Dynamic Airspace Sectorization using Controller Task Load

Ingrid Gerdes, Annette Temme, Michael Schultz
Institute of Flight Guidance
DLR Braunschweig
Germany
ingrid.gerdes@dlr.de, annette.temme@dlr.de, michael.schultz@dlr.de

***Abstract*— Within this paper a new approach for a dynamic airspace sectorization based on controller task load is presented. We combine fuzzy clustering, Voronoi diagrams and evolutionary algorithms to create an adaptive and time dependent sectorization regarding to a harmonized controller task load. Our optimization strategy considers a predefined set of evaluation parameters and interim sectorizations are implemented for a smooth transition between the evolutionary adaptations of the sector structure. Furthermore, we developed a method to adapt Voronoi diagrams to a non-convex border. A short overview about the used methods is given and several tests of different evaluation functions are performed. The last part of this paper concentrates on the creation and evaluation of the interim sectorizations.**

Keywords-dynamic airspace secotors; controller task load; fuzzy clustering;Voronoi diagram; evolutionary algorithms

I. INTRODUCTION

Today the construction of airspace sectors is mainly done manually, which results in systematical deficiencies during the (dynamic) day of operations. Sector adaptation is often restricted to the division or merging of existing sectors considering the actual traffic flow. But no automatic adjustment of the basic layout takes place (e.g. sector borders). Nevertheless, an adaptive, more flexible sectorization would result in a more efficient use of the airspace [1], especially in case of significant weather phenomena, temporally restricted areas (e.g. volcanic ash) or operational deficiencies (e.g. controller strike).

Since the sectorless airspace management is a revolutionary concept mainly addresses the upper airspace [2]our research focus on an automatic, adaptive and time dependent sectorization with respect to a harmonized task load of the controller. Especially, our time dependent sectorization of the airspace is able to immanently consider the east-/westbound traffic patterns or the movement/development of weather phenomena which influence the task load distribution of airspace sectors significantly. Furthermore, new technologies like automated separation assurance and 4D trajectory operations influence the feasibilities for an automatic and dynamic sector design [3].

Several authors have developed ideas for an automatic creation of airspace sectors. One early approach has been developed by Delahaye et.al. [4] in 1998 with a focus on Evolutionary Algorithms. Other approaches used graph theory, Voronoi diagrams, integer programming, or clustering of flight tracks. For a detailed overview of recent work, we refer to [5] and [6]. Zelinski et al. [7] provide an interesting comparison of 8 different methods. Many of these ideas are of more theoretical nature and not all are directly applicable to an realistic nonconvex airspace problem (see figure 1.). With a clear focus on research projects coping with dynamic traffic flows influenced by weather phenomena or movement of volcanic ash clouds [8][9], we developed the AutoSec (*Automatic Sectorization*) idea. This approach allows the creation of an efficient airspace structure with respect to a number of evaluation factors like harmonized task load / variation or appropriate sector size (e.g. performance benchmarking [10]). Furthermore, the concept will enable sector adaptation in a time-dependent way by dynamically adapting the positon and shape of the sectors regarding to the actual necessities and restrictions (e.g. capacity, task load, controller availability, or stability / resilience). AutoSec is an essential and superior approach to combine unstructured / sectorless airspace and today's rigid structures to achieve both a balanced and more efficient use of the airspace.

The target of our contribution is prove the concept of the AutoSec approach, integrating ideas from "Fuzzy Clustering" as initial allocation of flights, "Voronoi Diagrams" to structure the airspace into sectors, and "Evolutionary Algorithms" to optimize the sectorization process. For a better understanding the three methods mentioned above are presented and explained in brief in the following section before presenting the general approach used by AutoSec.

II. FUNDAMENTAL THEORY

A. *Voronoi Diagrams*

A Voronoi diagram [11] is a fundamental approach to structure an area into several sections with a required center point (cf. figure 1.).

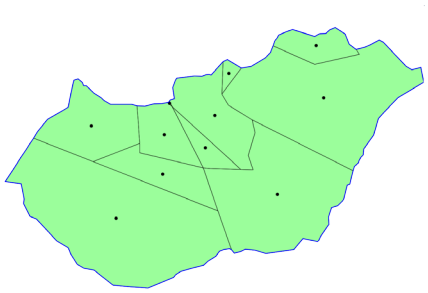

Figure 1. Hungarian airspace with actual sectors (Border in blue). Source: DDR2-Data of Eurocontrol.

Edges are created in the middle between two neighboring center points and vertices are those points which are associated with three different center points. An area containing all points with the same center point is called “face”. Examples for applications of Voronoi diagrams are the positioning of post offices in cities or the balanced allocation of people to electoral districts. For our approach the Fortune algorithm is used to construct a Voronoi diagram out of a set of predefined center points [11]. The decision for this type of algorithm was made because it is very fast, widely spread in the literature and easy to understand. Nevertheless, there are many other algorithms like Incremental Construction, Divide & Conquer or even constructing the Delaunay Triangulation as dual of a Voronoi diagram [13].

