

Dynamic Airspace Sectorization using Controller Task Load

I. Gerdes, A. Temme, M. Schultz

SESAR Innovation Days 2016

10.11.2016



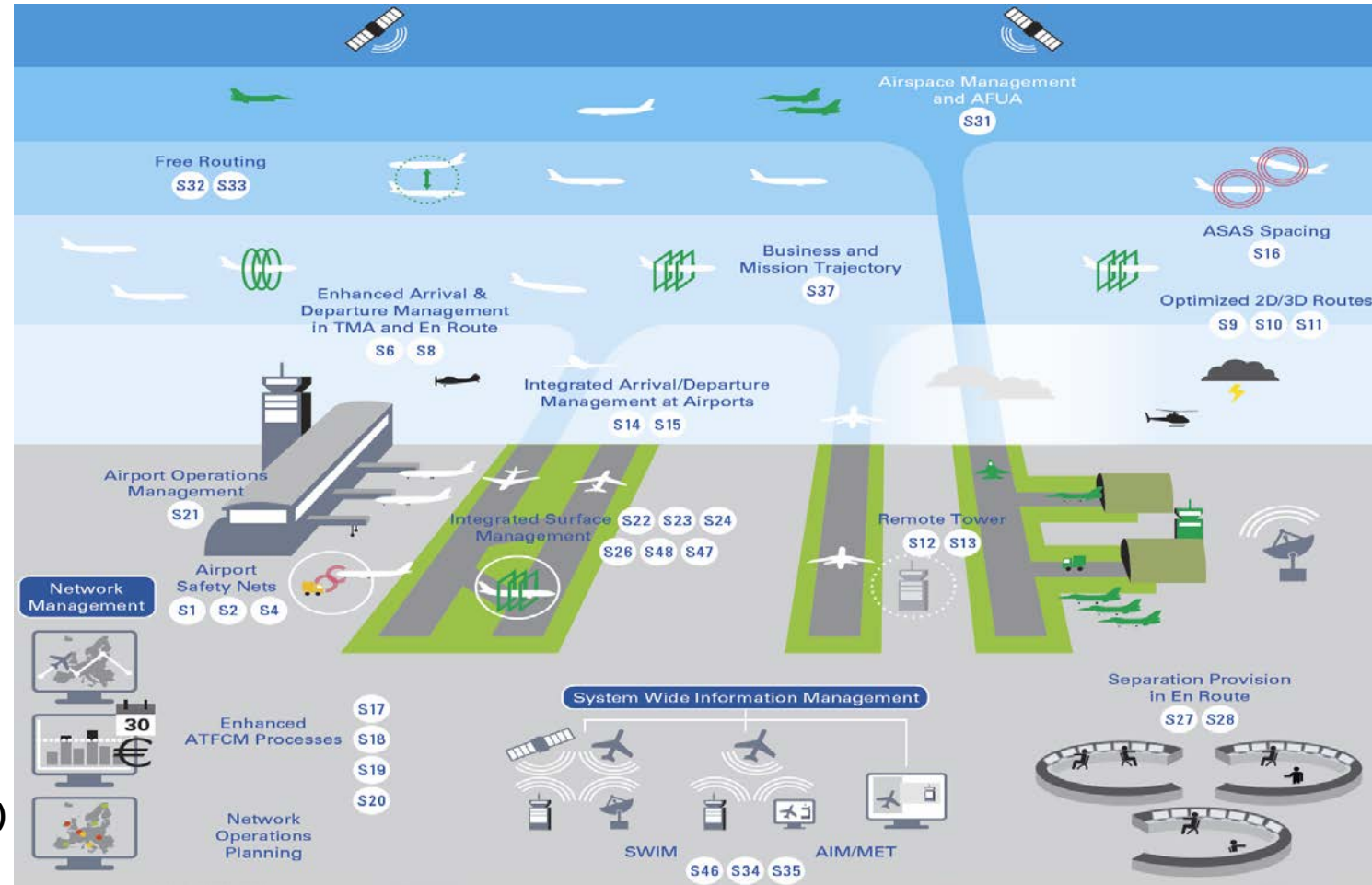
Wissen für Morgen



SESAR

solutions at current release 5

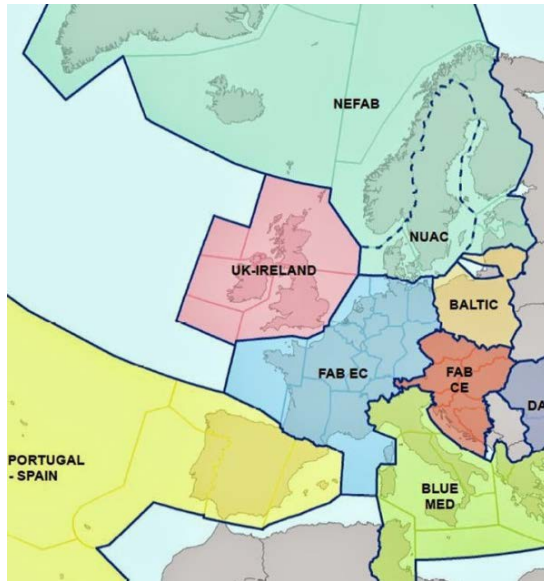
- Airport Integration & Throughput
 - S1 Runway Status Light
 - S2 Airport Safety Nets for controllers ...
 - S4 Enhanced Traffic Situational Awareness ...
 - S12 Single Remote Tower operations ..
 - S13 Remotely-Provided Air Traffic Service ...
 - S21 Airport Operations Plan (AOP) ...
 - S22 Automated Assistance to Controller ...
 - ...
- Network Collaborative Management
 - S17 Advanced Short ATFCM Measures ...
 - S18 Calculated Take-Off Time (CTOT) ...
 - S19 Automated support for Traffic ...
 - S20 Collaborative NOP for Step 1
 - S31 Variable profile military reserved ...
- SWIM
 - S34 Digital Integrated Briefing
 - S35 MET Information Exchange
 - S46 Initial SWIM
- Moving from airspace to 4D trajectory management
 - S32 Free Route through the use of Direct Routing
 - S33 Free Route through Free Routing (cruise/vertical)
 - S37 Extended Flight Plan



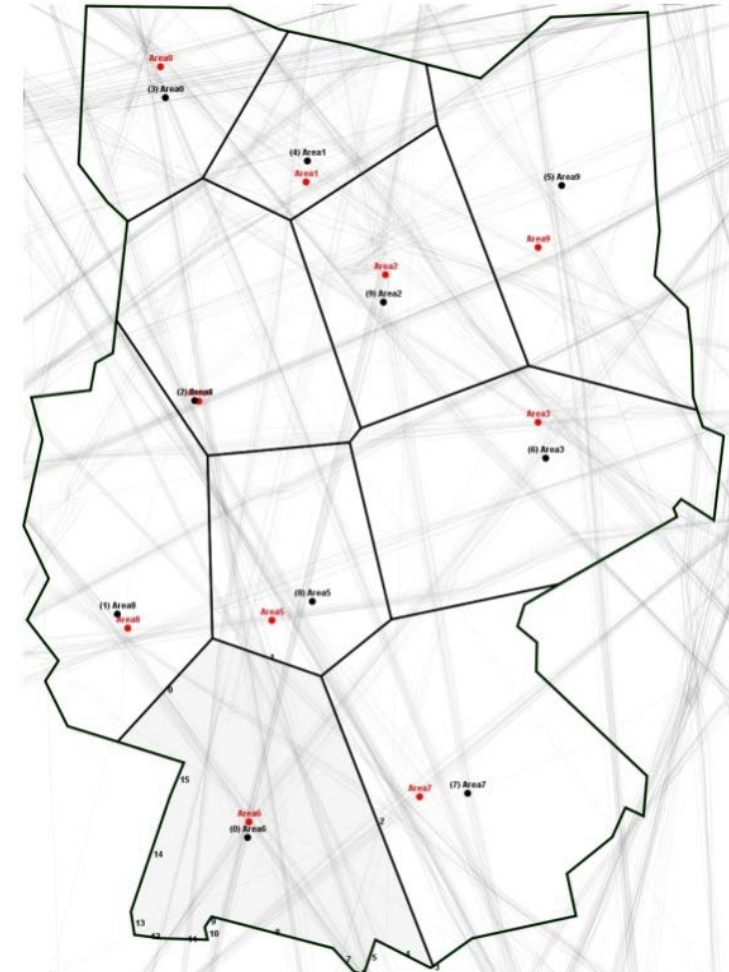
Advanced Concepts in ATM

dynamic airspace sectorization

- FABEC - economy of scale ?



