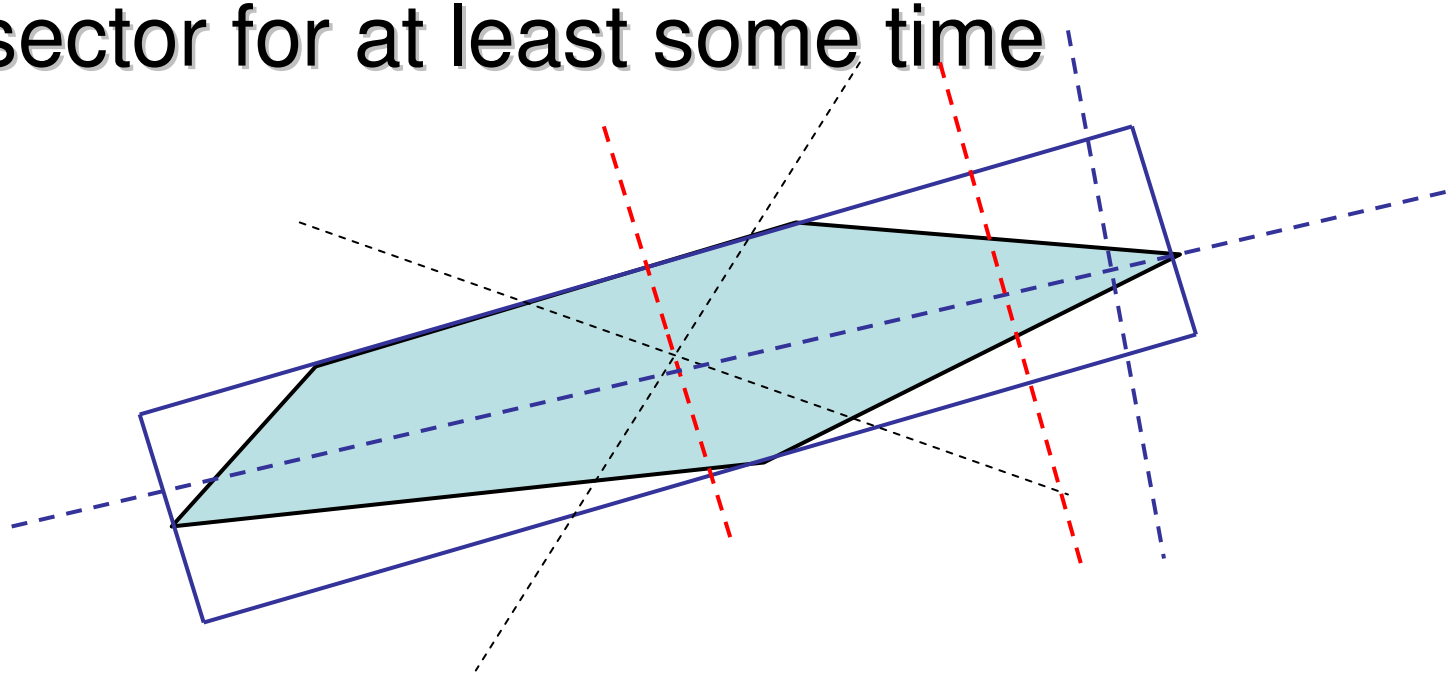
k=2

Subproblem: rectangle (x_1, x_2, y_1, y_2)



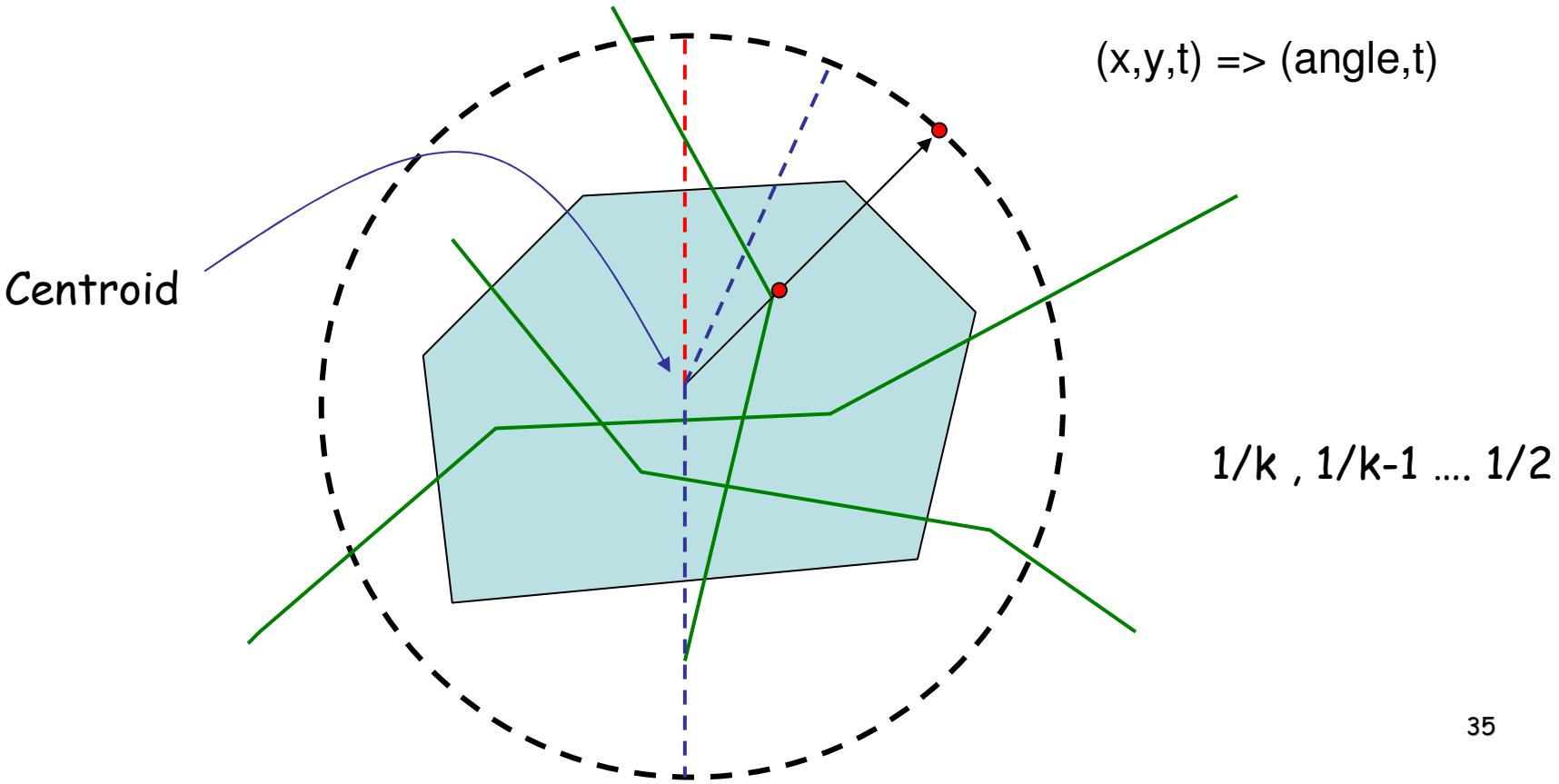
Avoiding Skinny Polygons

- To ensure that the flights remain in the sector for at least some time



Pie-Cuts

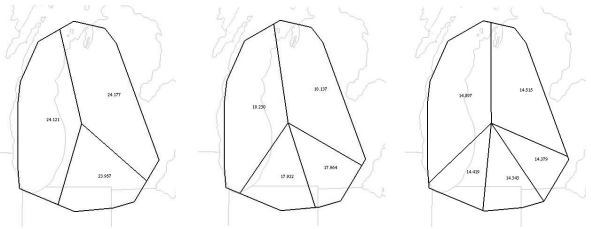
- Extension of 1D to Polar coordinates

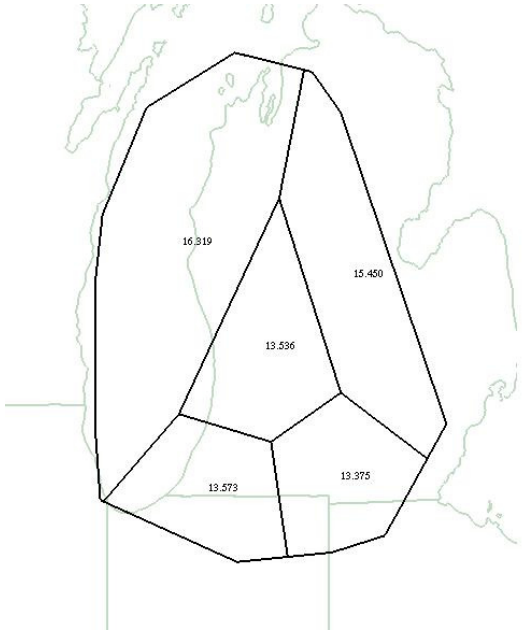


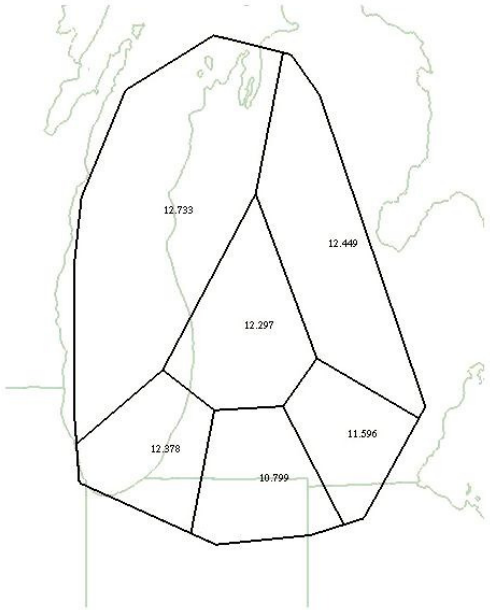
Combining BSP and Pie-Cuts

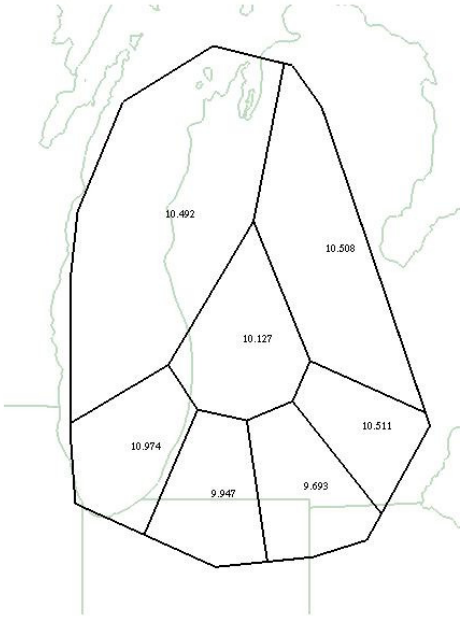
- Given a region to sectorize,
  - Try Pie-cuts starting with 5 or 6 down to 3
  - If Pie-cut was successful recurse on the smaller pieces
  - Else use BSP to sectorize and recurse on the two pieces

Wheel Cuts




16.319
15.450
13.536
13.573
13.375

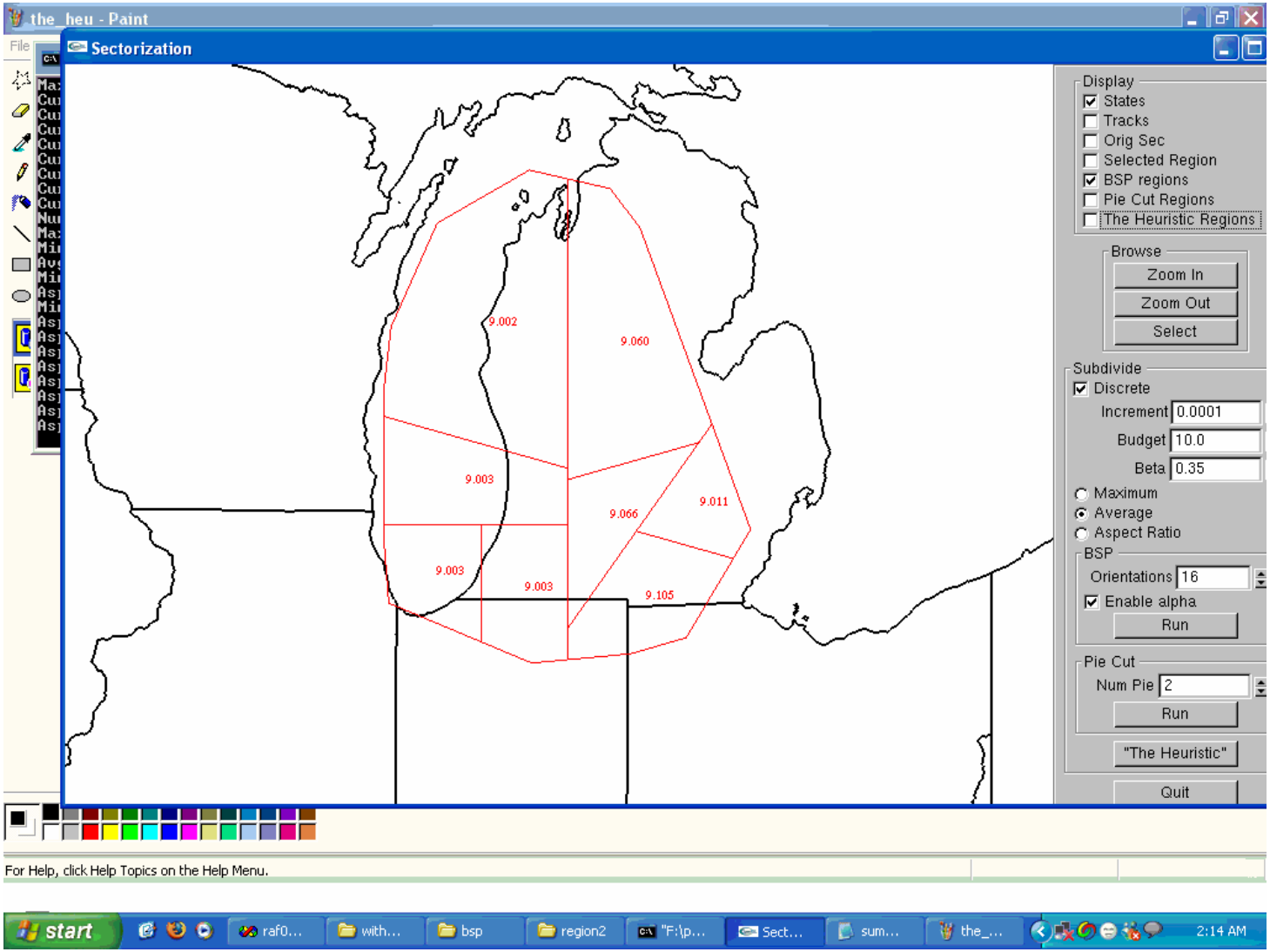

12.733
12.449
12.297
12.378
11.596
10.799


10.492
10.508
10.127
10.974
10.511
9.947
9.693

Analysis Data

- Utilize historical track data as input to optimization
- Any set of tracks (e.g., direct routes, wind-optimized routes, predicted reroutes around weather) can be input
- Experiments record workload parameters, including total # aircraft ever in a sector, time averages, and an estimation of coordination workload