B. Evolutionary Algorithms

The typical principles of evolutionary algorithms follow biological evolutionary theory (cp. [14] or [15]). In nature, a group of individuals mix their genetic material, especially the information coded in chromosomes, to get better chances to survive in a hostile environment by a higher degree of adaptation. For an evolutionary algorithm, a population of solutions for an artificial problem is coded as sequence (chromosome) of parameters (genes) describing the problem solution.

These fundamental principles are transferred to the optimization of a technical/operational environment. Using a set of available solutions, specific components/parameters are mixed and possible mutated to generate a new set of solutions. The assessment of these new solutions regarding their fitness to the underlying utility function results in a hierarchy where the most appropriate solutions are used to be the parental generation for the next generation of solutions.

C. Fuzzy Clustering

The aim of clustering techniques in general is to find a partition of a given dataset. In a fuzzy partition, a datum is not necessarily assigned to a unique class or cluster. Instead, membership degrees are associated with each datum and each cluster. These membership degrees give information about the ambiguity of the classification. Fuzzy clustering techniques can adapt to noisy data and not well separated classes. The fuzzy clustering techniques used for our approach are based on optimizing an objective function. The clustering solution consists of cluster centers, i.e. the center of gravity of a cluster w.r.t. a distance function, and membership degrees. For an overview on fuzzy clustering and its applications, see e.g. [16][17].

D. Air Traffic Management

The sectorization of the airspace considers requirements of ATC (safety, capacity, and efficiency), users (unhindered access) and environment (restricted areas over cities, residential areas, etc.) [18]. Particularly, ATC demands for a sufficient airspace design to accommodate routes, holdings, or generate aircraft sequences. ATC sectorization is primary triggered by territorial aspects (air sovereignty) and secondarily triggered by operational demands like handling of mixed traffic, balanced work load, or procedure design.

In order to respond to the needs and future challenges of the air traffic (seamless European airspace), the European Commission initiates the Functional Airspace Block (FAB) design to implement multinational management to increase the airspace efficiency. Our dynamic airspace sectorization could be an enabling technology to consolidate airspaces by reducing coordination efforts, adaption of procedures, or balanced use of resources.

III. Restrictions and General Approach

The usage of Voronoi diagrams as well as of evolutionary algorithms alone is not perfect for the application to the sectorization problem because both methods have shortcomings when it comes to a realistic nonconvex airspace.

For the Voronoi diagram it is especially the lack of flexibility which is caused by the fixed position of the center points, which are the only parameters responsible for the resulting partition of the airspace. Therefore, the structure of the partition depends strongly on number and position of the center points used. Furthermore, Voronoi diagrams are usually restricted to convex border forms to ensure that each connection of two points of the plane is always inside the border. Because the form of an airspace sector often depends on the form of the underlying country borders or regions, a convex border polygon is generally not available (see figure 1.). On the other hand, an evolutionary algorithm needs some substantial information about a suitable sector structure concerning the number and principle form of sectors.

To overcome the shortcomings of the identified limitations described above, we developed a combined approach which consists of four steps:

1. Partitioning of flight data according to predefined time intervals for the calculation of cluster centers using fuzzy clustering methods [17].
2. Calculation of Voronoi diagrams based on the cluster centers for each time interval created in step 1 as initial solution for step 3 (section II A).

3. Adaption of the resulting diagram to a general border form.
4. Application of the Evolutionary Algorithm to the adapted Voronoi diagrams.

The current focus of AutoSec lays on a co-operation with the airspace module from the fast-time simulation environment AirTOp [12], to calculate the controller task load and to create time and flow dependent sector structures. AirTOp is able to simulate the flights for special scenarios, e.g. in case of avoiding volcanic ash and to give the changed flight trajectories as output to AutoSec, which than can be used for the creation of an appropriate sector structure.

Thus AutoSec currently is and will be a major element of the DLR research projects aiming at airspace efficiency considering ecological, economical and operational aspects. This paper will focus on step 3 and 4, which will be explained within the next section.

IV. IMPLEMENTATION

A. *Data Structure*

To store data of a Voronoi diagram an appropriate data structure called *Doubly Connected Edge List* (DCEL, [11]) is used. A DCEL consists of three lists (vertices, half-edges, faces) where the elements of every list are connected in several ways. For the edge list, each (undirected) edge is divided into a set of two half-edges (directed) with opposite directions called twins (see Figure 2.). Furthermore, the information about the previous and the next half edge when moving counter-clockwise though the half-edges of the corresponding face, the origin vertex, and a pointer to the corresponding face are stored with every half-edge additionally. Faces are finally defined by the center point and a pointer to a single half-edge belonging to this face. So, every half-edge is linked to the corresponding face and each face is defined by only one corresponding half-edge.

B. *Introduction of unrestricted border forms (Line-Segment-Intersection)*

Another problem to be solved is the integration of a nonconvex border line for an airspace region. Because the selected Voronoi algorithm is restricted to a convex border polygon, the closest rectangle including it is used as border.

The next steps for the creation of a sectorization considering an authentic order polygon are:

1. Transform the border polygon into another DCEL.
2. Calculate coordinates for all breakpoints between the half edges of both DCELs and sort the breakpoint list with increasing y-coordinate values.
3. Copy both DCEL's into a common DCEL (overlay).
4. Move through the breakpoint list and reconstruct the affected half edges and vertices (create new half edges and vertices, assign new previous, next and twin) and create a new face list for the overlay DCEL.
5. Remove all vertices, half edges, and faces which are outside the border polygon.
6. Substitute the border half edges of the outer sectors by a set of two half edges connecting the breakpoints (see mutation operator of Evolutionary Algorithm).