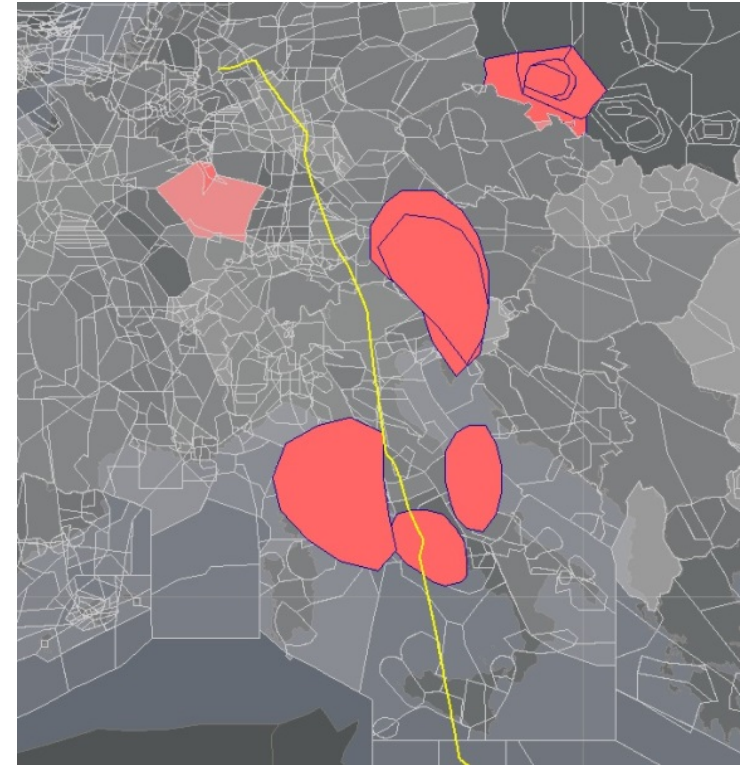
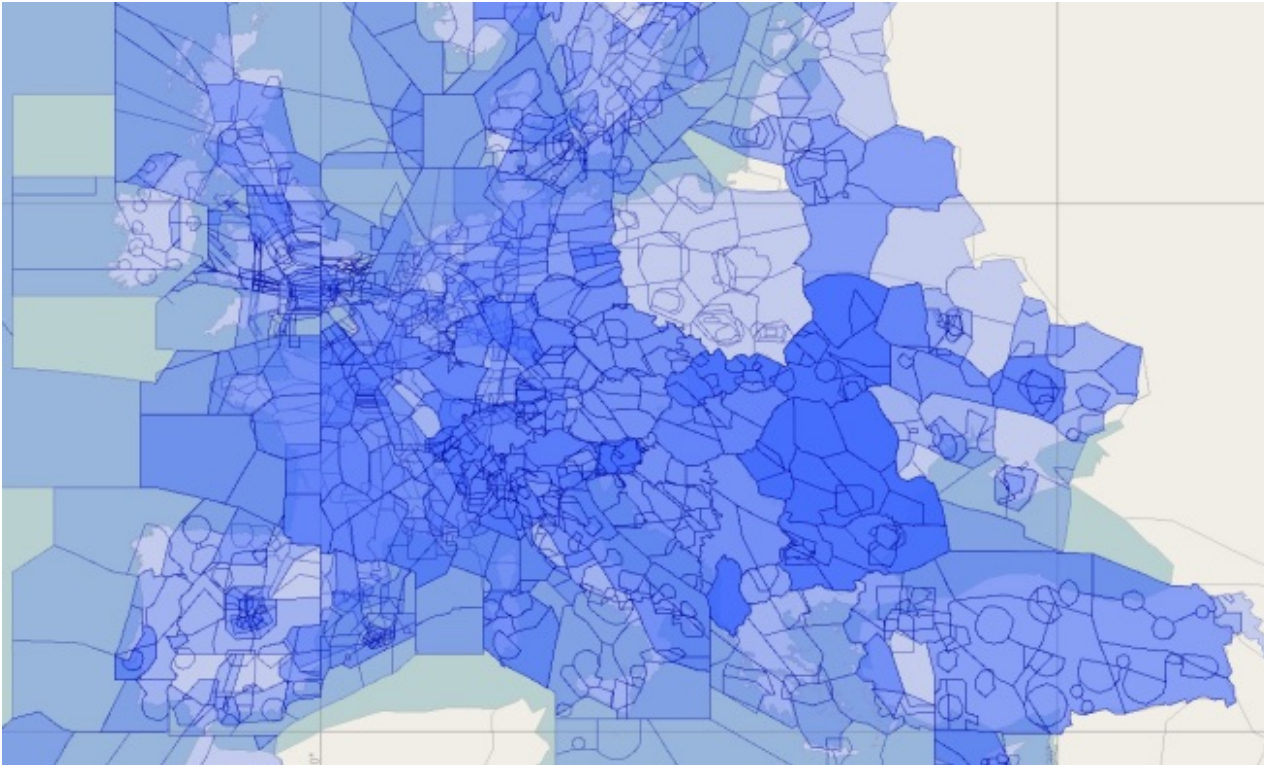
- Idea
 - Traditionally: flow follows structure
 - New: structure follows flow
 - Methods: clustering, evolutionary algorithms