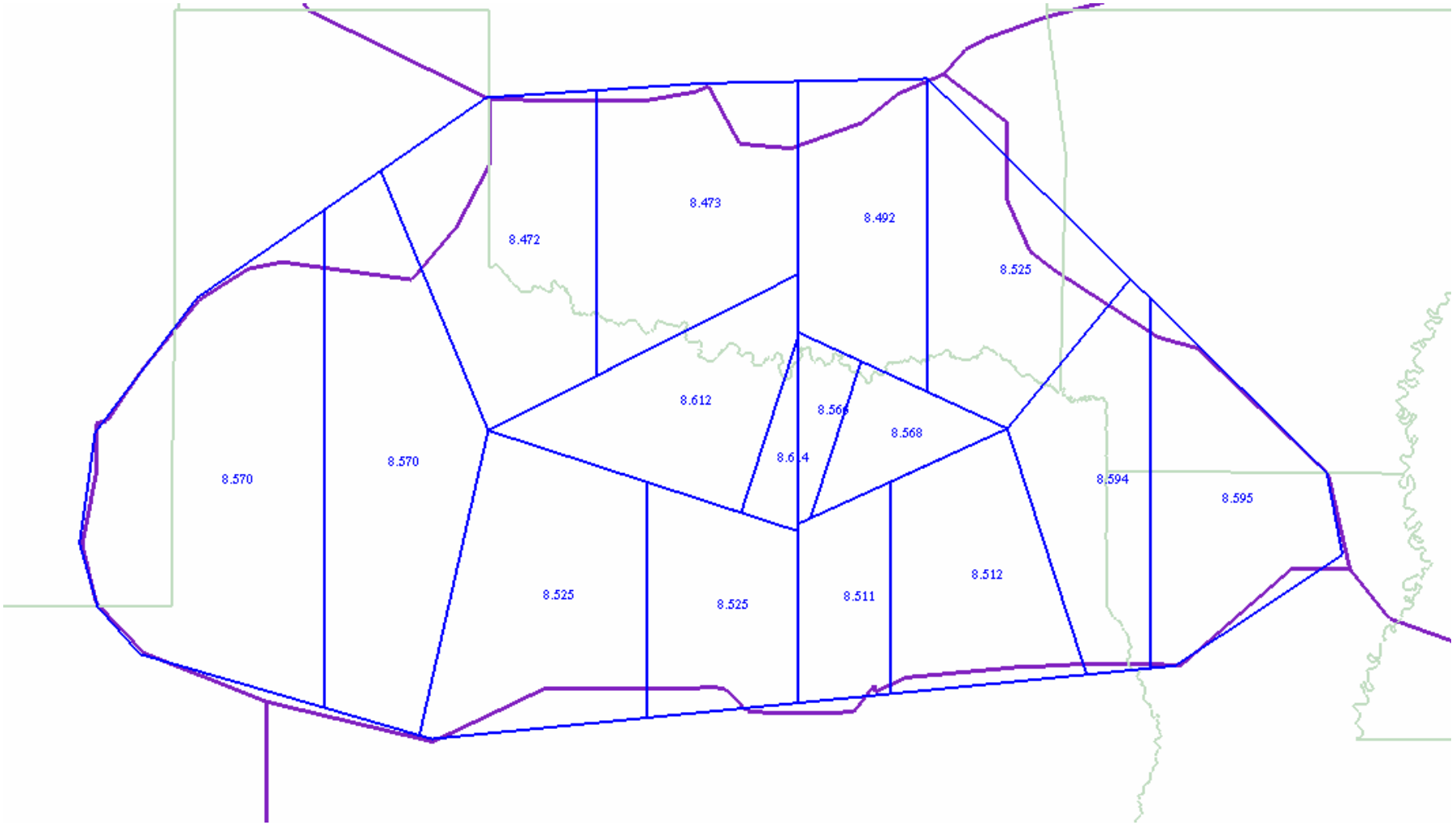
GeoSect1.0


the_heu - Paint
Sectorization
9.002
9.060
9.003
9.066
9.011
9.003
9.003
9.105
Display
States
Tracks
Orig Sec
Selected Region
BSP regions
Pie Cut Regions
The Heuristic Regions
Browse
Zoom In
Zoom Out
Select
Subdivide
Discrete
Increment 0.0001
Budget 10.0
Beta 0.35
Maximum
Average
Aspect Ratio
BSP
Orientations 16
Enable alpha
Run
Pie Cut
Num Pie 2
Run
"The Heuristic"
Quit
For Help, click Help Topics on the Help Menu.
start
2:14 AM

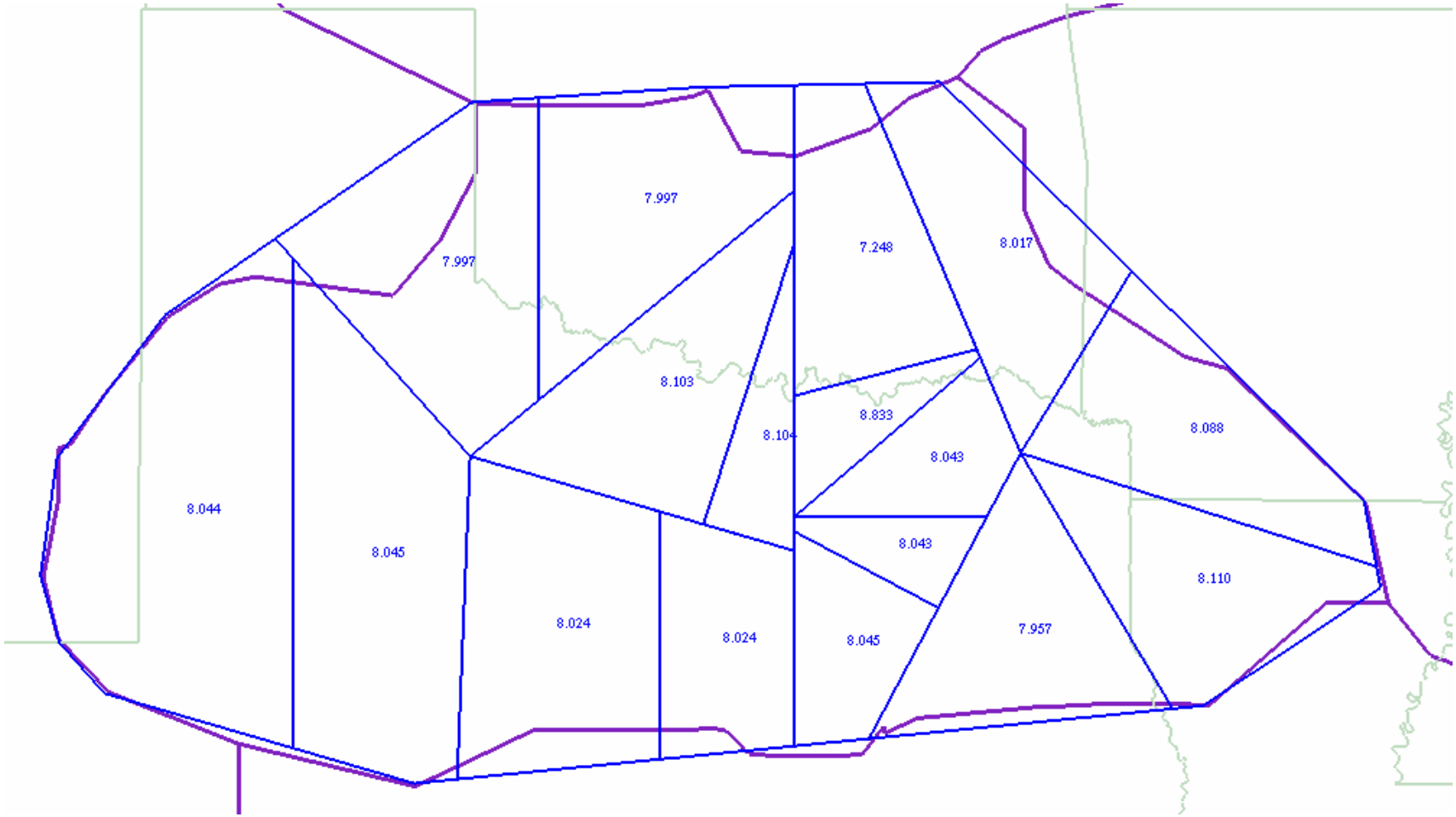
Experimental Setup

- 24hr track data (74588 flight tracks)
  - avg complexity 59.26 (no. of bends)
- Sector data
  - low altitude sectors
- US map data from Tiger Database

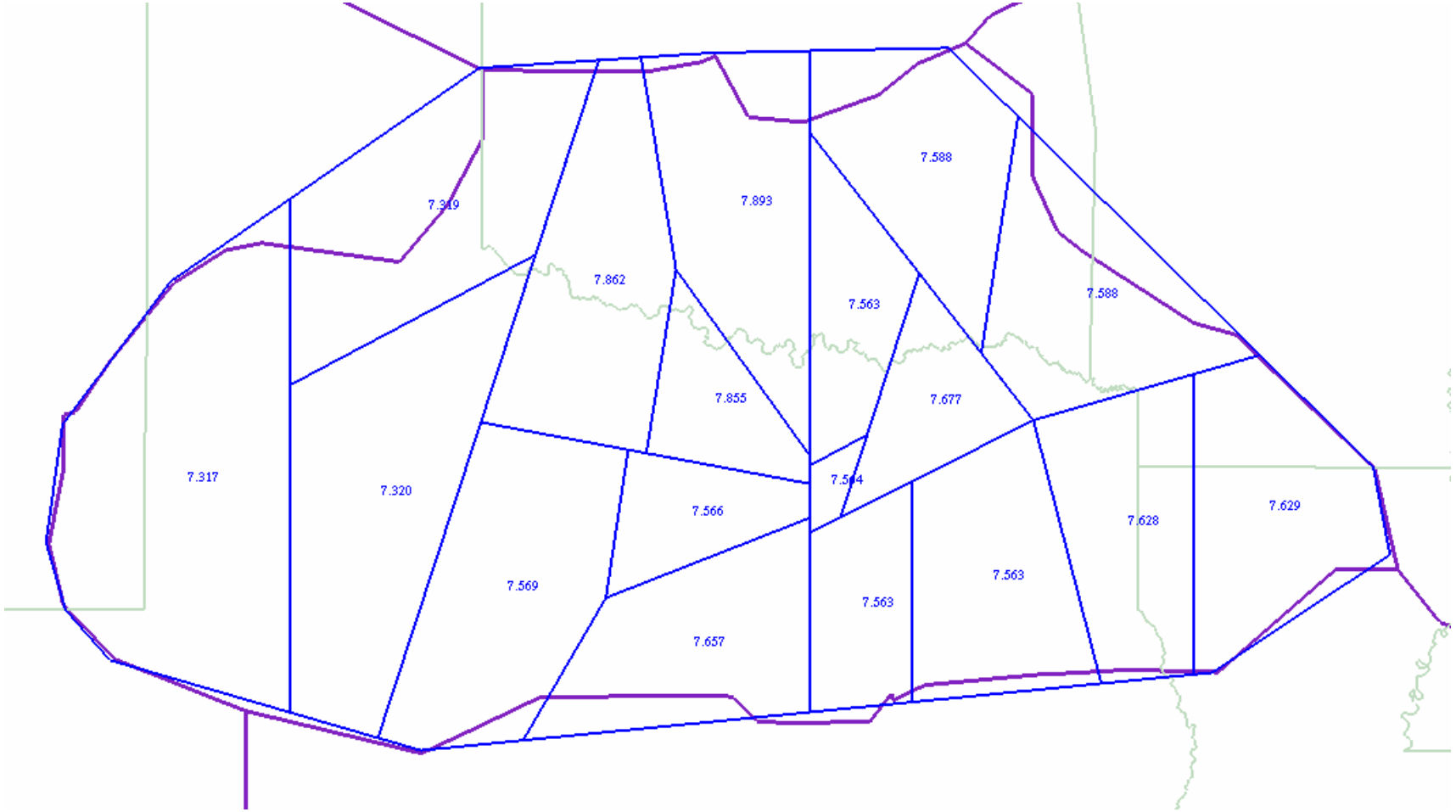
ZFW Divided into 16 Sectors



ZFW Divided into 17 Sectors



ZFW Divided into 18 Sectors



Discretization

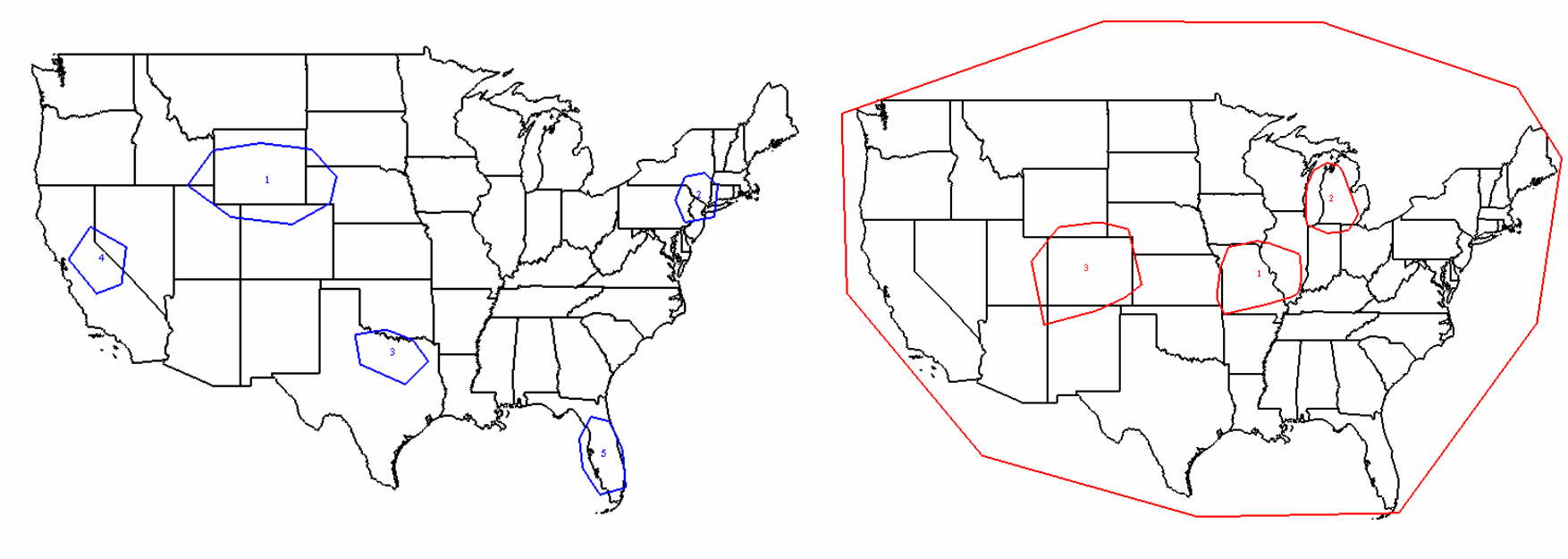
- No. of orientations to try
- Also the critical points are discretized (without any performance degradation due to the density of the track data)