This approach is based on a "line-segment-intersection" method [11]. It is possible that a sector is divided and that a center point is not included in a new sector. Therefore, the center of gravity is calculated as new center point for each sector.

Figure 3. shows border polygon and Voronoi diagram after the last step. The border is for illustration only and shows, which parts have to be inserted in the sectors of the Evolutionary Algorithm before the task load of the complete sectors can be evaluated.

C. *Task Load Calculation*

Since the connection of AutoSec and AirTOp is one important target for the development of AutoSec, the controller task load is calculated similar to the methods we already implemented in the AirTOp environemnt. For AirTOp a system of tasks / subtasks were defined and time values for the duration of each task and the frequency (if necessary) were advised.

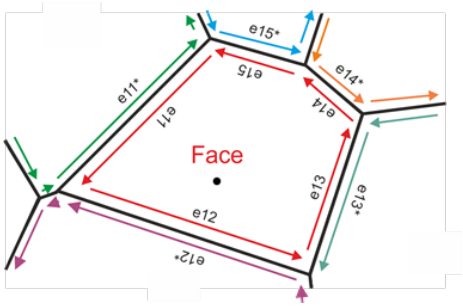

Figure 2. Example for the segmentation of a face into a system of half-edges and twins (marked with*). E11* is twin, e15 is the previous and e12 the next half-edge of e11.

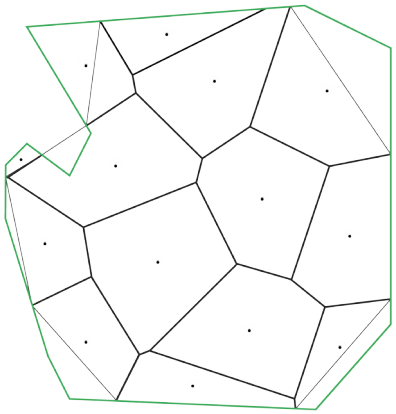

Figure 3. Voronoi (black) and Border-DCEL (green) after step six .

For the type of controller tasks and the estimation of time values a comprehensive study was carried out at the DLR [20].

The definitions, subdivision of controller tasks and times used by several companies like the German ANSP provider DFS or EUROCONTROL were taken into account. As a result a system of 55 tasks for radar, planning, arrival, airport, tower, and apron controllers were defined with a total of 129 subtasks. TABLE I. shows an example with some tasks for a radar controller. As shown in this table the handling of possible conflicts is an important and time consuming task (see column "Time" in TABLE I.). Therefore, the identification of conflicts is very important for each controller task load calculation and a conflict detection algorithm has to be applied (implemented similar to [21]). Afterwards, all task load values for all aircraft inside the sector borders are summarized to a key value.

For the calculation of the sector task load, the flight time of an aircraft in each sector has to be calculated (recurring monitoring every 120 seconds) as well as entry and exit times, speeds and the position of the entry and exit points for appointing previous and next Area Control Centers (ACC).

D. Structure of the Evolutionary Algorithm of AutoSec

For AutoSec the vertices of the Voronoi-diagram created before are used as genes and a sequence of these vertices with a pre-defined order forms a chromosome. For crossover two chromosomes are selected and there is a small chance for each gene of the chromosomes to be exchanged between them (Uniform Crossover, [14]). For the mutation operator there is again a small probability for each gene to undergo mutation. For most vertices this means a random variation of the x- and y-coordinate, but for vertices situated on the border polygon of the observed airspace this is much more complicated. Each point of the border polygon has to stay a border point after mutation, so this restricts the mutation possibilities severely. Furthermore, it is not possible to include the points of the border polygon as additional Voronoi vertices because this would lead to a situation where some vertices cannot be mutated at all. This problem was solved by introducing two adaptations to the algorithm structure and the mutation operator.

The first one is the replacement of the part of the border polygon of the associated face between the two border breakpoints by an auxiliary edge, connecting these breakpoints directly (see Figure 4. Figure 3.). These breakpoints are added as Voronoi vertices. Now, the number of vertices for each face stays always the same.

The next necessary adaptation considers the border as a line and allocates a percent value from 0 to 100 to the position of each border vertex depending on the distance to the first vertex and the length of this line. With this adaptation the mutation of a border vertex can be handled as the random selection of a percentage between the percentage values of the preceding and the successive border vertices. If necessary, the real course of the border can be easily inserted using the percentage values of the border Vertices. Nevertheless, it is necessary to prevent the creation of inner points and self-intersecting lines by special procedures. This holds for normal vertices when applying mutation and crossover operators as well.

The most important function for each Evolutionary Algorithm is the evaluation function. With the focus on uniformly distributed task load the following factors are taken into account:

- 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

Especially the factor "standard deviation of interior angles" (interior angles SD) was introduced to ensure sector structures without acute angles. For the selection of chromosomes for the next generation a fixed number of best and different chromosomes are selected instantly before choosing the remaining chromosomes randomly.