Test - Operational scenario

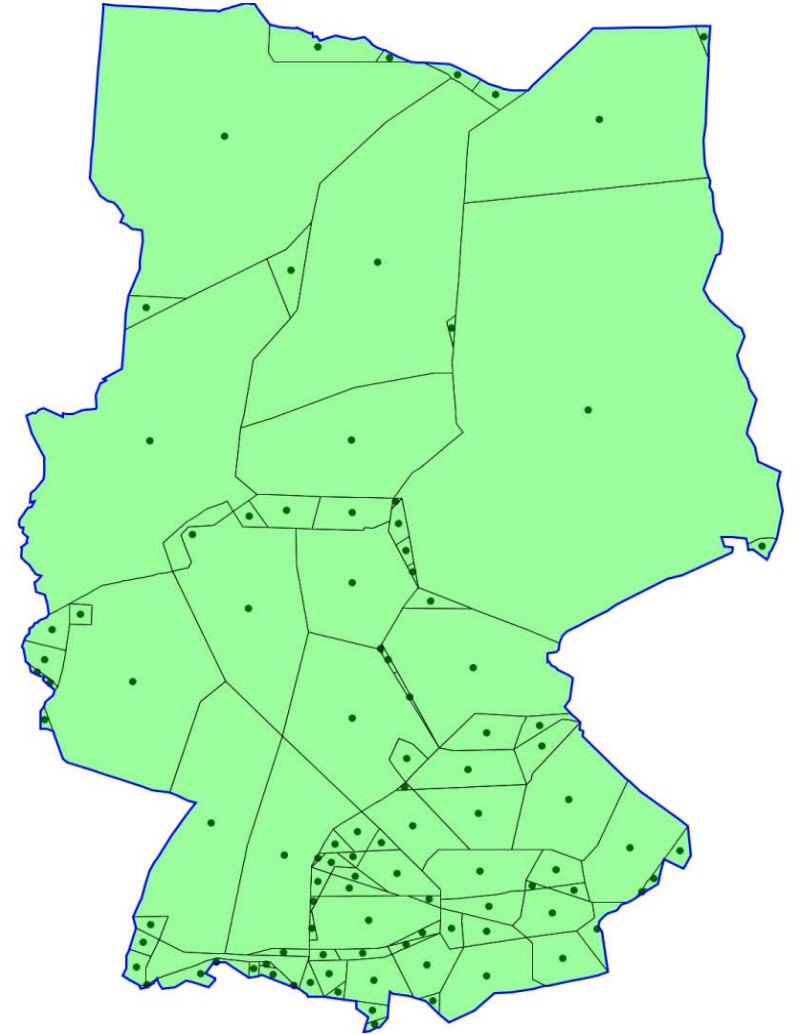
Air Traffic Management

European examples



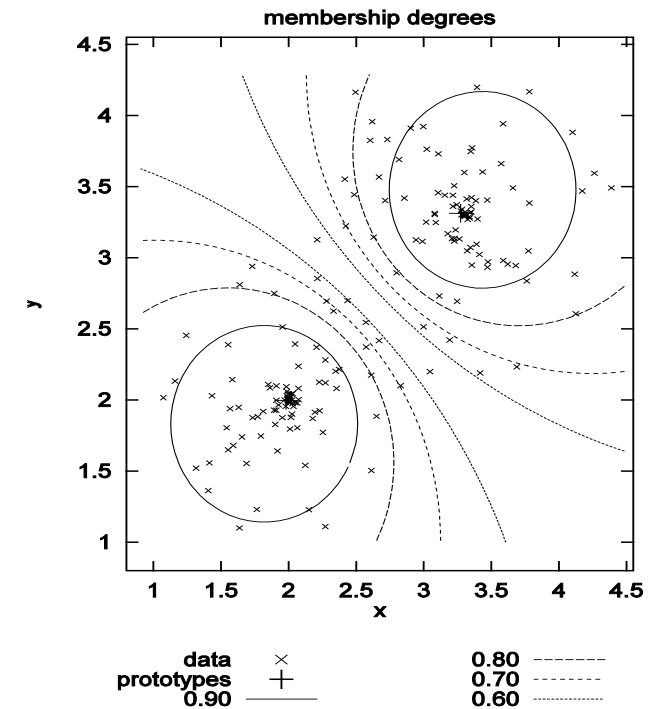
Research Target

- higher flexibility at airspace sectorization considering traffic density
- Adaption to
 - changing traffic demands over the day with
 - smooth transition between succeeding sectorization
- Balancing
 - traffic load of sectors
- Transition
 - combine the unstructured/sectorless airspace approach with the rigid structures of today
- Three-step Approach:
 - **Fuzzy Clustering** – hot spots
 - **Voronoi-diagrams** – initial structure
 - **Evolutionary Algorithms** – optimization



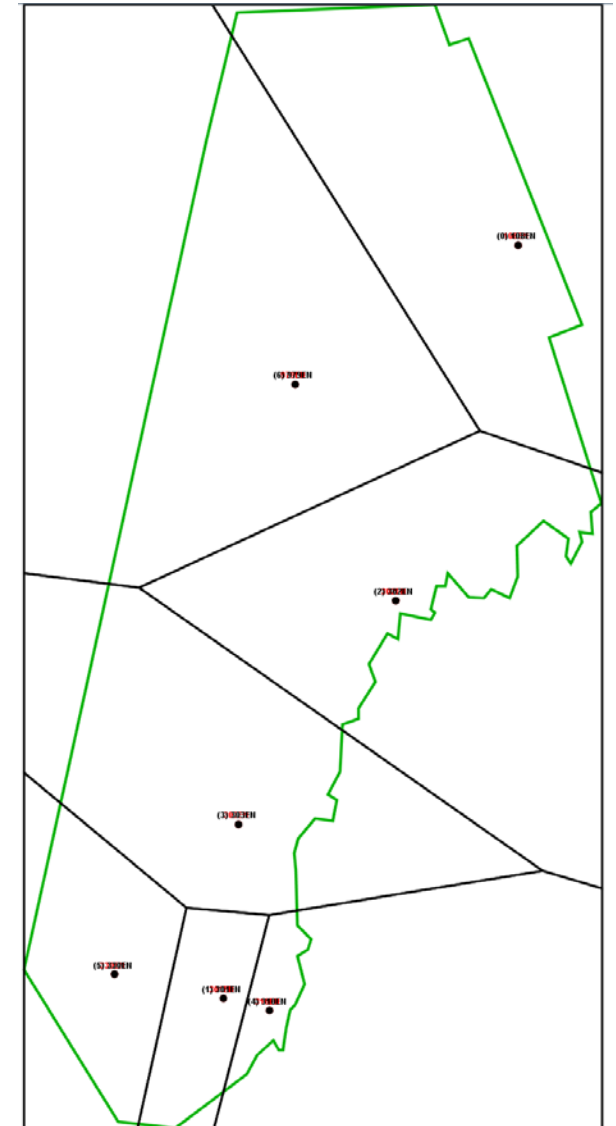
Approach

- Fuzzy Clustering
 - Partitioning of data into subsets using predefined criterias
 - Creation of data centers (center point) for each subset

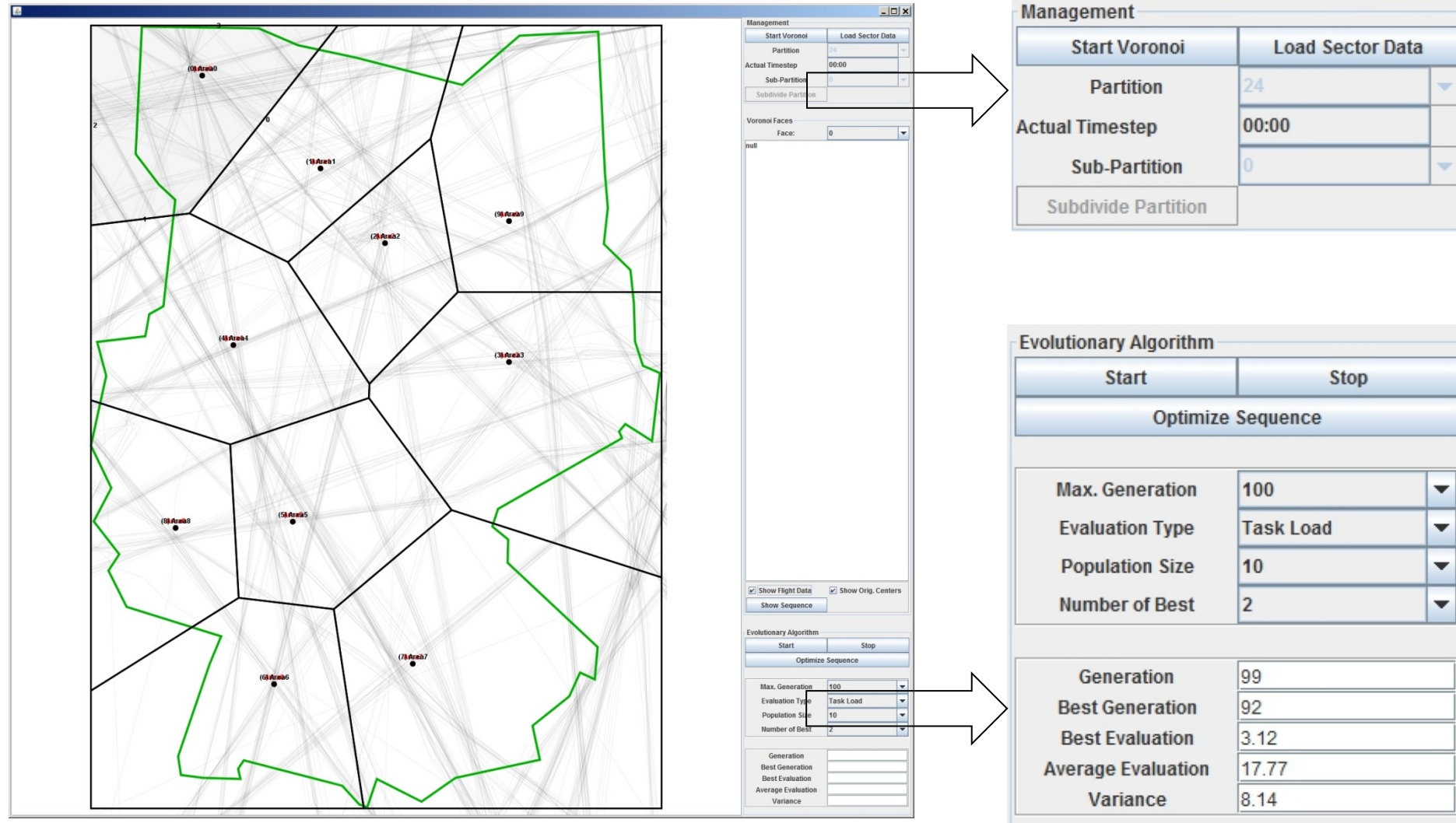


Approach

- Fuzzy Clustering
 - Partitioning of data into subsets using predefined criterias
 - Creation of data centers (center point) for each subset
- Voronoi Diagrams
 - Structure the airspace by defining edges consisting of all points having the same distance to two center points
 - Vertices are created where 3 edges collide
- Evolutionary Algorithms
 - Storing a fix number of possible solutions for a problem (population) in a structure (mimics a genetic chromosome)
 - Applying operators for mutating the chromosome or exchanging information between two chromosomes
 - Run through several generations by selecting chromosomes for the next generation in dependence of their problem solving quality