Tuning the Parameters

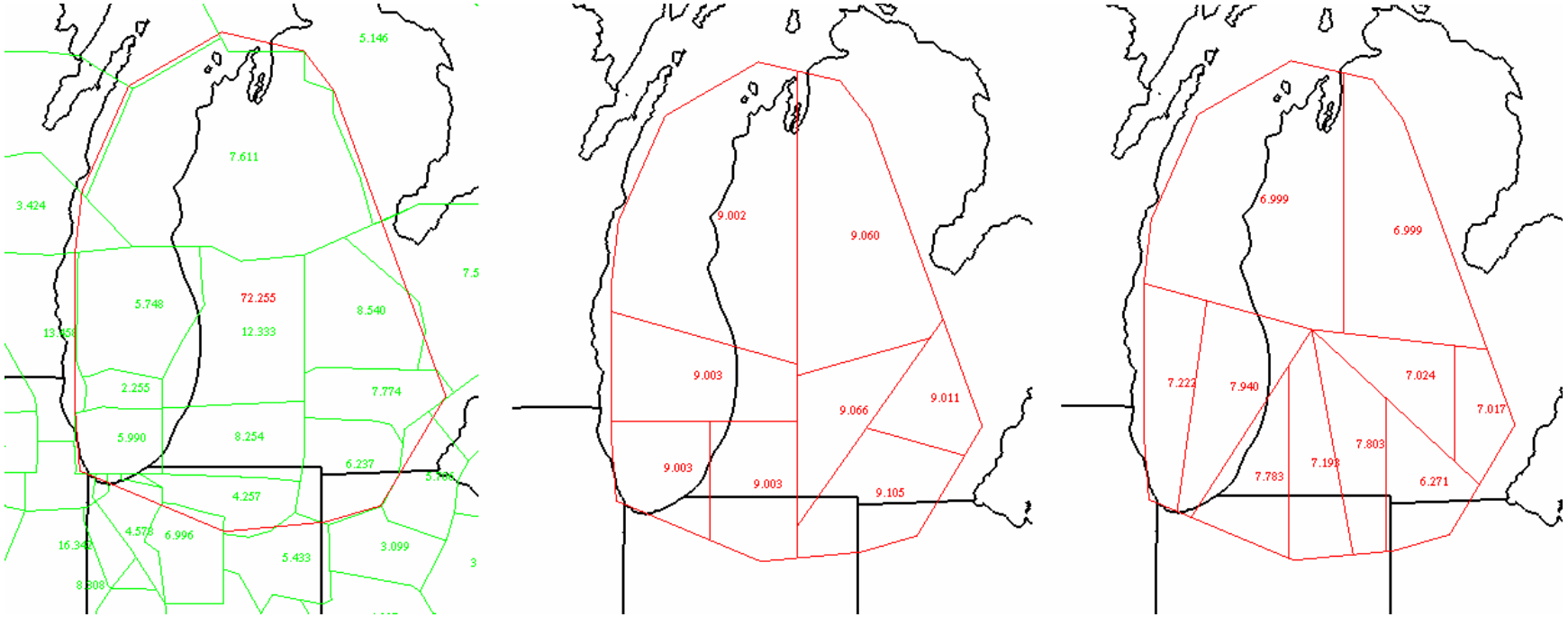
No. of different orientations

Threshold for aspect ratio beta

Level of Discretization



Results


5.146
7.611
3.424
5.748
72.255
12.333
8.540
7.5
2.255
7.774
5.990
8.254
6.237
4.257
4.578
6.996
5.433
3.099
9.002
9.060
9.003
9.066
9.011
9.003
9.003
9.105
6.999
6.999
7.222
7.940
7.024
7.017
7.803
7.783
7.193
6.271

Results

Sectorization	No. of Sectors	Time Average Workload			Worstcase Workload			α		
		Max	*Avg.*	*Std.Dev.*	*Max*	*Avg.*	*Std.Dev.*	*Avg.*	*Min*	*Std.Dev.*
Original Sectors	10	12.33	6.899	2.578	44	26.8	8.340	0.548	0.264	0.169
Final Heuristic	10	7.94	7.22	0.474	31	27.7	1.952	0.54	0.350	0.165
BSP	8	9.105	9.032	0.037	34	31.625	1.798	0.636	0.384	0.185

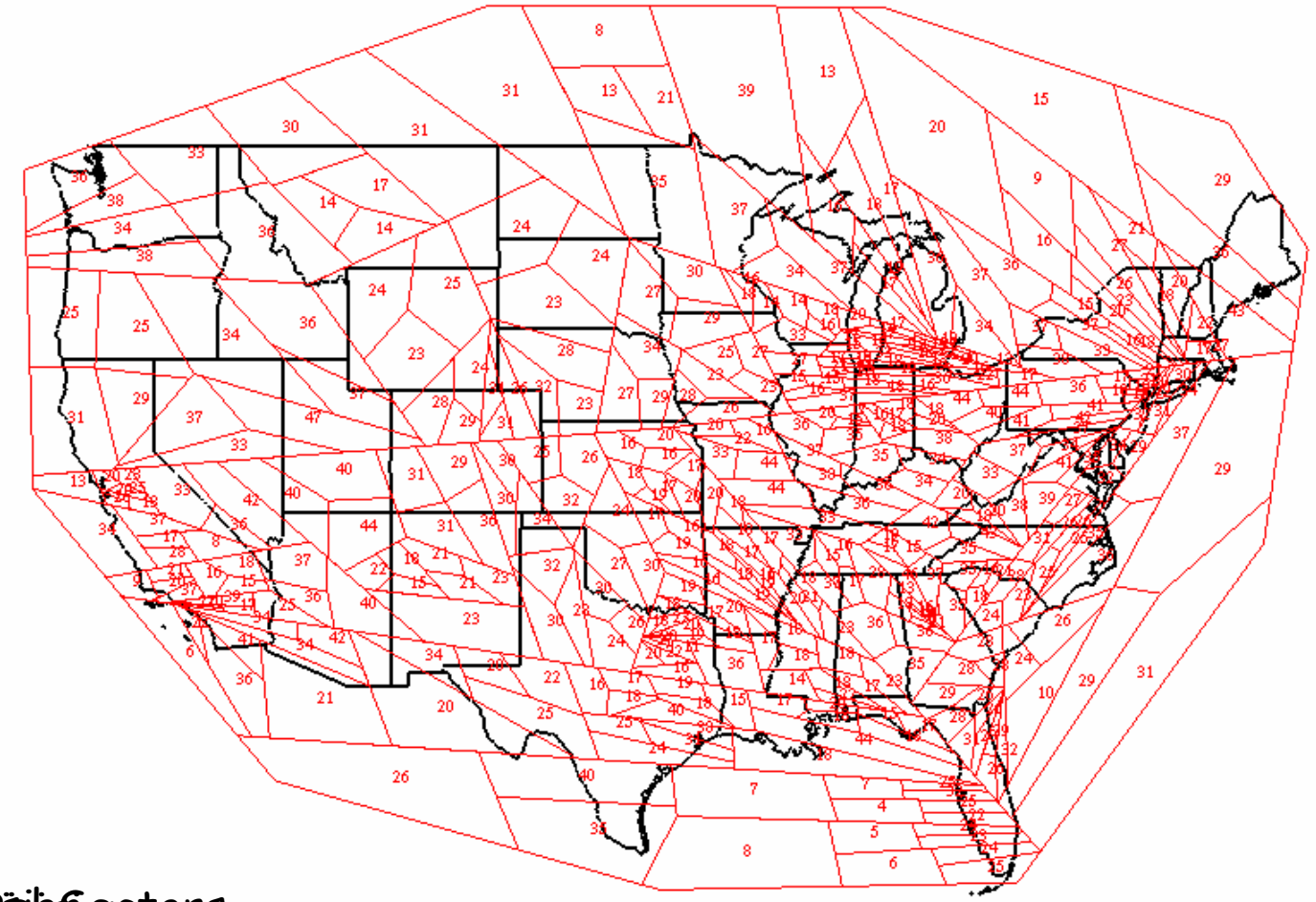
Figure 6: The statistics for pure BSP, the final heuristic and the original sectors for Region 2

Max goes down in both time-avg and worstcase

Std. Deviation improves significantly

Min Aspect Ratio improves

Results: NAS-wide Sectorization



Boundaries of Sectors

Results

Sectorization	No. of Sectors	Time Average Workload			Worstcase Workload			α		
		Max	*Avg.*	*Std.Dev*	*Max*	*Avg.*	*Std.Dev.*	*Avg.*	*Min*	*Std.Dev.*
Original Sectors	411	24.519	6.283	3.378	87	24.569	10.437	0.316	0	0.241
Final Heuristic	411	7.335	6.365	0.157	39	25.297	2.586	0.45	0.15	0.185
Final Heuristic	411	9.283	6.365	0.294	34	25.253	2.539	0.506	0.25	0.152
Final Heuristic	411	8.938	6.365	0.457	40	25.426	2.939	0.532	0.30	0.151
BSP	411	7.343	6.365	0.0715	34	25.207	2.567	0.588	0.15	0.188
BSP	411	9.568	6.365	0.426	36	25.11	2.882	0.60	0.25	0.181
BSP	411	9.545	6.365	0.512	35	25.1	2.849	0.578	0.30	0.164
Pie-Cut	412	11.085	6.35	2.901	47	25.041	8.812	0.286	0.021	0.175

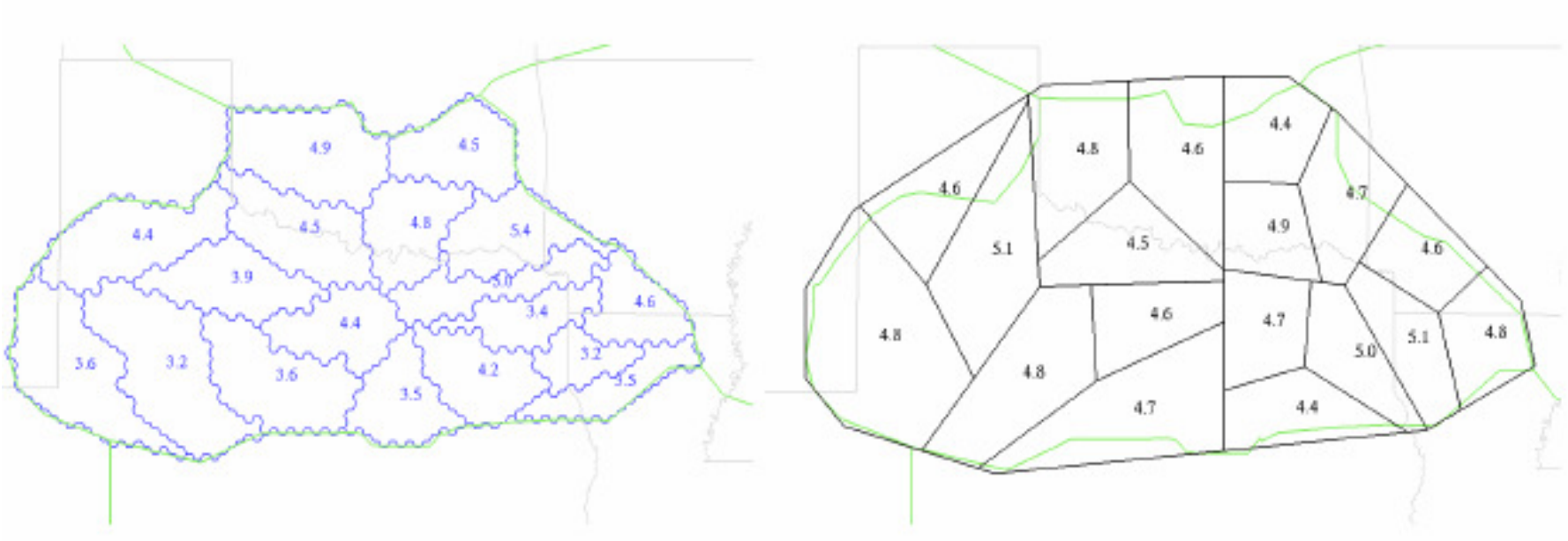
Results: Comparison of GeoSect Options

Sectorization	No. of Sectors	Time Average Workload			Worstcase Workload			α		
		Max	*Avg.*	*Std.Dev.*	*Max*	*Avg.*	*Std.Dev.*	*Avg.*	*Min*	*Std.Dev.*
Original Sectors	411	24.519	6.283	3.378	87	24.569	10.437	0.316	0	0.241
Final Heuristic	388	8.999	6.742	1.511	41	26.281	5.303	0.558	0.350	0.139
Final Heuristic	412	8.93	6.35	1.24	41	25.22	4.47	0.548	0.35	0.136
Final Heuristic	493	7.994	5.307	1.180	36	22.117	4.269	0.554	0.350	0.143
BSP	256	10.911	10.211	0.195	49	36.066	3.567	0.634	0.351	0.164
BSP	412	10.127	6.35	2.09	43	24.971	6.333	0.606	0.35	0.16
BSP	512	5.99	5.110	0.219	34	21.531	2.403	0.610	0.350	0.168
Pie-Cut	369	11.955	7.090	2.913	48	27.290	9.210	0.299	0.034	0.162
Pie-Cut	412	11.085	6.35	2.901	47	25.041	8.812	0.286	0.021	0.175
Pie-Cut	503	8.972	5.201	2.030	45	21.871	6.736	0.303	0.023	0.174