Several authors have stated that a convexity constraint should be applied to the created sectors to ensure that an aircraft visits every sector only once (cf. [4][19]). They also demanded a constraint limiting the inter sector flow.

TABLE I. EXAMPLE FOR SOME CLASSIFICATION TYPES AND TASK LOAD TIMES FOR CONTROLLER ACTIONS WITHIN A RADAR SECTOR (MO: MONITORING, RT: RADIO TELEPHONY, CO: COORDINATION, CL: CLEARANCES, CS: CONFLICT SEARCH, CR: CONFLICT RESOLUTION)

Controller	Main Type	Sub Type	Task-Name	Time [s]	Repeat every x Seconds	Group
Radar	Sector_Entry	CHANGE_SECTOR_IN_CRUISE_FROM_SAME_ACC	Initial Call	11	-	RT
			Initial Monitoring	14	-	MO
			Receipt Flight Strip	3	-	CO
		CHANGE_SECTOR_IN_CRUISE_FROM_DIFF_ACC	Initial Call	15	-	RT
			Initial Monitoring	14	-	MO
			Receipt Flight Strip	3	-	CO
	Conflict	CONFLICT_TYPE_1	Conflict Detection	17	-	CS
			Conflict Resolution	60	-	CR
	Recurring-Monitoring	RECURRING_MONITORING	Monitoring	5	120	MO

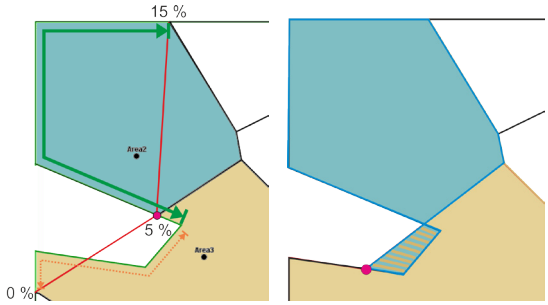

Figure 4. Left: Possible mutation area for the red border breakpoint between 0 and 15 percent of the border polygon. Auxiliary edges are shown in red, allowed area as green vector, the forbidden part of the border as dotted orange vector. Right: Result of an incorrect border mutation.

These constraints are included indirectly by the evaluation factors "interior angle SD" and "number of flight intervals" but not as mandatory. As far as the convexity constraint is concerned todays sectors are not necessarily convex (see Figure 1.).

V. Parameter and Function Optimization

For an appropriate selection of crossover and mutation probabilities and the structure of the evaluation function, several tests with different parameter sets have been carried out. These tests were divided into three groups, the first for the parameters, the second for the determination of the elements of the evaluation function and the third for the calculation of weights representing the trade-off between the different elements of the evaluation function. For the first two tests the same experimental setup was used. It consisted of a set of an artificial airspace section with the size of 200 x 200 Nautical Miles, 13 randomly selected center points distributed in this area and 50 randomly created flights. For the evolutionary algorithm a population size of 40 chromosomes of length 13 (number of Voronoi vertices) per generation and a number of 6 best / different chromosomes for immediate selection was used. Furthermore, the number of generations to be simulated was set to 1000.

A. *Optimization of Parameters*

The parameter tests with 8 different sets for mutation (ranging from 0.05 to 0.2) and crossover probabilities (ranging from 0.05 to 0.25) led to a mutation value of 0.05 and a much higher crossover probability of 0.2. The second value is much higher because the structure of the chromosomes in the population tends to share several gene values, especially for good solutions. Therefore, the crossover operator exchanges very often the same values between the selected chromosomes. But this does not hold for the mutation operator which modifies every gene it is applied to.

B. *Evaluation function*

To select the best combination of evaluation parameters for the evaluation function, two combinations where compiled and compared. The first approach is called "basic evaluation function". It includes only the task load SD and the interior angle SD and therefore reflects the aim of distributing the task load uniformly over all sectors. The second version additionally includes two other factors. The first is the difference between the sum of task loads over all sectors for the actual run and the lowest task load value over all sectors found so far and the second the same for the number of flight intervals ("extended evaluation function"). The subtraction of the lowest values found is necessary to avoid very high values for task load sum and flight interval number with increasing flight numbers, which then would suppress the influence of interior angle SD and task load SD. The results for this comparison are shown in TABLE II. together with the evaluation value for the Voronoi diagram which is used as basic chromosome for the creation of the start population for every evolutionary algorithm.

Surprisingly, the task load value for the basic version exceeds even the value of the non-optimized Voronoi diagram dramatically, but the task load SD is very low. A closer look at the resulting sector structure has revealed that because the task load itself was not part of the evaluation function it was increased for those sectors with low task load values for the sake of a reduced task load SD. This was done by creating acute angles which were able to catch flights more often and therefore increased the portion of task load for entering and leaving sectors. This can be seen in column "Interior Angles SD" and "# Flight Intervals".

Because it cannot be the target of an optimization tool for controller task load to increase the task load itself neglecting the form of the sectors and the average task load the basic version of the evaluation function will not be used further. So, we can conclude that an evaluation function for an automatic sectorization should not only include parameters for the observed problem (e.g. distribution of task load) but as well some factors which stimulate the creation of sector forms favored by controllers.