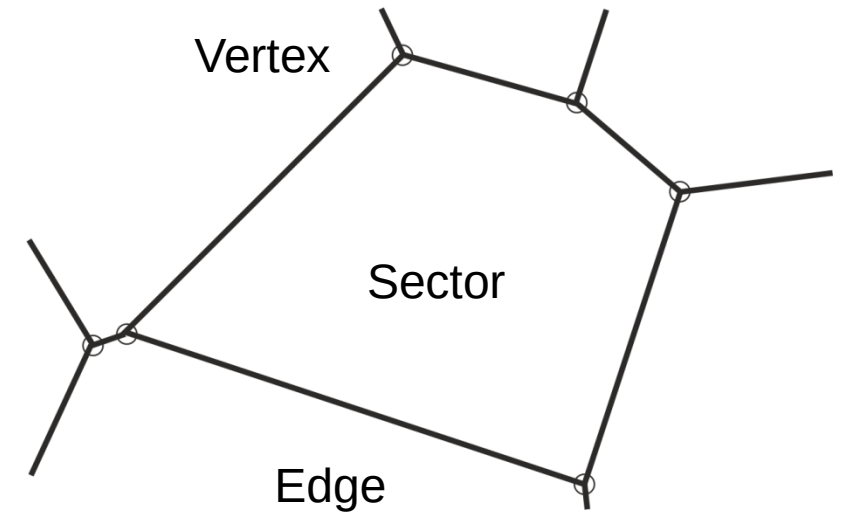


Tool: AutoSec



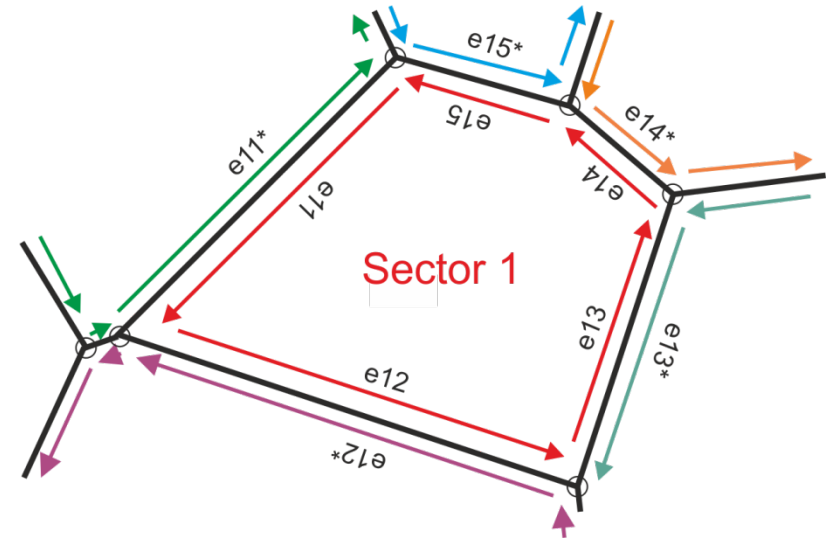
Data Structure

- As data structure for the Voronoi diagram a DCEL (Doubly Connected Edge List) was used
- A DCEL consist of three lists which are connected by pointers, one for vertices, one for edges and one for the sectors



Data Structure

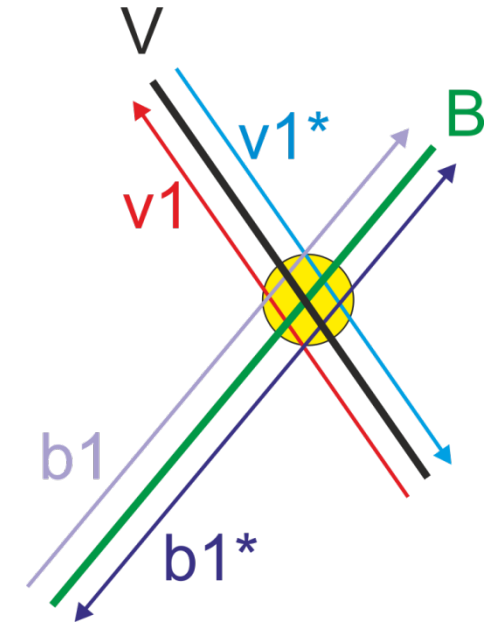
- As data structure for the Voronoi diagram a DCEL (Doubly Connected Edge List) was used
- A DCEL consist of three lists which are connected by pointers, one for vertices, one for edges and one for the sectors
- Each undirected edge is divided into a pair of directed edges with opposite directions
- Each sector is created by the sequence of half-edges in counter clockwise direction
- Each half-edge is connected to the previous and next half-edge by pointers



Implementation

adaption to non-convex borders

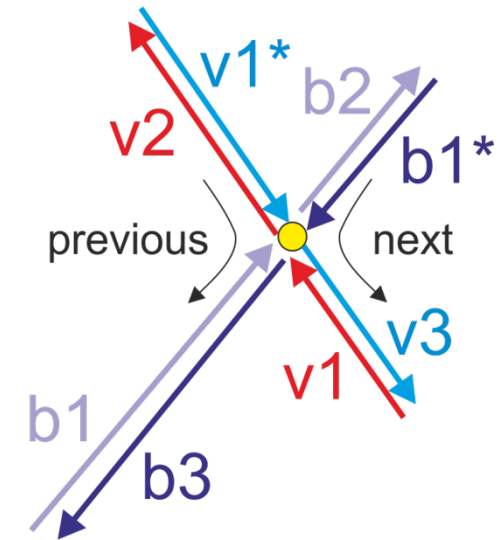
- Transformation of the structure of the border-polygon into a DCEL.
- Copy both DCELs into a common DCEL (Overlay).
- Calculation of breakpoints between all half-edges of Voronoi and border DCEL and ordering these breakpoints with increasing y-coordinate.



Implementation

adaption to non-convex borders

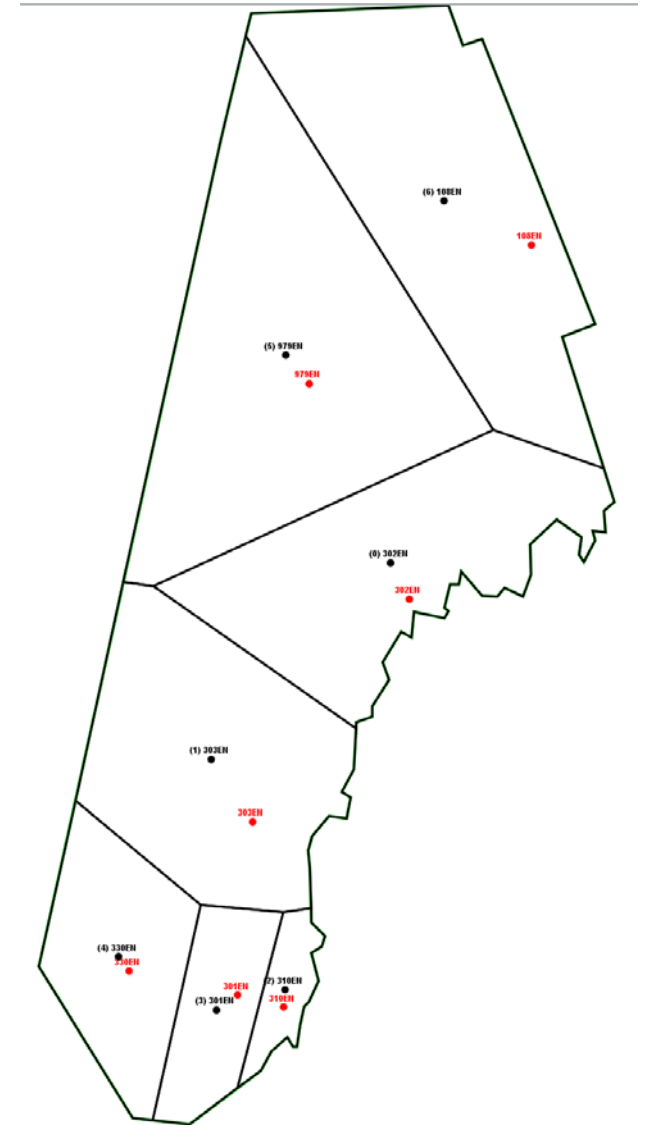
- Transformation of the structure of the border-polygon into a DCEL.
- Copy both DCELs into a common DCEL (Overlay).
- Calculation of breakpoints between all half-edges of Voronoi and border DCEL and ordering these breakpoints with increasing y-coordinate.
- Move through the breakpoint list and reconstruct the pointer for the affected half-edges and vertices.