Figure 7: The statistics for pure BSP, the final heuristic and the original sectors for Region 2

Comparison with Clustering Methods

Sectorization	No. of Sectors	Time Average Workload			Worstcase Workload			α		
		Max	*Avg.*	*Std.Dev.*	*Max*	*Avg.*	*Std.Dev.*	*Avg.*	*Min*	*Std.Dev.*
IP Method	18	5.408	4.184	0.658	20	16.611	2.059	0.210	0.442	0.148
Final Heuristic	18	5.158	4.771	0.194	23	18.167	2.034	0.319	0.600	0.173



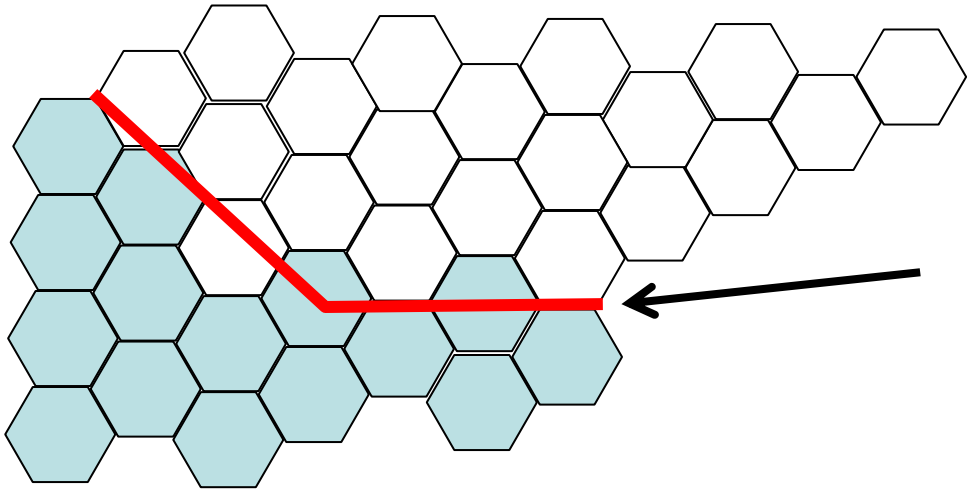
Optimal Smoothing Problem

Given: Integer-programming-based solution (hex cells) (from Arash)

Goal: Compute "optimal" smoothed boundaries

(combine MILP and GeoSect techniques)

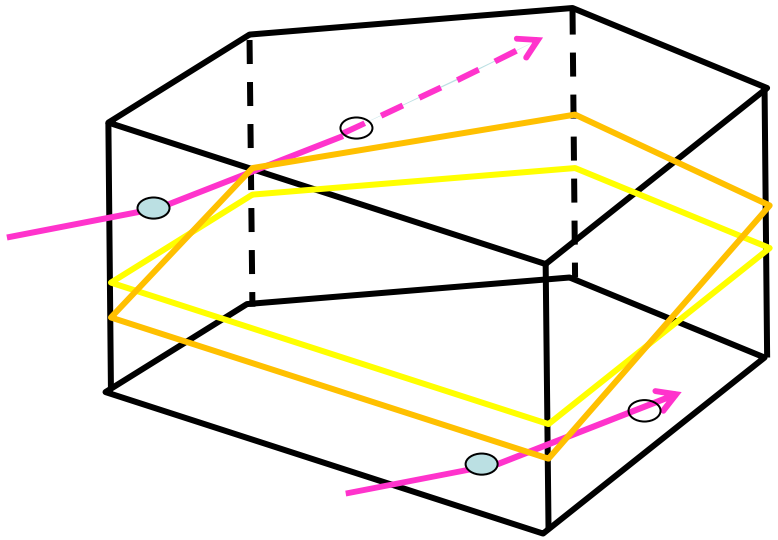
Method: Use CG concepts of optimal paths, link distance, and optimal workload partitioning. Among all possible polygonal chains of k "links" (edges), that join two degree-3 vertices in the hex-cell map, find the optimal path according to an objective function based on min-max workload on each side of the path.



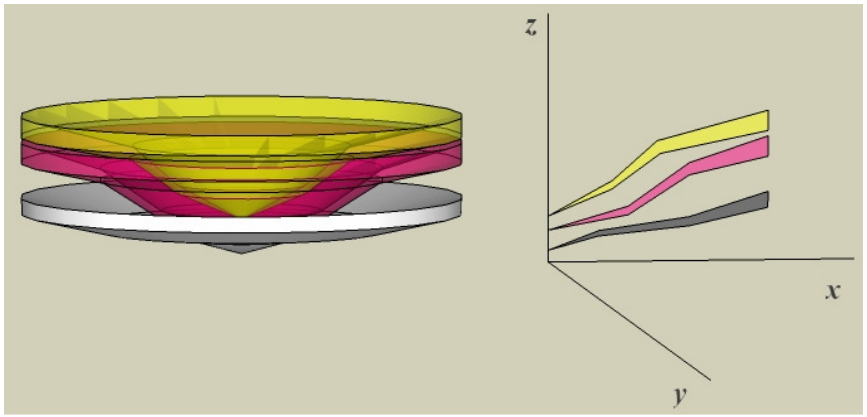
Optimal 2-link partition path between two adjacent sectors of hex-cells

3D Partitioning

- Types of cuts:
  - z = constant (flat)
  - Slanted cuts, cones (centered at airports)

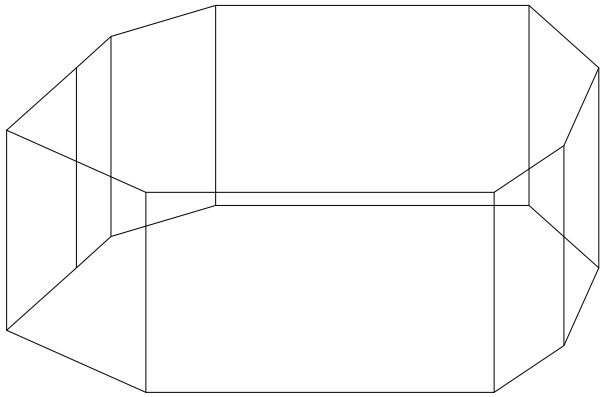


  - Challenge: cruise-climb airspace

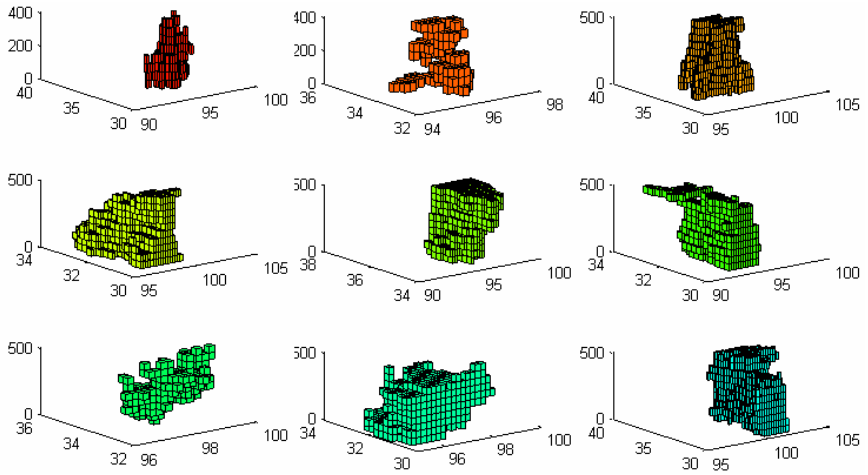


Difficulty of MIP Formulation

- 3D Sectors should be *right prisms*
  - The top and bottom lie directly over each other



Right Prism



Not Right Prisms

- Therefore right prism constraints must be added to the MIP formulation

Algorithm Description

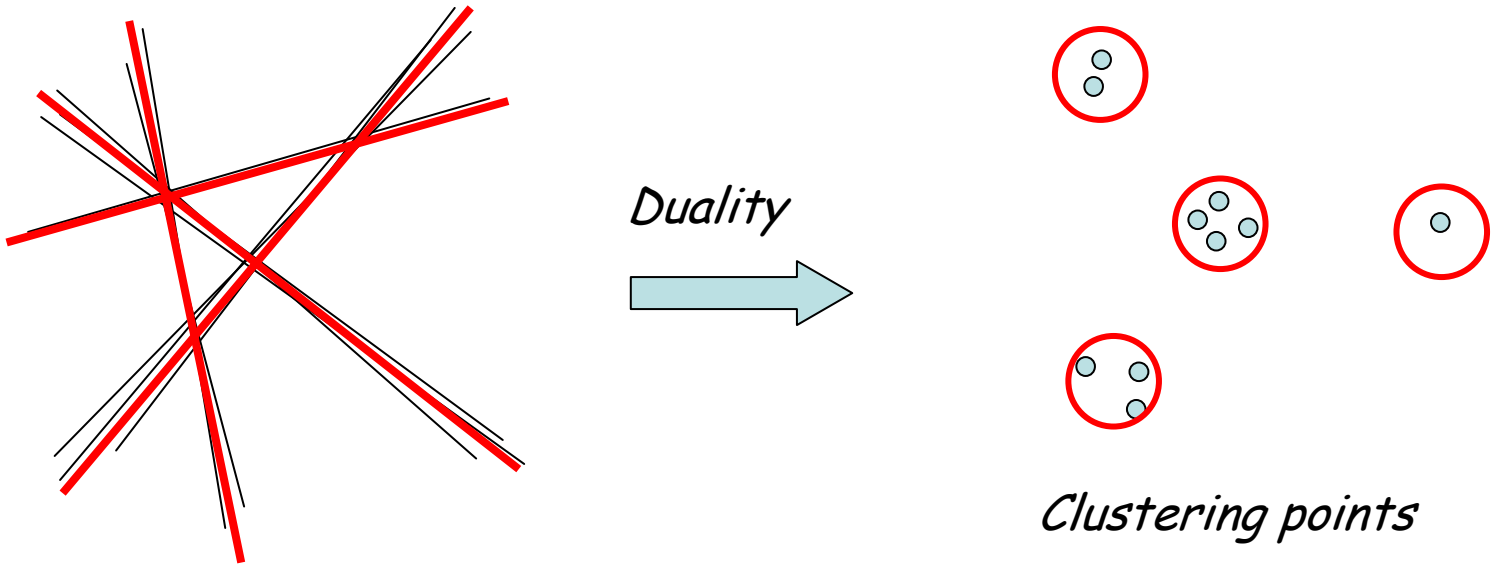
- Current algorithms in GeoSect:
  - "Top-down" – start with a large portion of airspace and recursively optimize a decomposition of it into "nice" subregions in order to optimize an objective function modeling workload
  - Implementation based on Binary Space Partitions, with subproblem optimization based on sweep techniques (from 1D problem statement)

Summary of Approaches

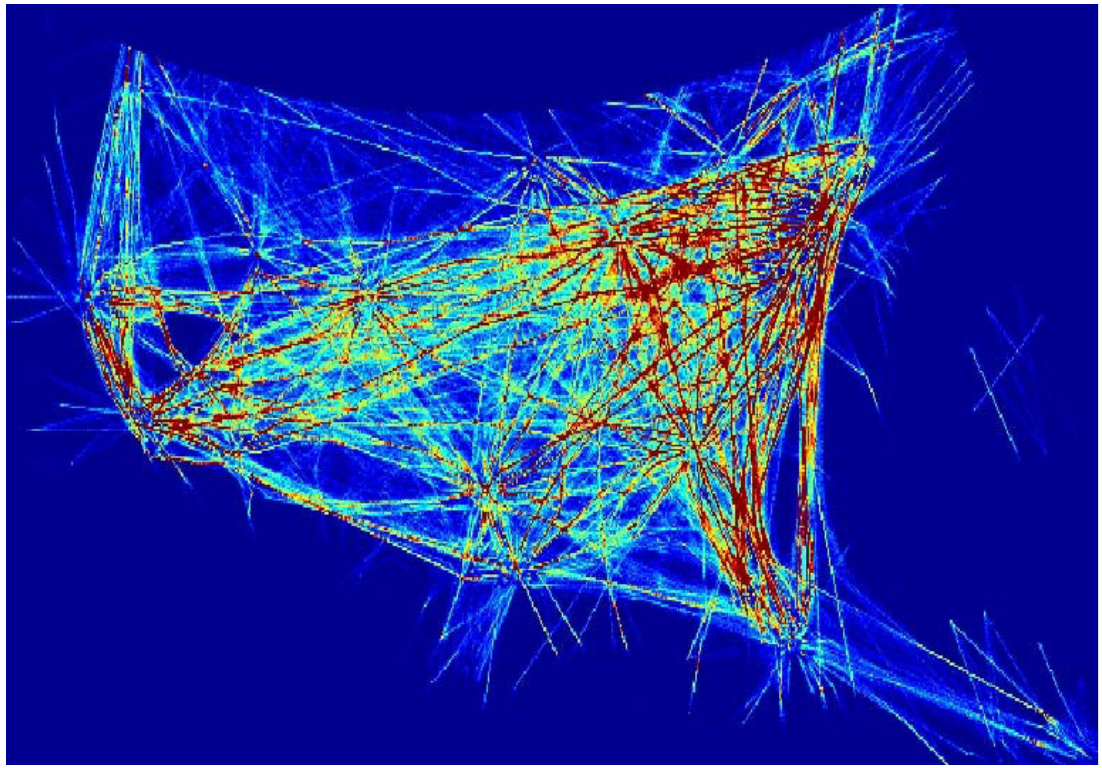
- Bottom-up
  - Cluster trajectories
  - Cluster elementary sector units (e.g., hex-cells) using MIP
- Top-down (recursive partition, BSP)
- Graph theoretic partitioning
- Genetic algorithms, neural networks
- Local re-partitioning
- Human-in-the-loop