The comparison between different evaluation functions has shown that there is especially a strong dependency between task load value and task load standard deviation. Tests with different virtual flight schedules with varying flight numbers revealed the necessity to introduce weights to balance the influence of both factors and to redesign the calculation of the evaluation value for each element of the evaluation function. Several different weights were developed by judgement and evaluated with a high number of simulation trials. Four of these variants representing typical cases are explained in more detail in this section.

TABLE II. Comparison of the Main Parameter for Limited and Complete Evaluation Function

Type	[s] Task Load	[s] Task Load SD	[°] Interior Angles SD	# Flight Intervals
Voronoi	16476	457	27.5	194
Basic Evaluation	18330	8	46.5	229
Ext. Evaluation	14939	352	25.9	165

The first step was the usage of the fraction of actual evaluation value and the best found value instead of subtracting the last from the first. So, the new evaluation value simply reflects the quality of the actual solution in comparison to the best found and is independent from the number of flights. Altogether more than 10 new versions of the evaluation function with different weights were developed and tested, where 4 were selected for more sophisticated simulation trials with a flight plan with 688 flights over a complete day and 80 chromosomes with 12 selected to remain unchanged for the Evolutionary Algorithm. Each simulation with these parameters needed a time of approximately 8:30 minutes using a standard pc.

TABLE III. shows the weights for the remaining four versions. The first two have fixed factors. The third features a task load weight which varies with the number of generations and includes the actual generation number of the evolutionary algorithm (genNr), the planned maximum number of generations (maxGen) and the fraction of task load and task load SD (RelationF) for the first generation. The third version allows an easier creation of solutions with good results for the task load SD at the beginning and an increasing pressure for good solutions for the task load for later generations. This ensures a more controlled creation of solutions which always take both factors into account.

For the last version a weight for the task load SD was introduced incorporating the other weights and again actual and maximum number of generations. Furthermore, not only the weights were changed but the type of selecting chromosomes for the next generation as well. Instead of using the evaluation values directly a ranking of evaluation values was used as basis for the calculation of the selection probability. The results for the test simulations are presented in TABLE IV. The columns show the results for the different parts of the evaluation function. Furthermore, the values of the evaluation function for the non-optimized Voronoi diagram are given in the first row as baseline result.

The results for Version one show an overweight for the task load SD, but reducing the weight for this element as done for version two turns the results to the other extreme. So, fixed values always tend to prefer either task load or task load SD. Therefore, version 3 including the relation between both factors was developed. The results for the mean values (M) in TABLE IV. are better than the baseline values for every element of the evaluation function but the SD of the interior angles. The results for task load and task load SD can be seen as a compromise between both elements. Unfortunately, the values for the standard deviations are quite high, indicating unreliability in the quality of the solution. To cope with this, version 4 was developed were the selection process is decoupled from the actual evaluation value. Instead of using a selection in dependence of the evaluation value a ranking was introduced for each factor. The sum of the ranking positions for all factors is than used for the calculation of the selection probability.

Again the results for task load and task load SD are a compromise between the possible values for both factors, but the interior angles and the flight intervals are as well lower than the baseline results. Furthermore, the standard deviation of all factors is considerably lower than for nearly all other versions. For the decision which version should be used for future simulations the fraction of every mean value to the corresponding baseline values were calculated and summarized for each version. The results are shown in the last column and lead to version 4 as the best evaluation function.

VI. INTEGRATION OF DYNAMIC TIME-COMPONENT

The target of AutoSec is the dynamic and automatic creation of airspace sectorizations with a focus on an appropriate partitioning of traffic data into time intervals. Furthermore, a smooth transition from one sectorization to the next should be created to prevent flights from leaving one sector, entering the next and then jump back to the first because the new sectorization makes this necessary. For this transition the so-called interim-diagrams are introduced which are inserted between the Voronoi diagrams (sectorizations).

So the general course of action for the creation of interim diagrams is as follows:

1. Calculation of Voronoi diagrams for each set of center points.
2. Mapping of Vertices between successive Voronoi Diagrams.
3. Splitting of the time intervals into smaller even-numbered sub-intervals.
4. Application of the Evolutionary Algorithm with the normal evaluation function to each Voronoi diagram.
5. Calculation of the corresponding number of interim diagrams taking the mapped vertices of the optimized Voronoi diagrams into account.

TABLE III. VARIANTS OF EIGHT FACTOR CALCULATION FOR THE EVALUATION FUNCTION

Version	Task Load (wtl)	Task Load SD	Interior Angles SD (wa)	# Flight intervals (wfi)
Version 1	1	1	1	1
Version 2	1	0.1	1	1
Version 3: Relation	(1 + genNr*2 / maxGen) * RelationF / 6	1	0.5	0.5
Version 4: Ranking	1	(1+wtl+wa+wfi) - genNr * 1.5 / maxGen	0.5	0.5

TABLE IV. RESULTS FOR THE DIFFERENT TRADE-OFF TEST VERSIONS (M: MEAN, SD: STANDARD DEVIATION).

	Task Load [s]		Task Load SD [s]		Iterior 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

6. Application of the Evolutionary Algorithm with an extended Evaluation function to each interim diagram.

The most important aspects will be described in more detail in the next sections.