Implementation

adaption to non-convex borders

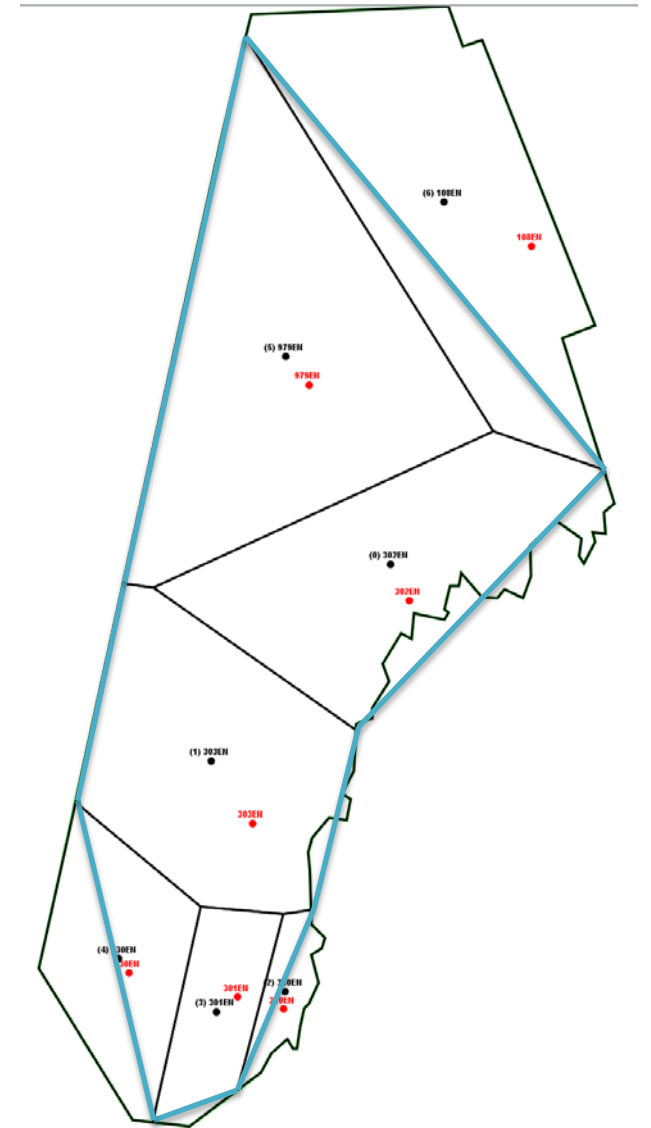
- Transformation of the structure of the border-polygon into a DCEL.
- Copy both DCELs into a common DCEL (Overlay).
- Calculation of breakpoints between all half-edges of Voronoi and border DCEL and ordering these breakpoints with increasing y-coordinate.
- Move through the breakpoint list and reconstruct the pointer for the affected half-edges and vertices.
- Create new sectors based on the reconstruction and remove all half-edges and vertices outside the border polygon.



Implementation

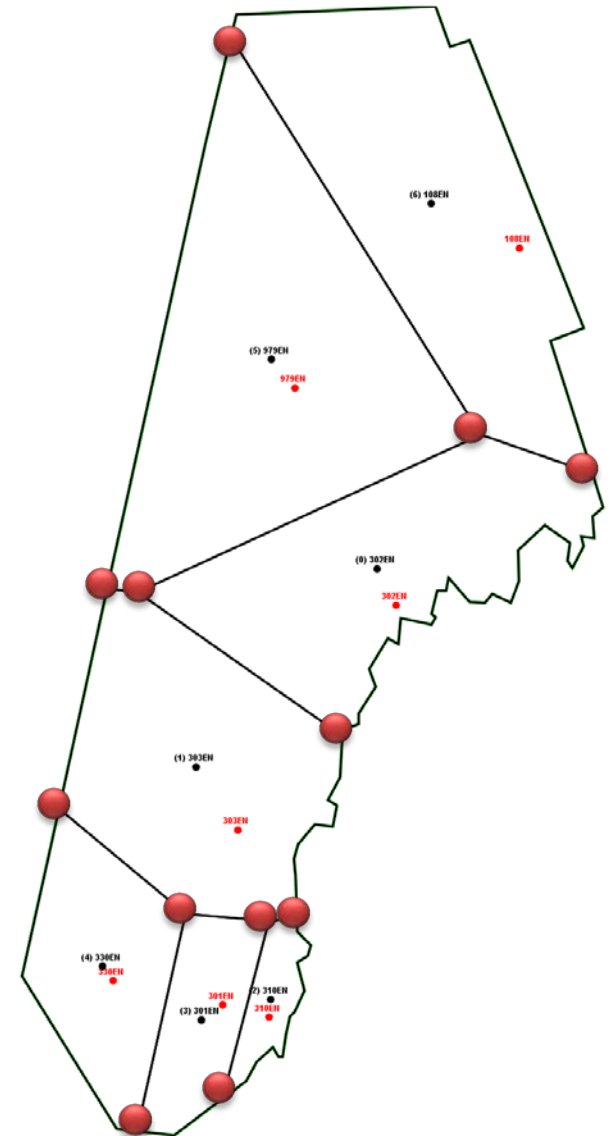
adaption to non-convex borders

- Transformation of the structure of the border-polygon into a DCEL.
- Copy both DCELs into a common DCEL (Overlay).
- Calculation of breakpoints between all half-edges of Voronoi and border DCEL and ordering these breakpoints with increasing y-coordinate.
- Move through the breakpoint list and reconstruct the pointer for the affected half-edges and vertices.
- Create new sectors based on the reconstruction and remove all half-edges and vertices outside the border polygon.
- Substitute the border-half-edges of the outer sectors by a set of two opposed auxiliary half-edges connecting the border breakpoints.



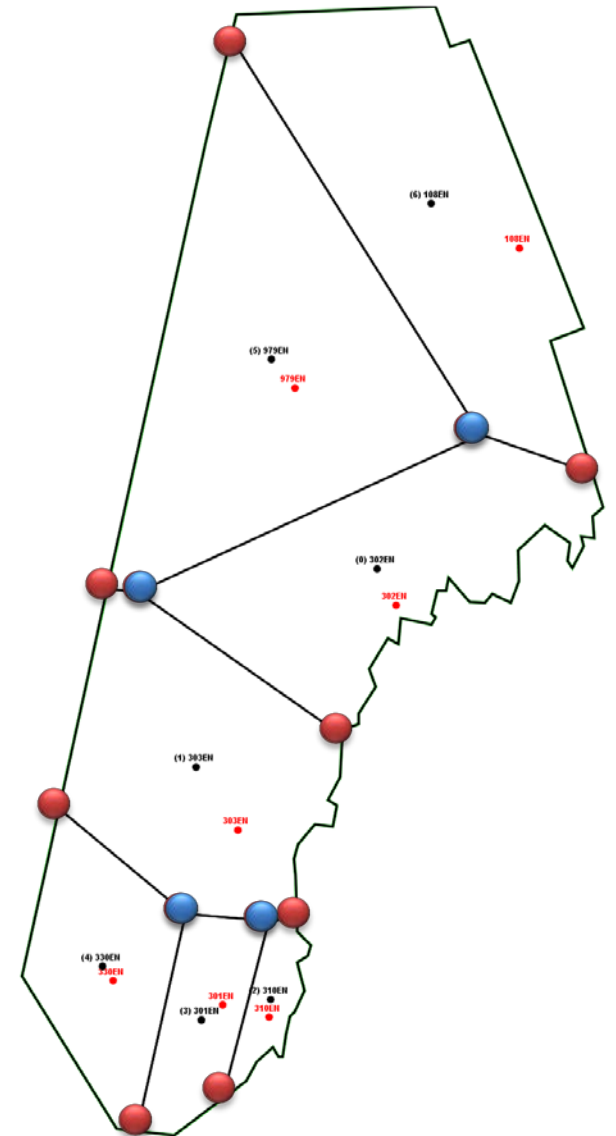
Structure of the Evolutionary Algorithm

- As chromosome a list of vertex points is used.