Bottom-Up Methods

- Methods based on clustering of trajectory data
- Example:



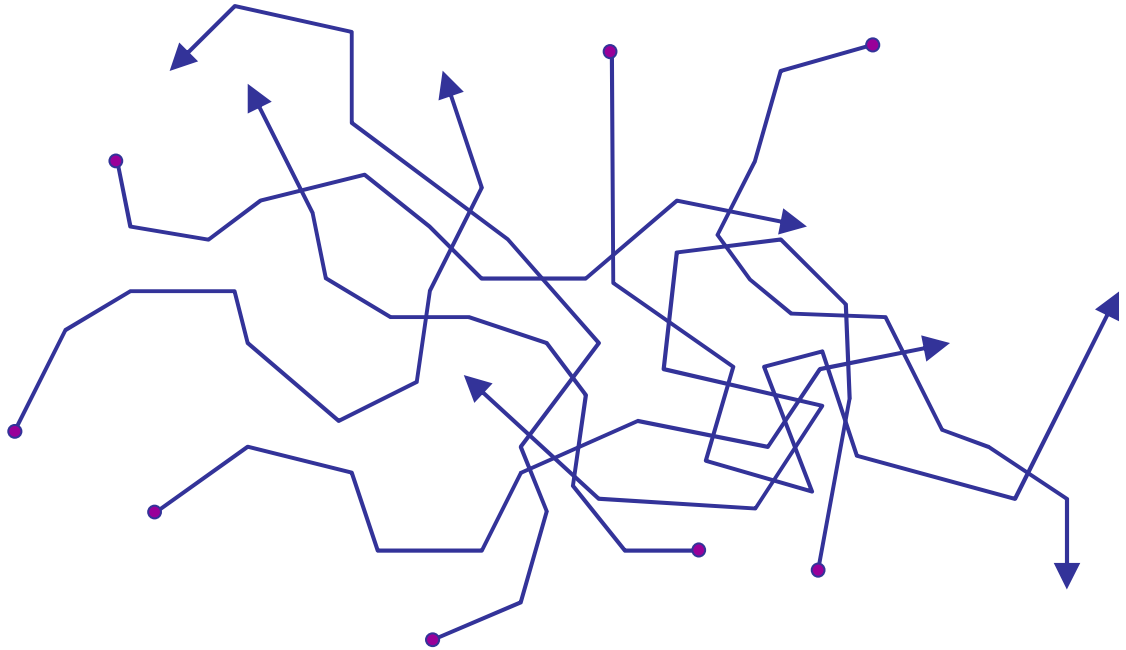
Clustering Trajectories: Discoverying Dominant Flows



A Weighted-Graph Approach for Dynamic Airspace Configuration Martinez, Chatterji, Sun, Bayen 2007

Patterns in Trajectories

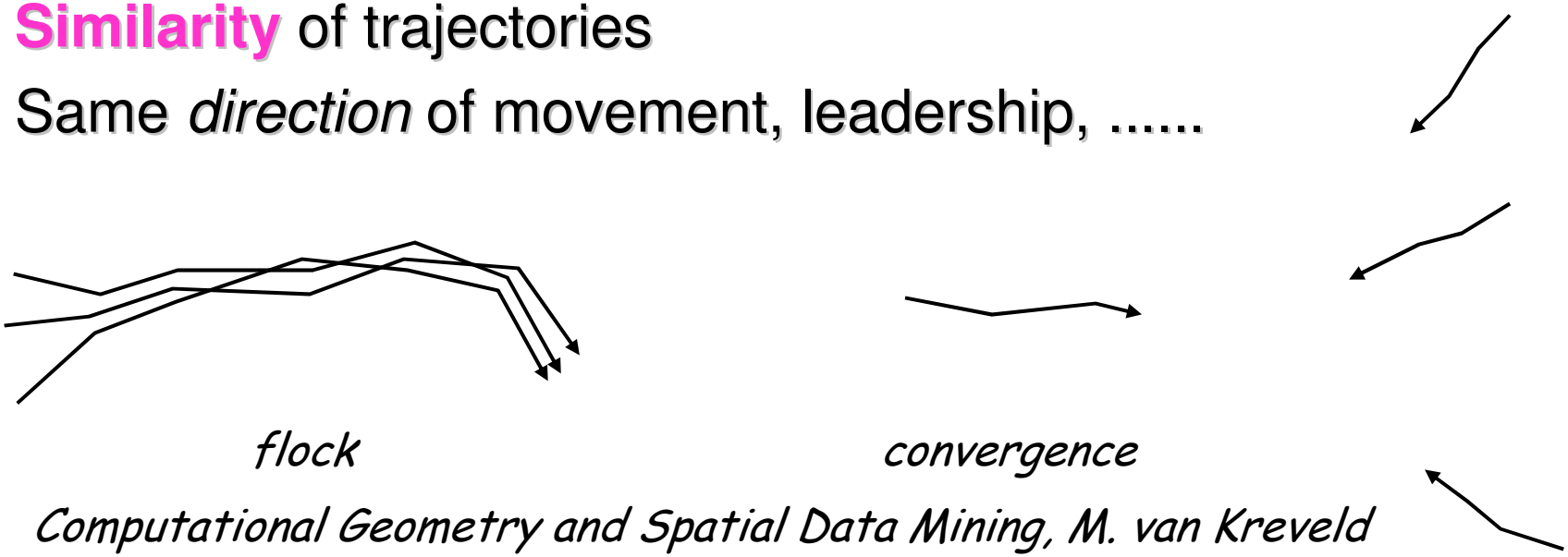
- *n* trajectories, each with *t* time steps
  → *n* polygonal lines with *t* vertices
- Already looked at *most visited location*



Computational Geometry and Spatial Data Mining, M. van Kreveld

Patterns in Trajectories

- **Flock**: near positions of (sub)trajectories for some subset of the entities during some time
- **Convergence**: same destination region for some subset of the entities
- **Encounter**: same destination region with same arrival time for some subset of the entities
- **Similarity** of trajectories
- Same *direction* of movement, leadership, ...



flock *convergence*

Computational Geometry and Spatial Data Mining, M. van Kreveld

Patterns in Trajectories

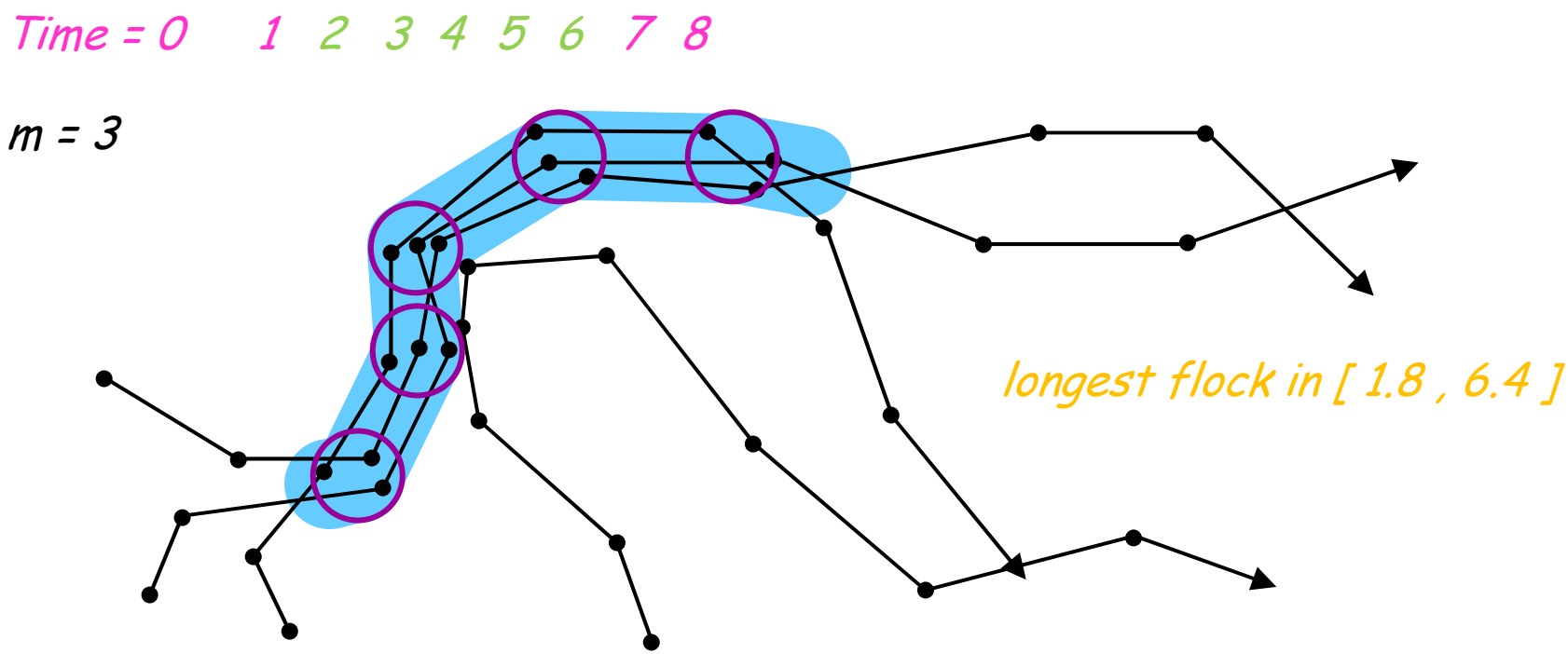
- Flocking, convergence, encounter patterns
  - Laube, van Kreveld, Imfeld (SDH 2004)
  - Gudmundsson, van Kreveld, Speckmann (ACM GIS 2004)
  - Benkert, Gudmundsson, Huebner, Wolle (ESA 2006)
  - ...
- Similarity of trajectories
  - Vlachos, Kollios, Gunopulos (ICDE 2002)
  - Shim, Chang (WAIM 2003)
  - ...
- Lifelines, motion mining, modeling motion
  - Mountain, Raper (GeoComputation 2001)
  - Kollios, Scaroff, Betke (DM&KD 2001)
  - Frank (GISDATA 8, 2001)
  - ...

Patterns in Trajectories

- **Flock**: near positions of (sub)trajectories for some subset of the entities during some time
  - clustering-type pattern
  - different definitions are used
- Given: radius r, subset size m, and duration T, a **flock** is a subset of size $\geq m$ that is inside a (moving) circle of radius r for a duration $\geq T$

Patterns in Trajectories

- **Longest flock:** given a radius *r* and subset size *m*, determine the longest time interval for which *m* entities were within each other's proximity (circle radius *r*)