A. Mapping of Vertices / Faces

The structure of Voronoi diagrams depends on the number and position of the center points. When using time-dependent sets of center points, the resulting Voronoi diagrams can be very different in the number of faces as well as in position and number of vertices. The target is to insert interim diagrams between successive pairs of Voronoi diagrams which should reflect the structure of the involved Voronoi diagrams. Therefore, it is necessary to identify faces and vertices, which exist in successive Voronoi diagrams V_i and V_{i+1}, which can be connected to each other. They can be used afterwards to calculate vertices for the interim diagrams between the connected vertices of V_i and V_{i+1} The connected vertex in a successive Voronoi diagram V_{i+1} is called Map, in the preceding diagram V_i preMap. If successive Voronoi diagrams have different vertex numbers, as many connected vertices as possible are identified. The remaining vertices stay unchanged for the interim diagrams.

In a first step connected faces are identified. For doing so the following steps are carried out:

1. All faces next to the border of the Voronoi diagrams are identified (so called border faces).
2. The distances between all center points of Voronoi diagram V_i to the center points of Voronoi diagram V_{i+1} are calculated.
3. Every face of the first diagram is connected to the face with the lowest distance between the center points of the second diagram (duplicate mappings possible).
4. Run through the face list for duplicate connections and connect those of the same type (border / inner face) preferential. Connect the other face to the next best until no more changes occur.

With the faces identified it is much easier to identify the vertices as well. So, the algorithm for connecting vertices will work as follows:

1. Map those vertices which are nearly similar.
2. Identify border vertices by comparing the related faces.
3. Calculation of the number of unmapped vertices per face.
4. Continue until no mapped face with only one unmapped vertex exists anymore.
    a. Direct mapping of the remaining nodes for all faces where only one vertex is unmapped and both connected faces feature the same number of vertices.
    b. Recalculation of the number of unmapped vertices for all faces.
5. Run through all faces in the sequence according to the number of unmapped vertices (lowest first):
    a. Mapping of the remaining vertices in dependence to the distance to the unmapped vertices in the mapped face.
    b. If the number of vertices for the mapped faces is different, remove duplicate mappings.
    c. Recalculation of the number of unmapped vertices for all faces.

For step 2 it is known that a border vertex can have only one half-edge which does not belong to the border. With this half-edge it is possible to identify the adjacent faces and therefore the mapped faces and the connected vertex as well (see Figure 1.).

B. Creation of Interim Diagrams

As mentioned before the biggest problem for the creation of interim diagrams is the perhaps completely different structure of successive surrounding optimized Voronoi diagrams.

To cope with this problem the number (nr) of interim diagrams inserted between two Voronoi diagrams should be even, so it can be divided into two equal numbered groups. The first group gets structure and data from the first optimized Voronoi diagram, the second group from the second diagram.

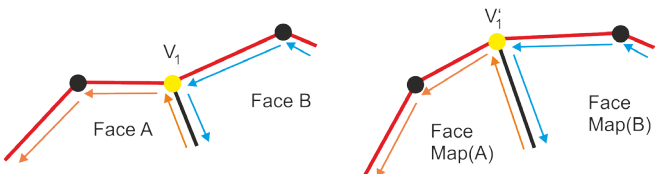

Figure 5. Identification of mapped border vertices V1 and V'1 using faces.

In the next step, the coordinates of all mapped vertices are recalculated proportional to the distance between the vertex of the first Voronoi diagram and its mapping.

Let V_j [i] be the vertex j of diagram V[i], i∈{0,..,nr+1}, V[0] the first and V[nr+1] the second of the successive surrounding optimized Voronoi diagram, $V_j^{map}[0]=V_k$ [nr+1] the mapped vertex k of the second Voronoi diagram for vertex j, then holds the following formula for the coordinates for i∈{0,..,nr/2-1} :

$$V_j[i](x) = V_j[0](x) + \left(\frac{i}{nr+1}\right) * \left(V_k[nr+1](x) - V_j[0](x)\right)$$
$$V_j[i](y) = V_j[0](y) + \left(\frac{i}{nr+1}\right) * \left(V_k[nr+1](y) - V_j[0](y)\right)$$

An equivalent formula based on the data of the second Voronoi diagram is used for the calculation of the coordinates of the second group of interim diagrams V_i, i∈{nr/2,…,nr}. Figure 1. shows a graphical interpretation of this equation.

C. *Application of Evolutionary Algorithm*

For the optimization process with the evolutionary algorithm the Voronoi diagrams are optimized first. Afterwards the coordinates of the vertices of the interim diagrams are calculated in dependence of the new coordinates of the mapped vertices of the optimized Voronoi diagrams. Because the interim diagrams should guarantee a smooth transition between the optimized Voronoi diagrams it is not possible to optimize them independently. At least the vertices have to stay close to their original values. Therefore, a special limitation formula for the mutation of vertex coordinates was added to the former mentioned evaluation function. In case of equal Map and preMap the difference to the original Vertex is calculated and set in relation to a predefined threshold value, where values above the threshold are penalized by doubling the distance.