Structure of the Evolutionary Algorithm

- As chromosome a list of vertex points is used.
- Mutation of x- and y-coordinates of inner points by random.



Structure of the Evolutionary Algorithm

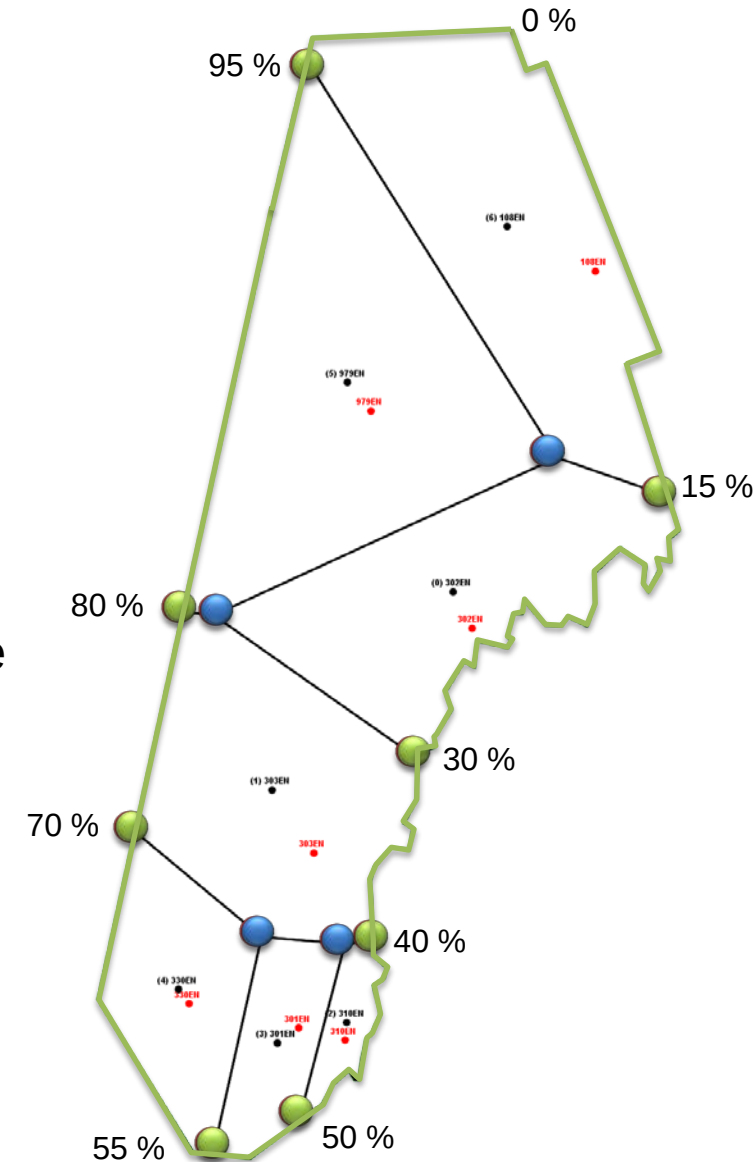
- As chromosome a list of vertex points is used.
- Mutation of x- and y-coordinates of inner points by random.
- More complex situation for border points because they have to stay on the border.

Structure of the Evolutionary Algorithm

- As chromosome a list of vertex points is used.
- Mutation of x- and y-coordinates of inner points by random.
- More complex situation for border points because they have to stay on the border.



- Considering the border polygon as a line.
- Associate each border breakpoint with the percentage value for the part of the distance from start to breakpoint point in relation to the whole distance.

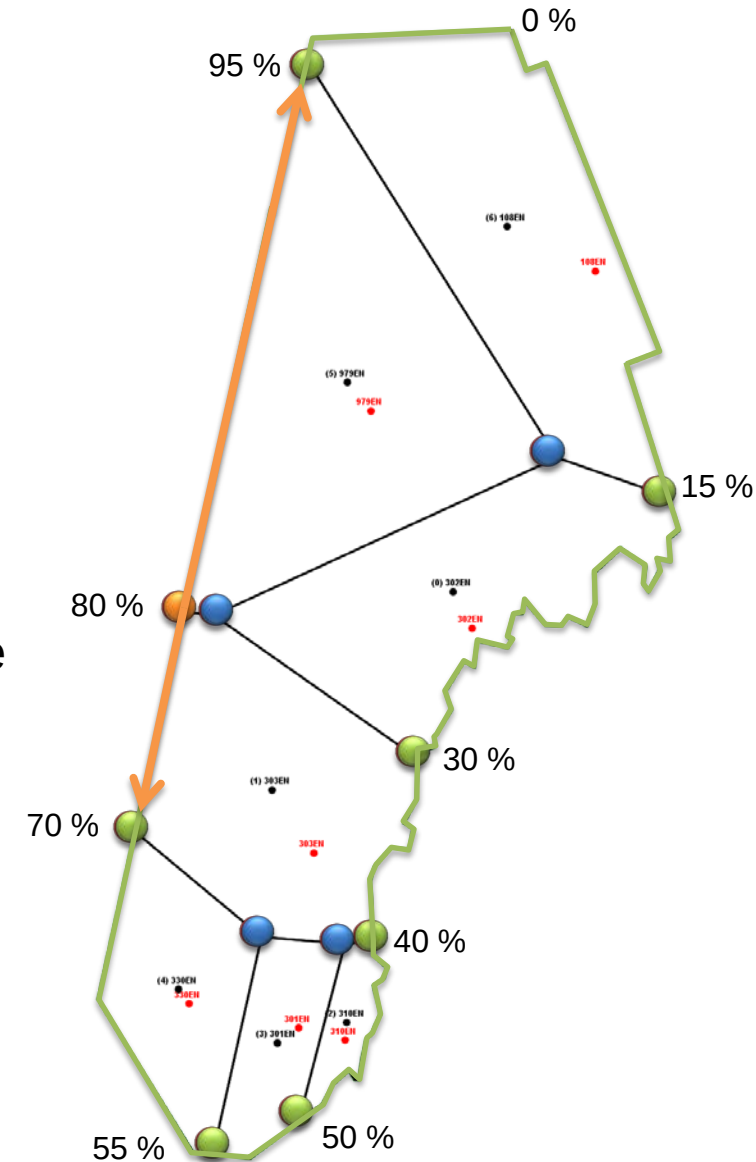


Structure of the Evolutionary Algorithm

- As chromosome a list of vertex points is used.
- Mutation of x- and y-coordinates of inner points by random.
- More complex situation for border points because they have to stay on the border.



- Considering the border polygon as a line.
- Associate each border breakpoint with the percentage value for the part of the distance from start to breakpoint point in relation to the whole distance.
- Mutate the percent values instead of the points.