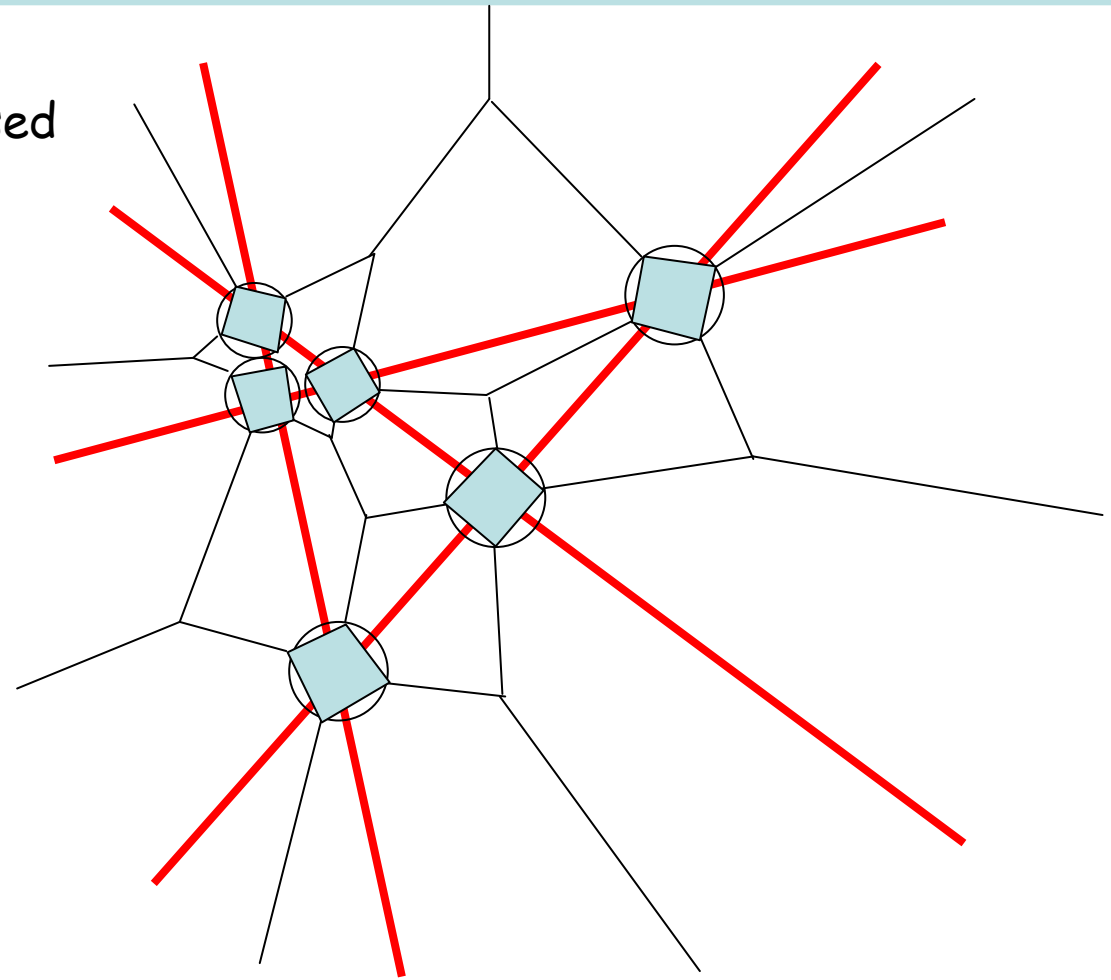


Computational Geometry and Spatial Data Mining, M. van Kreveld


Photo: C. Tonn

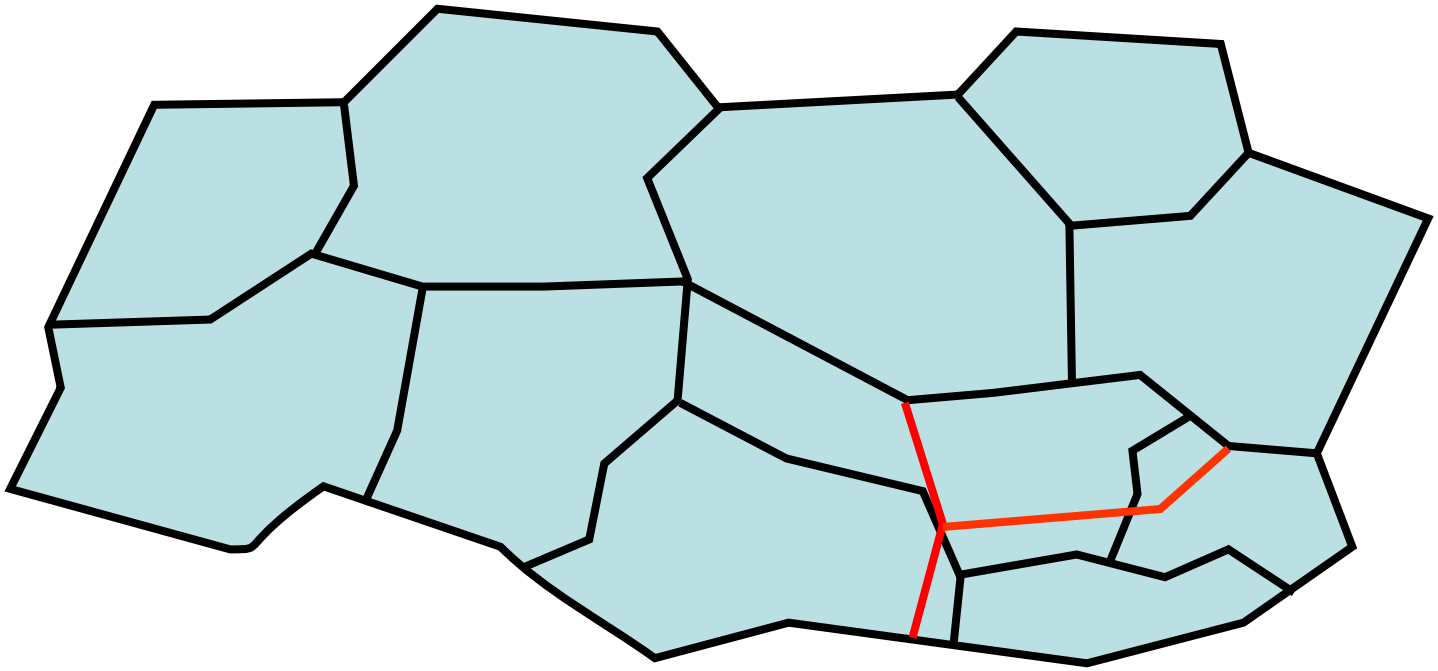
Voronoi-Based Partitioning

Voronoi-based partitioning



Another approach: Use arrangement of flow-conforming cuts to define elementary airspace units (vs. hexcells)

Local Re-Partitioning



Re-Partition step replaces one region with a "best" partition into a k-face planar subdivision

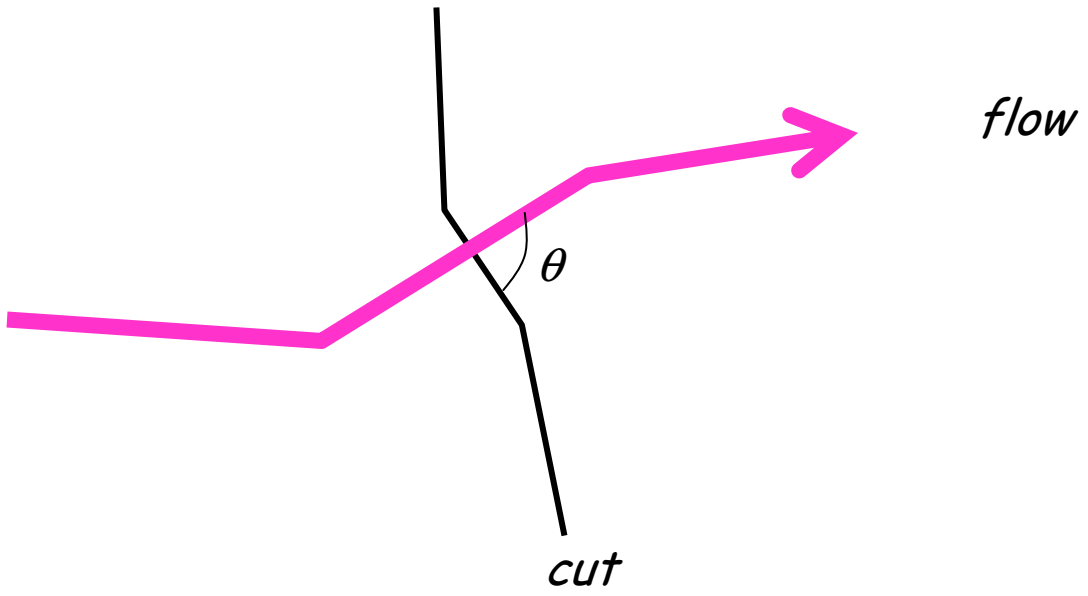
Computational Geometry for DAC: GeoSect Methodology

Joseph S. B. Mitchell and Girishkumar Sabhnani
Stony Brook University

STONY BROOK
STATE UNIVERSITY OF NEW YORK

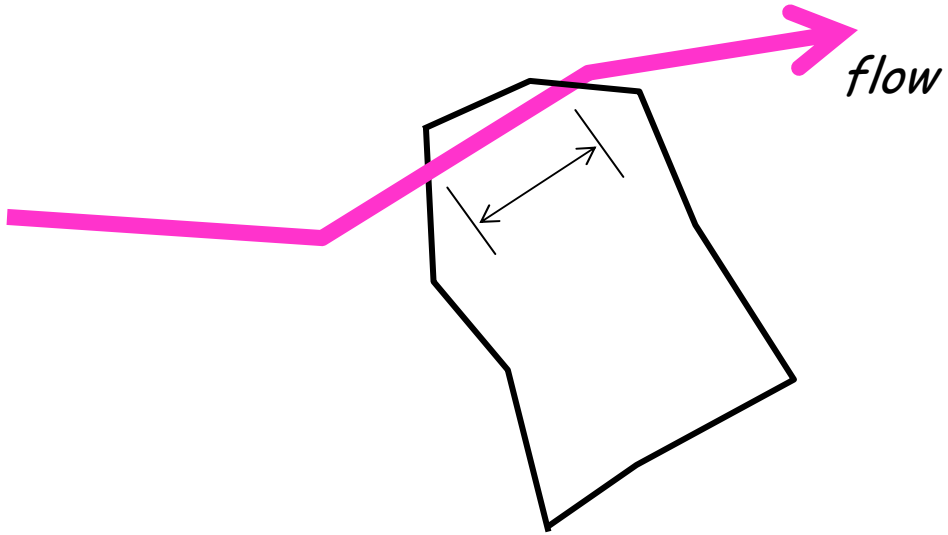
Conforming to Dominant Flows

- How should a polygonal cut behave wrt dominant flows?
  - (1) Dominant flows should cross cut nearly orthogonally



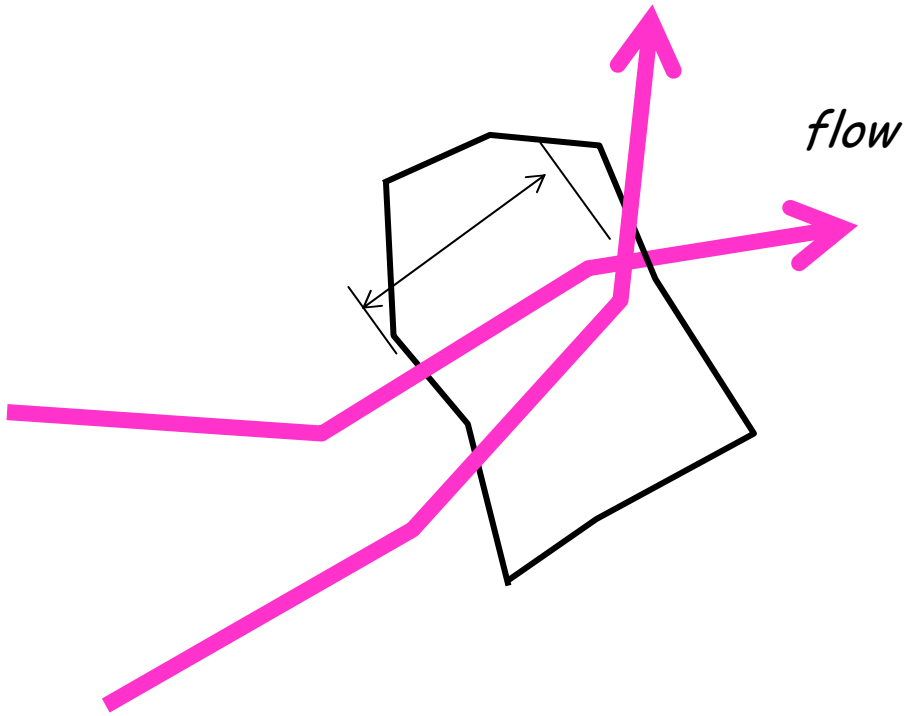
Conforming to Dominant Flows

– (2) Dominant flows should spend at least a minimum amount of time in a sector before leaving



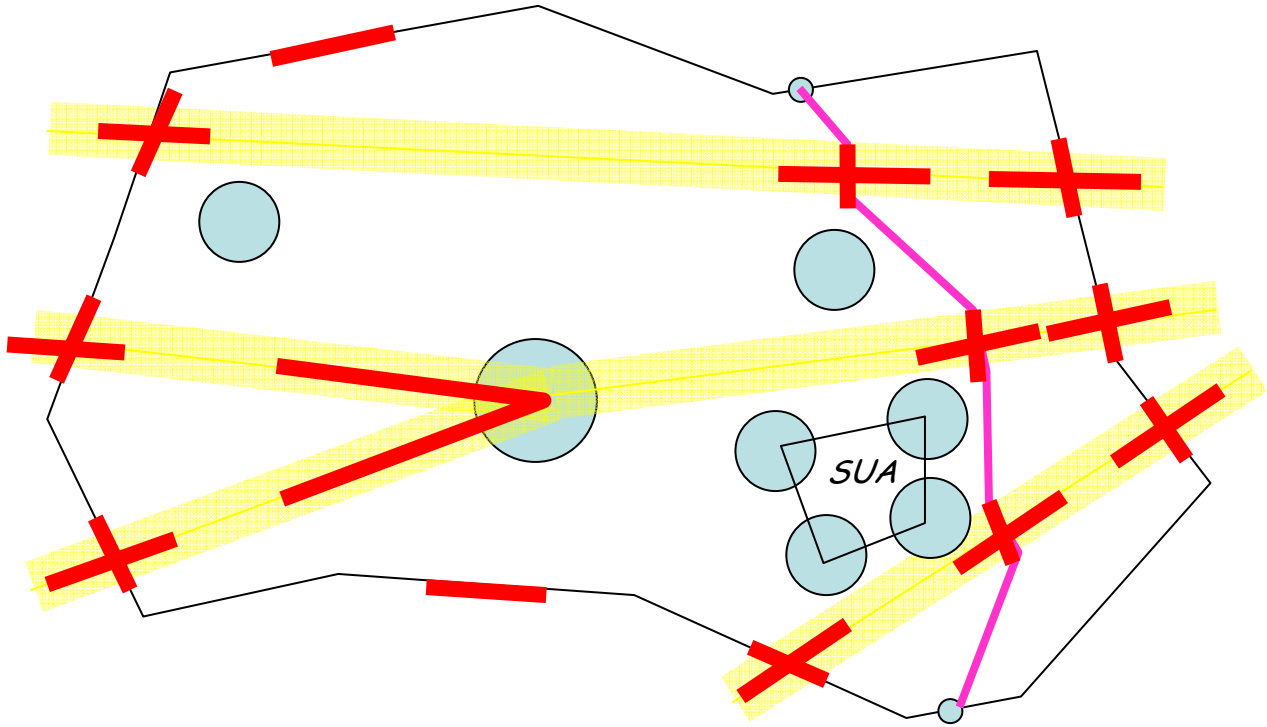
Conforming to Dominant Flows

– (3) Merge points and route intersections should have ample distance from upstream entry point of sector



Flow-Conforming Cut Problem

- Given a set of dominant flows, we compute a polygonal cut that conforms to the flow, satisfying all constraints:

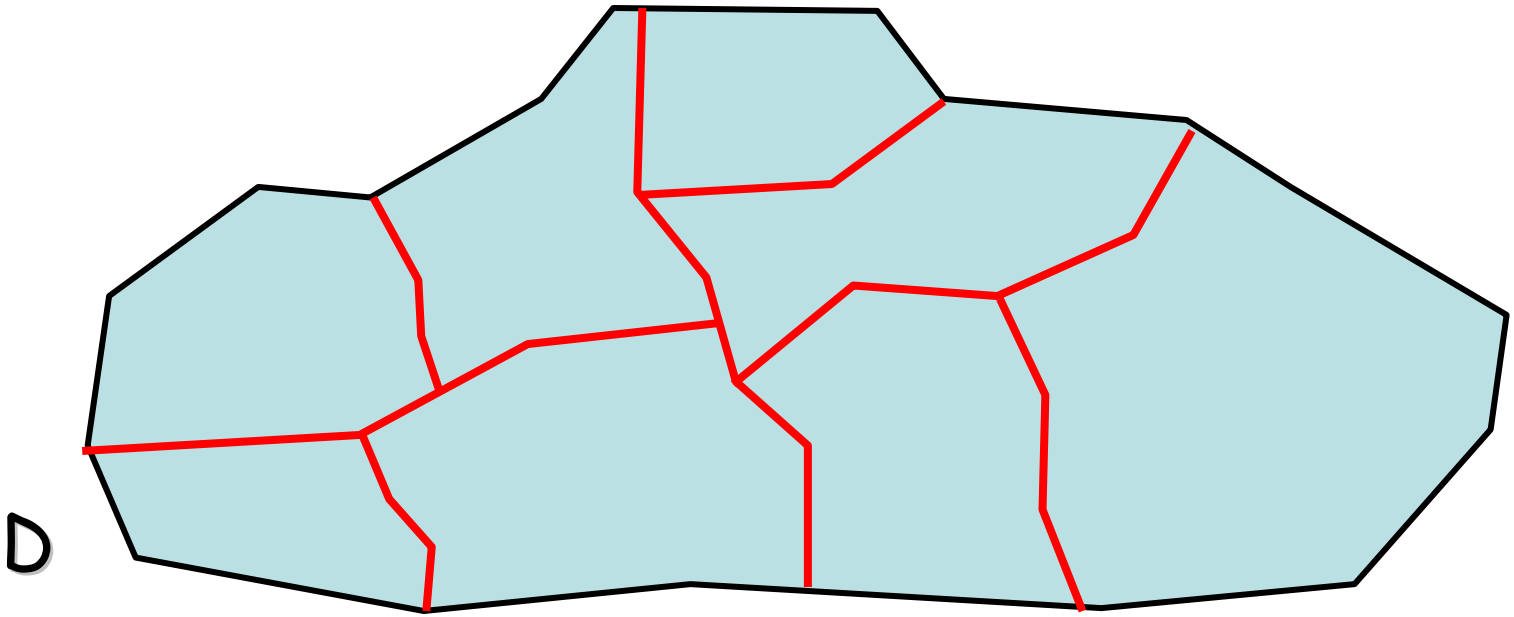


Key: Every candidate cut is guaranteed to conform to dominant flows.