In case of different map and preMap an ellipsoid between Map and preMap is used to define an allowed area (see Figure 7.). Every ellipsoid is described by a defining distance and two focus points F_1 and F_2. The border of the ellipsoid consists of all points where the sum of the distances from this point to the focus points F1 and F2 is equal to the defining distance.

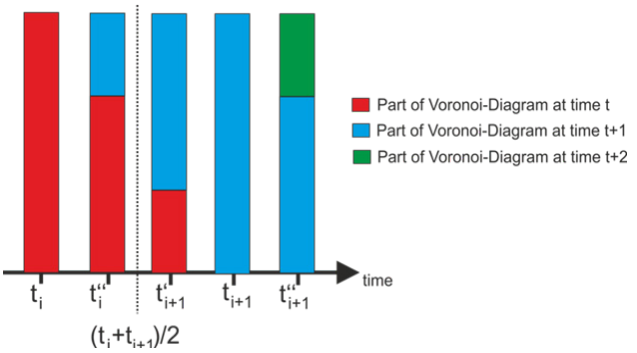

Figure 6. Example for the transition from one Voronoi diagram (red) to the next (blue) with nr = 2 interim diagrams in dependence of the time interval t_i.

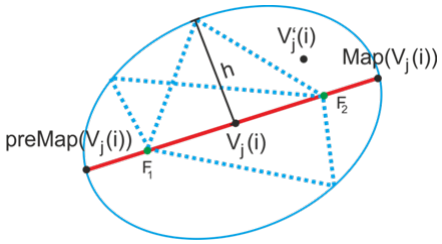

Figure 7. Definition of an allowed area for vertex Vj(i) with different Map and preMap.

Now a "vertexCloseness" for each vertex can be defined as the fraction of the sum of the distances form the new point to the focus points and the defining distance of the ellipsoid (sum of distances from preMap to the focus points as well). Again a new position outside the ellipsoid gets twice the penalty points as an inside point. Then the sum of the vertexCloseness for each vertex is used as corresponding evaluation value of a chromosome.

In addition a weight factor (wvc) for the evaluation function has to be determined. Because the interim diagrams depend strongly on the optimized Voronoi diagrams, it is not possible to compare the result of different simulation runs directly. For an estimation of the influence of the weight factor and the vertexCloseness on the optimization, the evaluation values for each part of the evaluation function of each interim diagram before the start of the optimization process were used as reference values.

D. *Results*

With this definition it is possible to calculate the improvement created by the optimization process. Together with the values for the average difference to the original position of each vertex this allows the selection of the best weight for the evaluation function. As Test-Scenario the same flight schedule and optimization parameters were used as for the weight test described in Section I.

To facilitate the possibility of interim diagrams two additional sets of center points were created and two interim diagrams between each pair of Voronoi diagrams were determined. The results are presented in TABLE V. The higher the values the better is the quality of this factor of the evaluation function. As test cases weights of zero, three, five, and ten were selected by judgement. Zero is used to demonstrate the general possibilities of an optimization that is not restricted to stay close to the original values. Nevertheless, the results for all weights show significant improvements, even for a factor of ten which rates the vertex closeness higher than all other factors of the evaluation function.

The noticeable high values for the average position difference are a result of the definition of the vertexCloseness factor. For this factor not the real distances but the fraction to the defining distance of the ellipsoid is used as measure for the quality of this factor.

TABLE V. RESULTS OF THE TEST RUNS FOR CALCULATING THE INFLUENCE OF THE OPTIMIZATION ON THE INTERIM DIAGRAMS. ALL RESULTS SHOW THE DIFFERENCE BETWEEN THE EVALUATION OF THE ORIGINAL AND THE OPTIMIZED INTERIM DIAGRAM.

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

In case of a higher distance between Map and preMap this defining distance can be high which allows a higher displacement of vertices before getting penalized.

Because the selected weight should provide a possibility to optimize the interim diagrams as well as to stay close to the surrounding optimized Voronoi diagrams, a weight of three was selected for future simulations.

Figure 8. shows exemplarily the sequence of interim diagrams between the optimized Voronoi diagrams two and three from one simulation run. For a closer examination a special sector marked in blue was used to demonstrate the movement of vertices in case of very different vertex position in the surrounding optimized Voronoi diagrams.

VII. OUTLOOK

With our approach we can demonstrate the efficient combination of fuzzy clustering, Voronoi diagrams, and evolutionary algorithms to enable an appropriate sectorization of the airspace regarding to the controller task load. This approach is currently used to evaluate airspace navigation service provider under both operational and economical aspects.

At the implementation level we currently focus an automatic import tool for DDR2 flight data of EUROCONTROL and integration into the existing tool chain to enable realistic test scenarios with actual flight data.