Utility Function

calculation of task load

- based on data for necessary task times used by DFS and EUROCONTROL
- system of 55 tasks for radar, planning, arrival, airport, tower and apron controller with 129 sub-tasks

Example

Controller	Main Type	Sub Type	Task-Name	Time [s]	Per x Seconds	Group
Radar	Sector_Entry	CHANGE_SECTOR_IN_CRUISE_FROM_S AME_ACC	Initial Call	11	-	Radio Telephony
			Initial Monitoring	14	-	Monitoring
			Receipt Flight Strip	3	-	Coordination
		CHANGE_SECTOR_IN _CRUISE_FROM_DIFF_ACC	Initial Call	15	-	Radio Telephony
			Initial Monitoring	14	-	Monitoring
			Receipt Flight Strip	3	-	Coordination
	Conflict	CONFLICT_TYPE_1	Conflict Detection	17	-	Conflict Search
			Conflict Resolution	60	-	Conflict Resolution
	Recurring-Monitoring	RECURRING_MONITORING	Monitoring	5	120	Monitoring



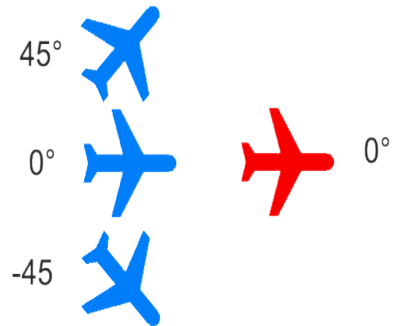
Utility Function

calculation of task load

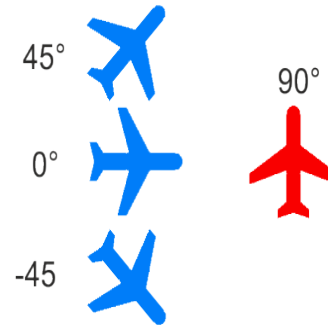
- based on data for necessary task times used by DFS and EUROCONTROL
- system of 55 tasks for radar, planning, arrival, airport, tower and apron controller with 129 sub-tasks

Conflict Types

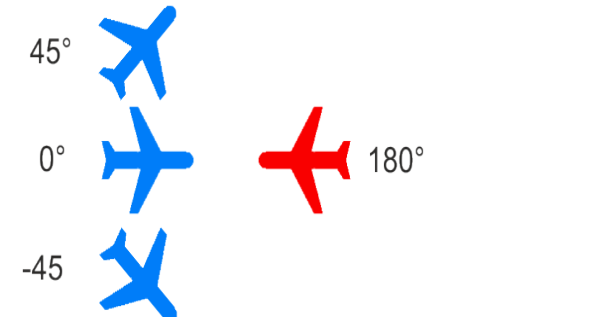
Consecutively:



Rectangular:



Opposite:



Evaluation Function

elements

Elements of evaluation function:

- sum of task load over all sectors [s]
- standard deviation (SD) of task load between sectors (task load SD) [s]
- standard deviation of interior angles in comparison to the average angle [°]
- number of flight intervals (partition of flight routes by sectors) over all sectors

Consequences

- standard deviation of interior angles was introduced to ensure sector structures without acute angles.
- “complexity constraint” is considered indirectly by the evaluation factors “interior angle SD” and “number of flight intervals”.



Evaluation Function

estimation of weights

- Tests with 688 flights over a day, 80 chromosomes and 10 simulation per variant
- 4 variants were tested in more detail:
 - version 1 tries to balance task load and task load SD
 - version 2 prefers task load
 - version 3 tries a weight for task load which changes with increasing number of generations
 - version 4 has a generation dependent weight for task load SD and furthermore a new process of selecting solutions for the next generation depending on their rank for each factor

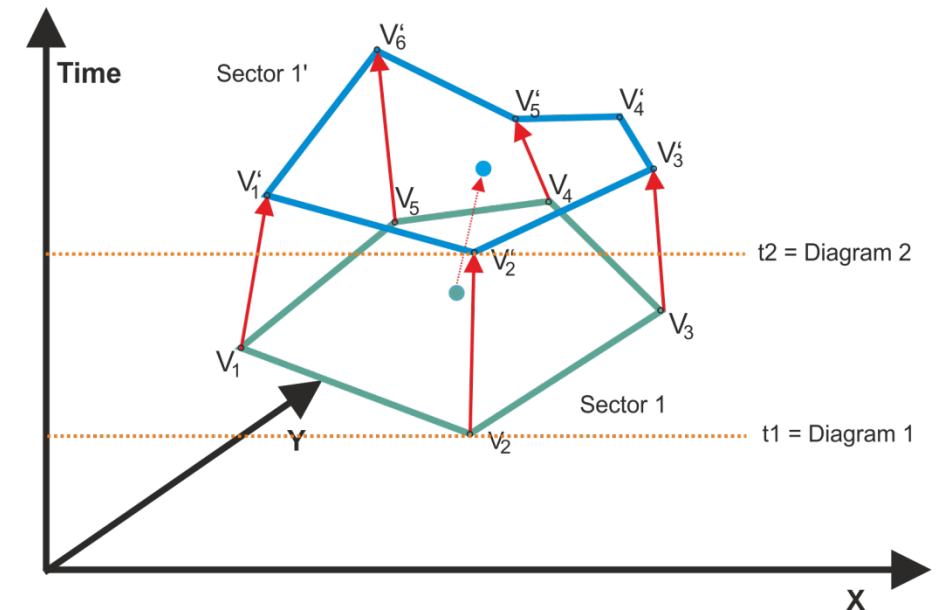
	Task Load [s]		Task Load SD [s]		Interior Angles SD [°]		# Flight Intervals		% Difference to Baseline
	M	SD	M	SD	M	SD	M	SD	
Baseline	166029		4172		30.5		2240		
Version 1	173482	5191	363	221	35.8	4.4	2381	98	321
Version 2	149548	1892	5109	689	16.3	0.4	1929	36	337
Version 3	159016	2934	605	237	31.8	1.8	2106	54	315
Version 4	157839	1185	819	150	28.5	0.9	2082	22	300



Integration of Dynamic Time-Component

general

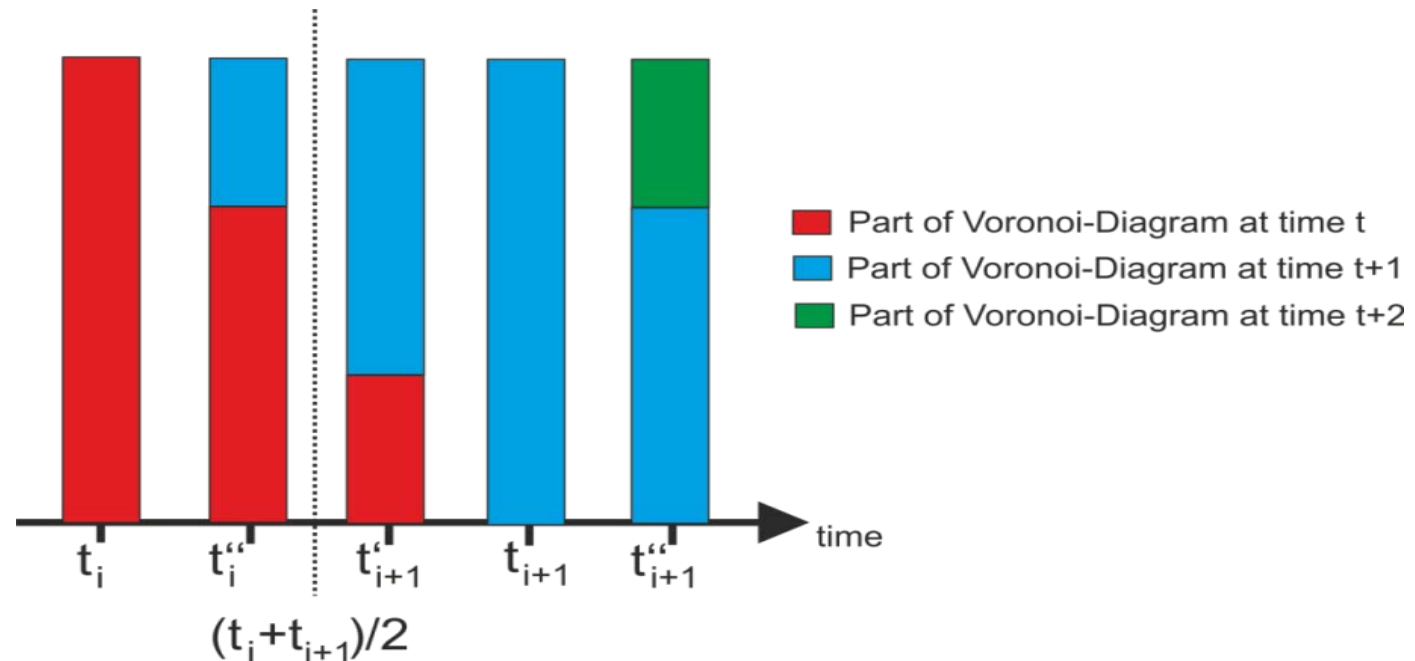
- flight data is portioned in dependence of the time for a better representation of different traffic demands over a day
- transition between successive sectorizations should be as „smooth“ as possible
- flights should be prevented from leaving one sector, entering the next and then jump back to the first because the new sectorization makes this necessary
- interim diagrams should be inserted between pairs of Voronoi diagrams and they should be created in such a way that they mirror the structure of the surrounding diagrams
- create combined interim diagrams for surrounding Voronoi diagrams the vertices of the Voronoi diagrams must be mapped