More General BSP

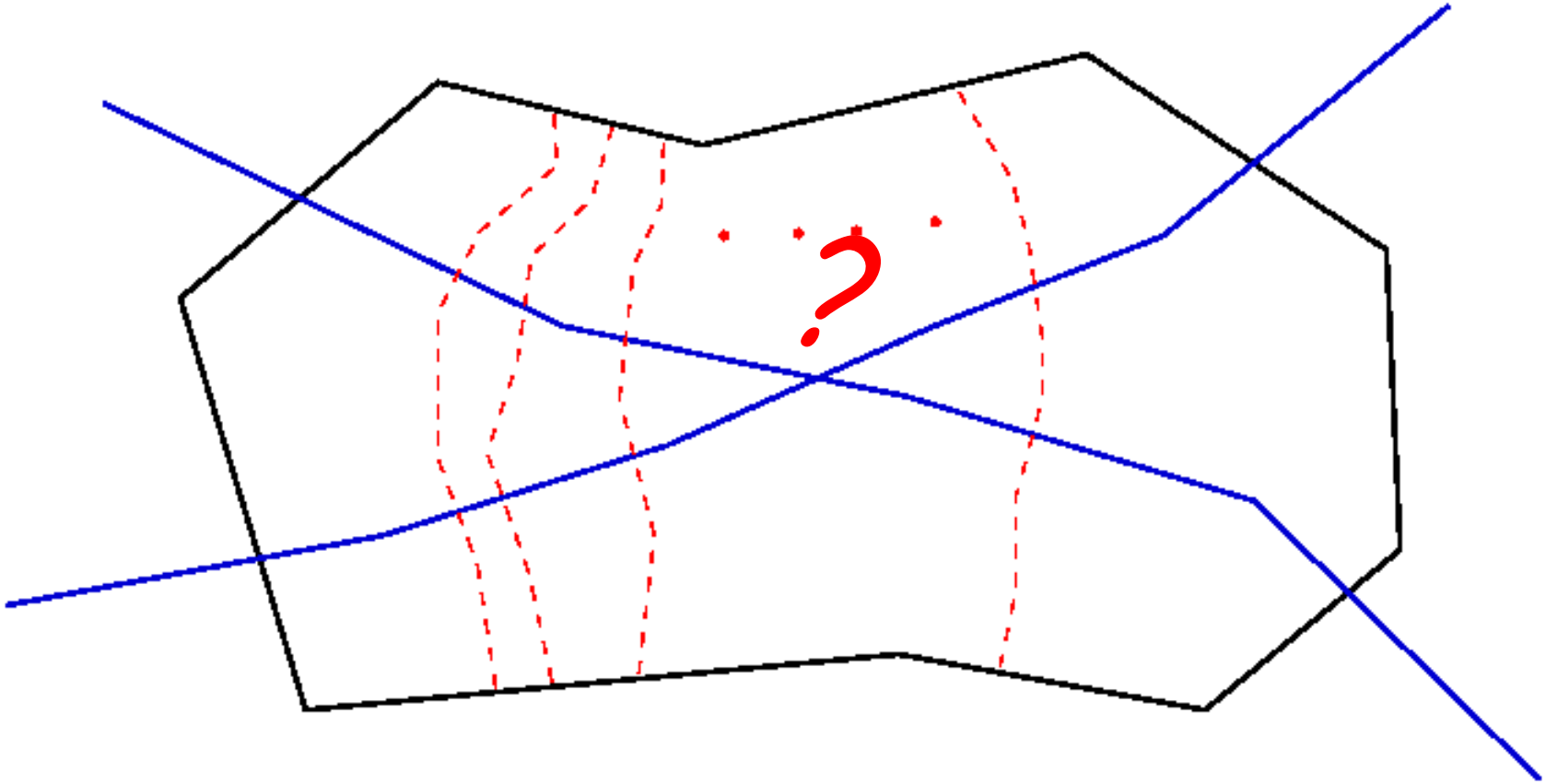
Each cut is a flow-conforming path

Workload Evaluator for each subregion.



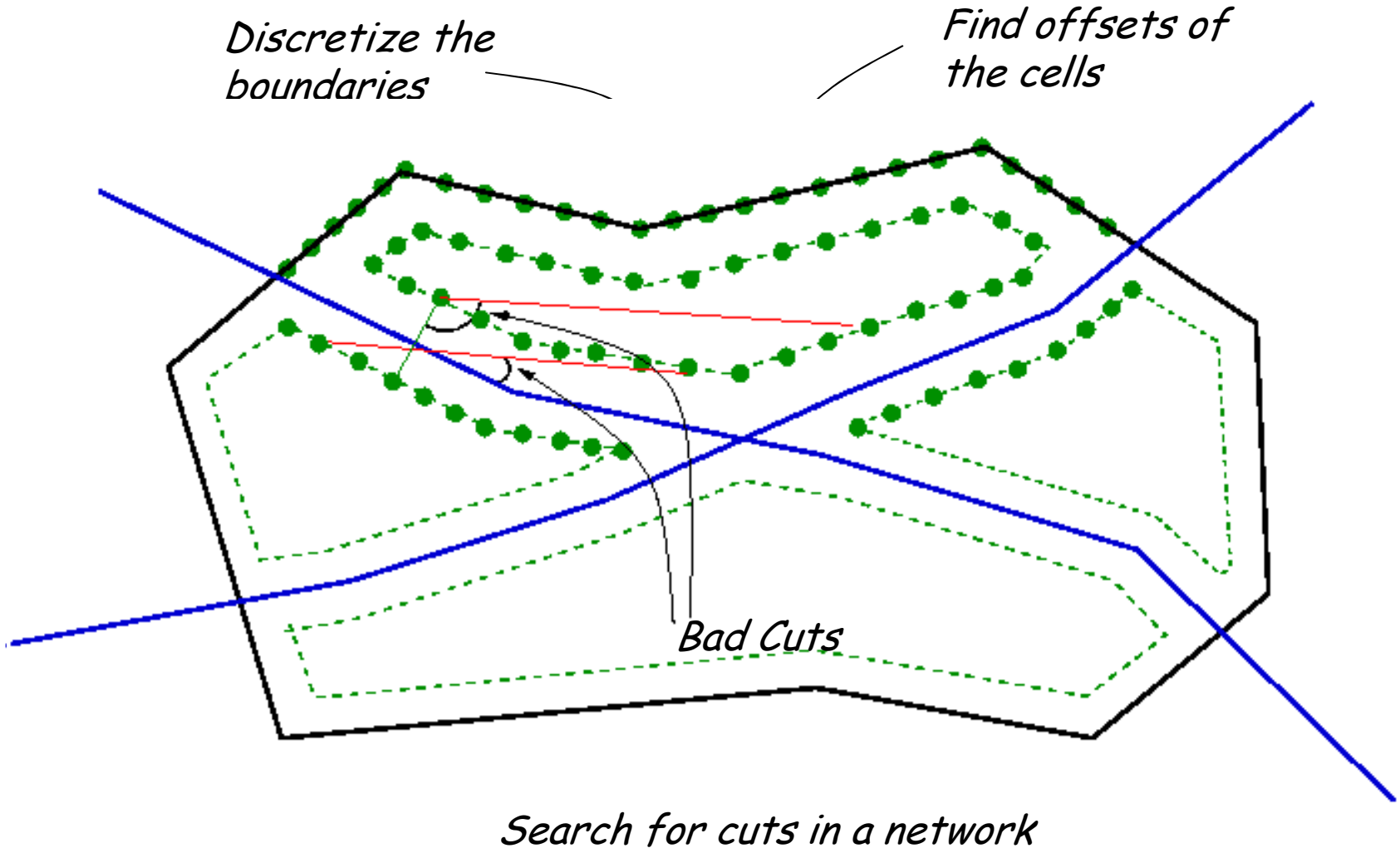
Basic Idea : Modification to BSP

Search for a good flow-conforming cut
by sweeping with a set of flow-conforming cuts

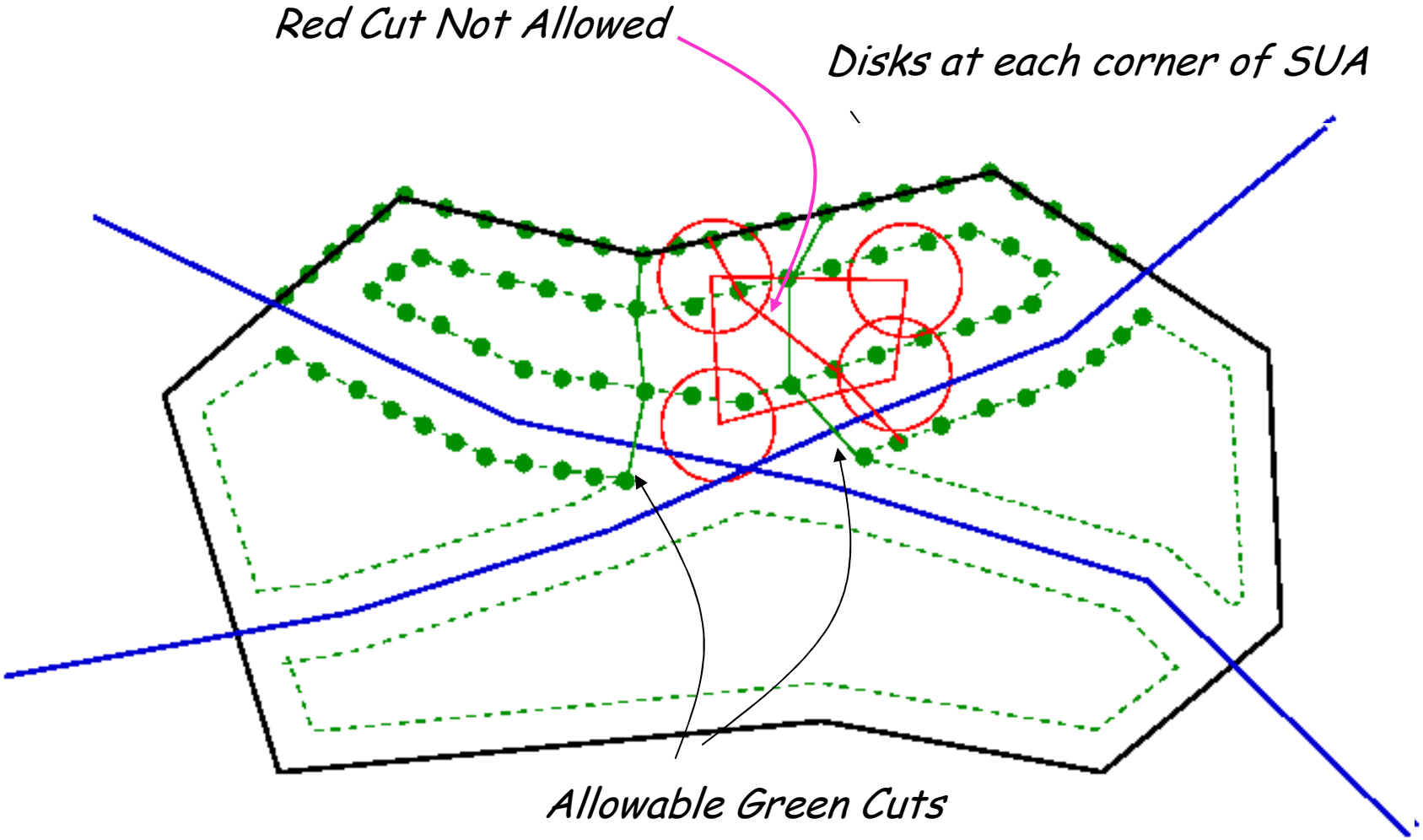


How to find these flow-conforming cuts?