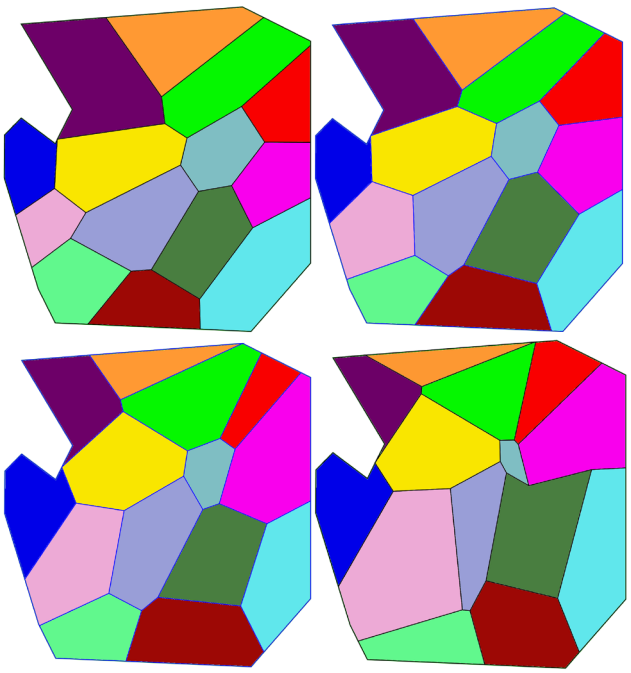

Figure 8. Example of a sequence of optimized Voronoi diagram two (upper left), interim diagrams three (upper right) and four (lower left) and optimized Voronoi diagram three (lower right) for a weight of three.

Further on, scenarios of DLR projects coping with sectorization and task load distribution are evaluated with respect to the benefit of dynamic airspaces. When the anticipated benefits of this approach have been confirmed in fast-time simulations, the next steps are workshops with air traffic controllers to verify a suitable degree of automatic sector adjustment and following thereon a usability study with humans-in-the-loop.

ACKNOWLEDGMENT

We would like to thank Mrs. Tanja Luchkova especially for her support concerning AirTOp and the DDR2 Database of EUROCONTROL.

REFERENCES

[1] SESAR Joint Untertaking, Automated support for dynamic sectorisation, [online] available: www.sesarju.eu/sesar-solutions/network-collaborative-management-and-dynamiccapacity-balancing/automated-support [Accessed August 2016]

[2] B. Birkmeier, “Feasibility analysis of sectorless and partially automated air traffic management”, PhD Thesis, TU Braunschweig, 2015

[3] P. Kopardekar, K. Bilimoria, B. Sridhar, „Initial concepts for dynamic airspace configuration“, in 7th AIAA Aviation Technology, Integration and Operations Conference (ATIO), 2007, Belfast, Ireland

[4] D. Delahaye, M. Schoenauer, J.M. Alliot, “Airspace sectoring by evolutionary algorithms”, in IEEE International Congress on Evolutionary Computation, 1998 .

[5] H.D.Sherali, J.M. Hill, “Configuration of airspace sectors for balancing air traffic controller workload”, in Annals. of Operations Research, vol 203, Springer , 2011, pp. 3-31.

[6] J. Li, T. Wang, M. Savai, I. Hwang, “Graph-based algorithm for dynamic airspace configurationd”, in Journal of Guidance, Control and Dynamics, vol 33, 2010, pp. 3-31.

[7] S. Zelinski, C.F. Lai, “Comparing methods for dynamic airspace configuration”, in 30th Digital Avionics Systems Conference, 2011.

[8] T. Luchkova et al., “Analysis of Impacts an Eruption of Volcano Stromboli could have on European Air Traffic”, ATM seminar, 2015

[9] M. Kreuz, T. Luchkova, M. Schultz, “Effect Of Restricted Airspace On The ATM System”. WCTR Conference 2016, Shanghai, China

[10] T. Standfuß, M. Schultz, “Benchmarking Of Area Control Center In FABEC Context”. WCTRS, Shanghai 2016

[11] M. de Berg, O. Cheong, M. van Keveld, M. Pvermars, “Computational Geometry, Algorithms and Applications”, Berlin, Springer, 2008.

[12] AirTOpsoft, “www.airtopsoft.com/”, [Online], available: http://www.airtopsoft.com/products.html. [Accessed February 2016]

[13] F. Aurenhammer, R. Klein, “Voronoi Diagrams”, [Online], available http://www.pi6.fernuni-hagen.de/downloads/publ/tr198.pdf [Accessed October 2016]

[14] Z. Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs”, Berlin Heidelberg, Springer, 1996.

[15] I: Gerdes, F. Klawonn, R. Kruse, „Evolutionaere Algorithmen“, Wiesbaden, 2004

[16] J. V. de Oliveira, W. Pedrycz, “Advances in Fuzzy Clustering and its Applications”. John Wiley & Sons, 2007.

[17] A. Keller, “Objective Function based Fuzzy Clustering in Air Traffic Management”. PhD-Thesis, Magdeburg, Otto-von-Guericke University, 2002.

[18] M. Kreuz, M. Schultz, “System Dynamics Approach towards ANSP Modeling”. AIAA Aviation, 2015

[19] H. Trandac, P. Baptiste, V. Duong, “A constraint-programming formulation for dynamic airspace sectorization”, in Digital Avionics Systems Conference, 2002.

[20] M. Meinecke, “Entwicklung und Evaluation von Lotsenarbeitsbelastungsmodellen in einer Schnellzeitsimulations-umgebung”, Master Thesis, Braunschweig, DLR, 2014

[21] I. Gerdes, M. Schaper, „Management of time based taxi trajectories coupling departure and surface management systems“, 11th ATM Seminar, Lisboa, 2015