Integration of Dynamic Time-Component

creation of interim diagrams

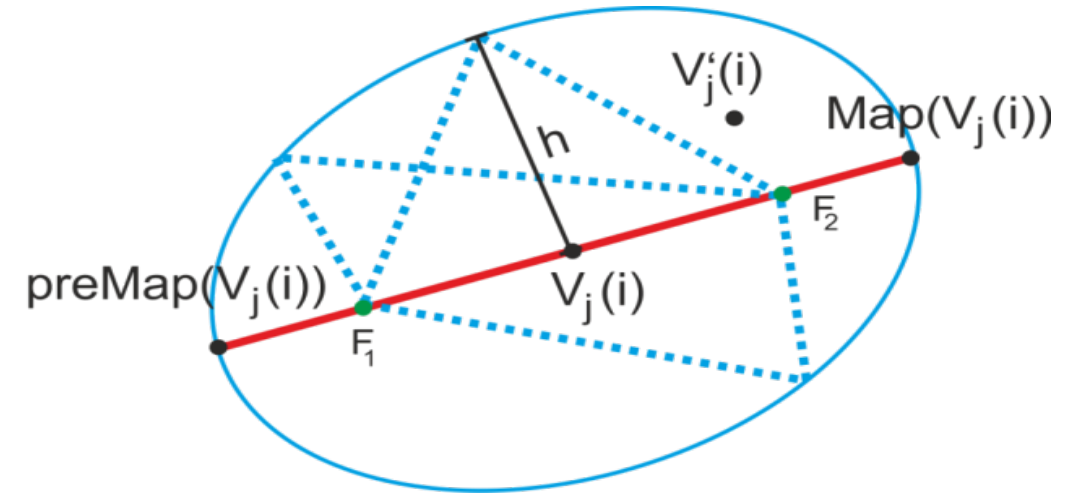
- the number of interim diagrams n inserted between a pair of Voronoi diagrams should be even.
 - first group of $n/2$ interim diagrams are based on the first Voronoi diagram
 - the second group on the second diagram
- each group of interim diagrams consists of the vertices number of the associated Voronoi diagram



Integration of Dynamic Time-Component

application of evolutionary algorithm

- introduction of the factor “vertexCloseness” into the evaluation function as a measure for the closeness to the vertices of the associated interim diagrams.
- usage of an ellipsoid for measuring the closeness in case of mapped vertices which do not have the same position.
- position change inside the ellipsoid is permitted, outside is penalized.
- three step approach for optimization:
 - Optimization of each Voronoi Diagram independently
 - Calculation of the vertex coordinates for the interim diagrams
 - Optimization of each interim diagram taking the “vertexCloseness” into account



Integration of Dynamic Time-Component

estimation of weight for „vertexCloseness“ in the evaluation function

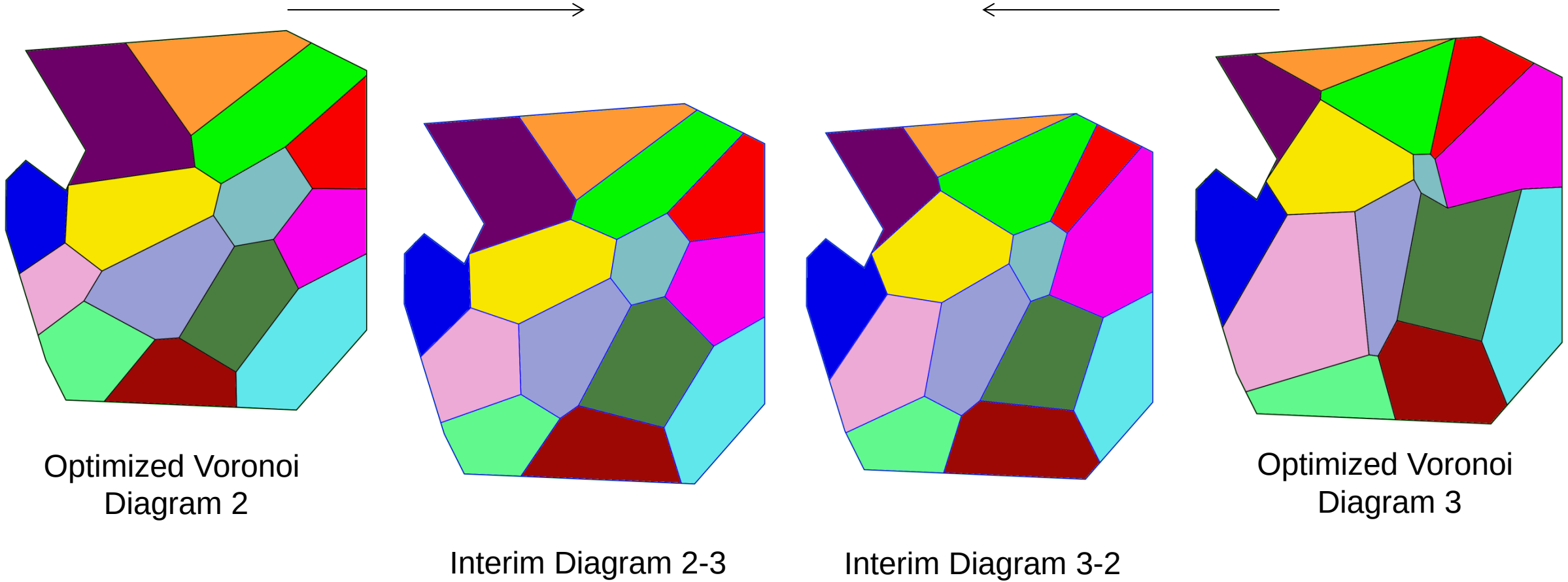
- tests were carried out with weights 0, 3, 5, 10
 - weight of 0 the vertexCloseness is not considered
 - weight of 10 the vertexCloseness is the most important factor in the evaluation function
- values show the difference between the original and the optimized interim diagram
- provide a possibility to optimize the interim diagrams as well as to stay close to the surrounding optimized Voronoi diagrams
- as a result: the weight of 3 was selected to be integrated into the evaluation function

Weight	Task Load [s]	Task Load SD [s]	Interior Angles SD [°]	# Flight Intervals	Avg. Position Difference [NM]
	Interim	1			
0	753.3	429.3	-1.4	18.2	21.3
3	736.6	143.8	1.0	19.5	5.2
5	519.5	116.7	0.9	12.4	3.5
10	328.2	91.1	0.1	8.3	2.4
	Interim	2			
0	592.4	247.1	1.0	16.2	14.3
3	604.0	84.2	0.3	16.7	4.8
5	497.3	68.3	-0.2	13.0	3.5
10	332.7	58.5	-0.3	8.7	2.4
	Interim	3			
0	1394.7	468.5	1.7	33.4	17.8
3	1095.2	106.3	0.2	27.8	5.9
5	563.8	97.6	0.0	14.1	5.3
10	545.3	62.5	-0.7	14.1	4.8
	Interim	4			
0	1597.9	486.0	5.6	38.1	16.1
3	1282.1	202.3	4.7	30.8	7.6
5	1110.9	203.9	4.3	27.8	6.7
10	797.5	136.7	3.2	20.3	6.0



Interim diagrams

example



Outlook

- presented
 - efficient combination of **Fuzzy Clustering**, **Voronoi Diagrams** and **Evolutionary Algorithms** for an automatic and dynamic sectorization regarding task load was presented
- next steps
 - import of DDR2 flight data of EUROCONTROL to enable realistic test scenarios with actual flight data
 - AutoSec will be integrated into current projects coping with sectorization and task load distribution
 - confirming the benefits of this approach in fast-time simulations
 - introduction to ATC controllers to verify a suitable degree of dynamic sector adjustment
 - usability study with humans-in-the-loop



Thank you.

Dynamic Airspace Sectorization using Controller Task Load

Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
German Aerospace Center

Institute of Flight Guidance

Dr.-Ing. **Michael Schultz**
Head of Department Air Transportation
Phone +49 (0) 531 295-2570
michael.schultz@dlr.de



Wissen für Morgen