Finding Good Candidate Cuts for BSP

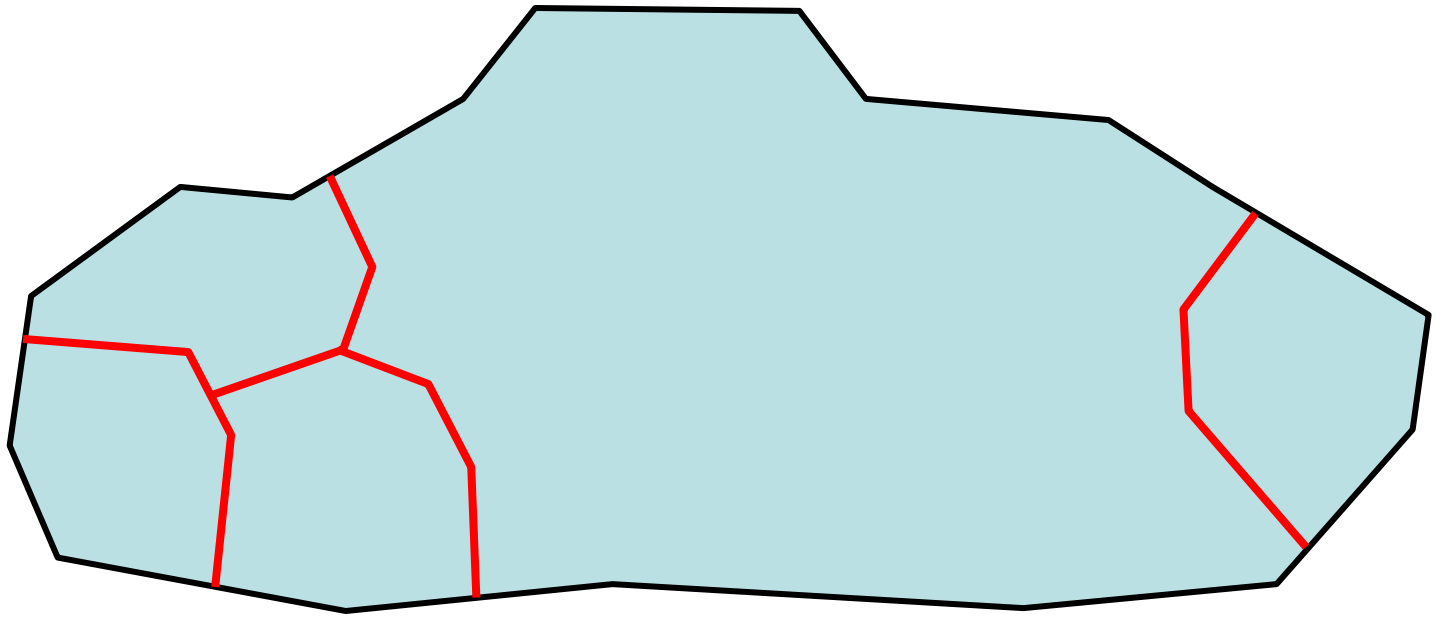


Conforming to SUA's



Another Approach

- Biting off individual sectors



Guarantees of Resulting Sectors

- Flows cross the sector boundaries “almost orthogonally”
- Angles at sector vertices are “big” (not acute)
- Each conflict point at crossings of dominant flows lies considerably within the sector boundaries
- The dominant flows never corner clip the sector boundary
- SUAs either lie fully inside one sector or lie favorable with respect to two or more sectors they cross

Each of these conditions is a parameterized constraint, depending on user-specified parameters.

Dynamic part of DAC

- Optimal determination of airspace adjustment triggers
- Tradeoff: Cost of adjustment, benefit from change

Dynamic Sectorization

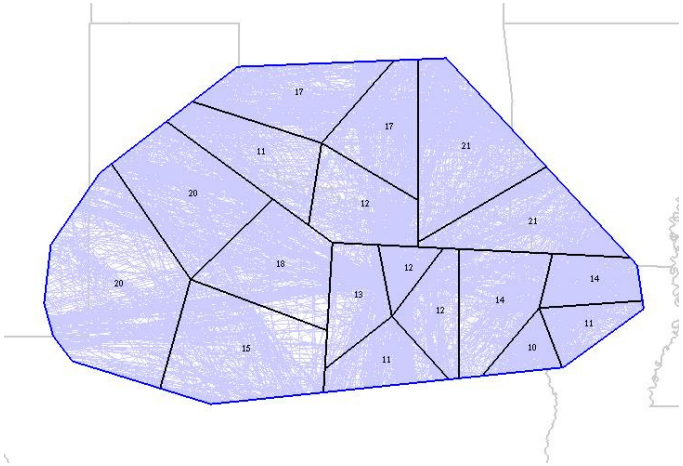
- Multiple Sector Designs for different times of the day

- How to find the optimal time of switching between sector designs ?

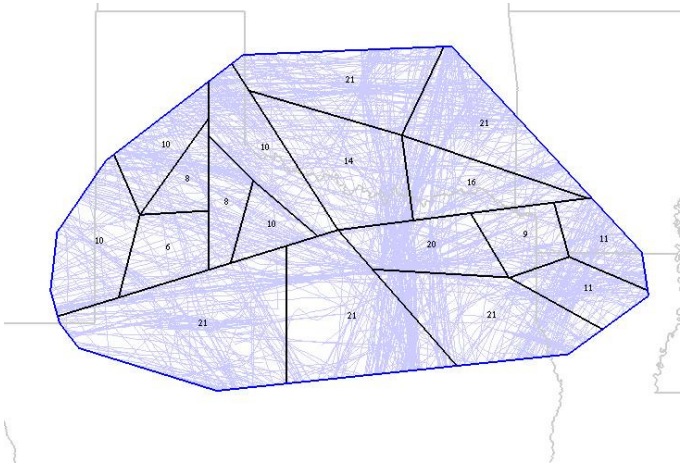
Dynamic Programming after discretizing the switch times

Applies to any sector design method (e.g., MIP methods).

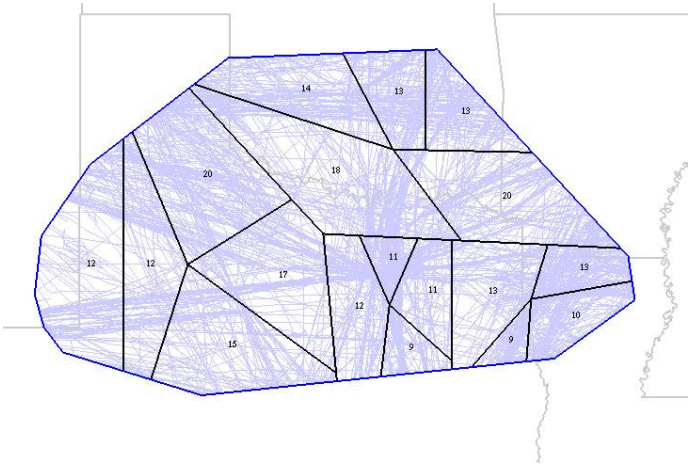
Multiple Sector Designs



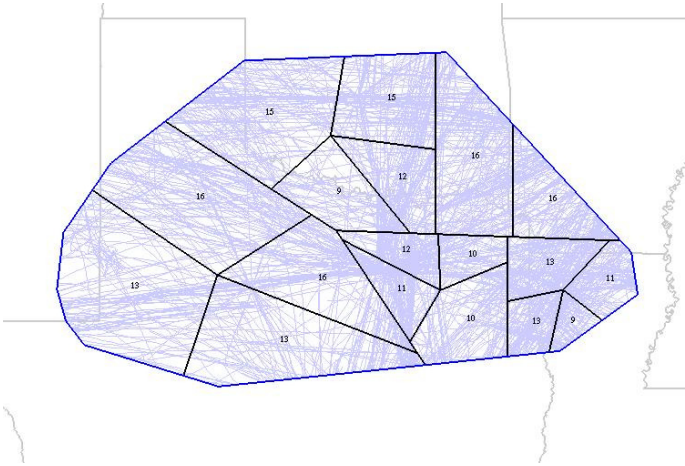
Whole Day



7:00 to 17:00



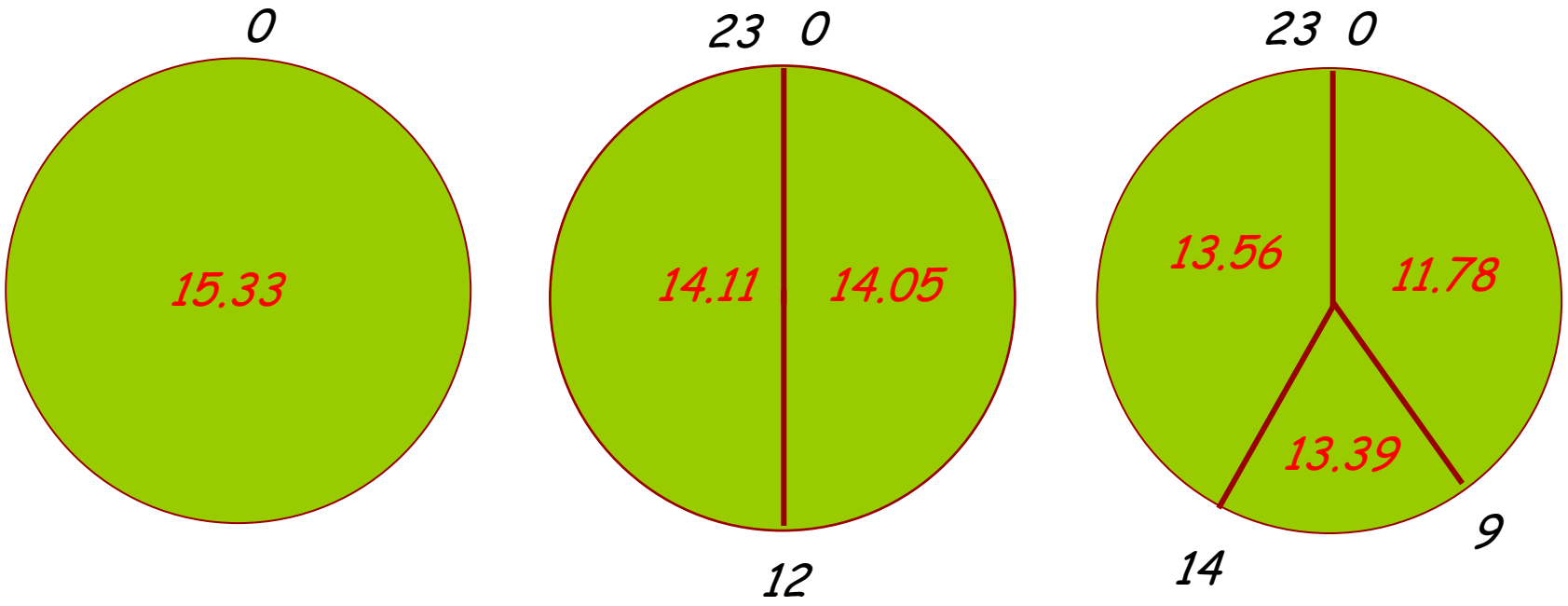
17:00 to 22:00



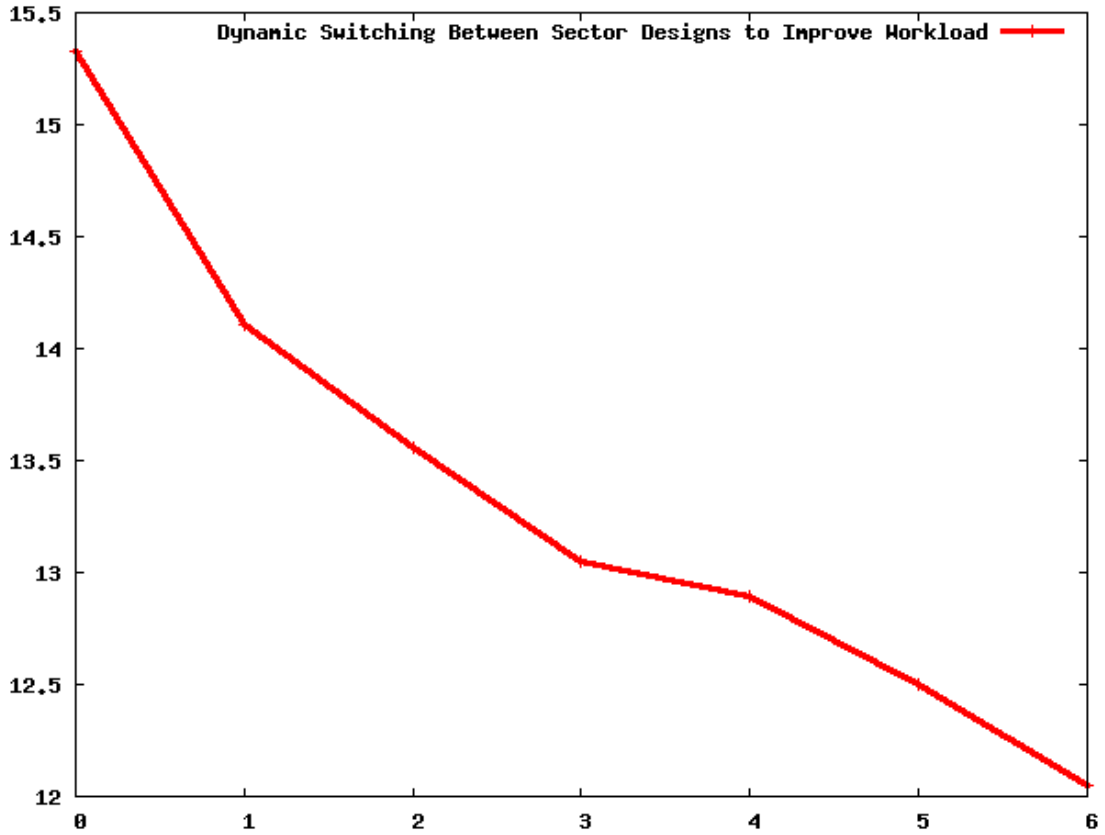
22:00 to 7:00

Results: Dynamic Sectorization

- Sectorizing ZFW (18 Sectors)
- Discretizing a day into 24 intervals (hourly splits)



Tradeoff: Workload Improvement, Number of Redesigns



Average Workload

Number of design changes during the day

Robustness of Designs

- Challenge: Formalize the notion of robustness with respect to forecast of demand

Current Efforts

- Integration with the Sector Optimization Tool (SOT) – optimization-based approach using the power of combinatorial optimization
  - Optimize sector smoothing
  - Devise hybrid method?
- Implement additional types of “cuts” in the BSP decomposition (*non*-convex sectors)
- Bottom-Up methods
- Optimal tube network design

Designing Configuration Playbooks

- Goal: Identify good configurations corresponding to mined historical data scenarios
- Rationale: Certain traffic patterns may tend to repeat over different time intervals, in response to certain events (e.g., weather impact)
- What time intervals? What events?
- Clustering, mining trajectory data

Optimal Design of Tubes

- Flexible airspace design
- Dynamic Airspace Configuration
  - Network of “tubes”, similar to highways
    - Dynamically designed/optimized
    - High volume, multiple lanes
    - Equipage requirements (navigation, communication)

Design of Tubes: Parameters

- Cross section: width (# lanes), height (# levels)
  - Flight (equipage) characteristics per lane/level
- Merge/split points?
  - # and separation

- On/off ramps, and separation standards: upper/lower bounds on λ_{off} , λ_{on} , λ_b
-