From Sets to Geometrical Sectors in the Airspace Sectorization Problem

Dac-Huy Tran, Philippe Baptiste, Vu Duong

***Abstract*—In the Airspace Sectorization Problem, airspace has to be partitioned into a given number of sectors, each of which being assigned to a team of air traffic controllers. The objective is to minimize the coordination workload between adjacent sectors while balancing the total workload of controllers and respecting some specific constraints. This problem is solved by a constraint programming formulation, in which each sector is defined as a set of vertices. In this paper, we present an approach to calculate the sector boundary from its vertices. The method ensures also the connectivity constraint of the sectors.**

***Index Terms*—Airspace Sectorization, Computational Geometry, Constraint Programming**

I. Introduction

Sectorization of airspace is a fundamental architectural feature of Air Traffic Management (ATM). The airspace is divided into a number of sectors, each one is assigned to a team of controllers. Controllers of a given sector have (1) to monitor flights, (2) to avoid conflicts between aircrafts and (3) to exchange information with adjacent sectors to take or to hand-on the control responsibility of flights. These tasks induce a workload which must be balanced between the sectors. On top of this balance constraint, several specific ATC (Air Traffic Control) constraints have also to be taken into account:

- *Route-based convexity constraint:* the same aircraft can not enter the same sector twice.
- *Minimum distance constraint:* the distance between a sector border and a vertex must be not less than a given distance.
- *Minimum sector crossing time constraint:* the aircraft must stay in each crossed sector at least a given amount of time.

The Airspace Sectorization Problem (ASP) is solved in a Constraint Programming [1] framework [2]. In this approach, follow the idea of [3], instead of defining sectors through a *geometric description*, we can define a sector as a convex *set of vertices*. The main advantage of this approach is to resume to a purely discrete problem. The airspace is then sectorized by the following steps:

- The airspace which is made of routes that cross each others is represented by a graph $G=\{V,E\}$. V is the set of crossing points v_i, and $(v_i, v_j)\in E$ if and only if there is a direct route from v_i to v_j. The graph G is labeled both on its vertices and edges by the static estimated workloads.
- The sectorization problem of the graph $G=\{V,E\}$, $|V|=n$, $|E|=m$ is then modeled in a constraint programming formulation by introducing n variables $x_i\in\{1..k\}$, where k is the number of sectors and $x_i = j$ means that the vertex v_i is in the subset V_j representing the sector j. The constraints of ASP are naturally stated, or by using predefined constraints of the solver, or by using user-defined constraints come with a specific constraint propagation algorithm. The optimization function is defined in terms of the minimum number of cut routes. A heuristic is also defined for variables and values ordering.
- A "good" initial solution is found by a recursive bisection scheme. At each step of bisection, solution is found with help of constraint programming formulation and then improved by the restricted Kernighan/Lin heuristic.
- At last, the solution obtained in the previous step is improved by a random local re-optimization scheme: a small group of adjacent sectors is selected randomly and again, the constraint programming formulation is used to find locally its optimal sectorization. This procedure is repeated and ends if after predefined N consecutive iterations, the solution is not longer improved.

As reported in [2], the result of this approach is promising. But we have at least two sub-problems to solve:

- *The computation of sector boundary:* until now, the sectorization is considered as a partitioning of the set of vertices and a sector is defined as a subset of vertices. From this partitioning, we must then compute non-overlapping sector boundaries such that each sector boundary contents all its vertices.
- *The connectivity constraint:* when defining the sector boundaries, we must ensure that the sectors are not fragmented. For example, the case in which sector A is fragmented in Figure 1 must be avoided:

Huy TRANDAC, Heudiasyc Laboratory, UMR CNRS 6599, University of Technology of Compiègne, Centre de Recherches de Royallieu, BP 20529, F-60205 Compiègne cedex

Philippe BAPTISTE, Ecole Polytechnique, LIX F-91128 Palaiseau

Vu DUONG, Eurocontrol Experimental Centre, Centre de Bois des Bordes, BP15, F-91222 Bretigny sur Orge cedex

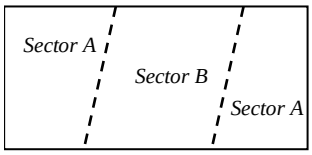

Figure 1: Connectivity Constraint

In the next, we will propose an approach, which can help us not only to compute the sector boundaries but also to ensure the connectivity constraint.

II. Sector Boundaries Computation and Connectivity Constraint

Assume that we have an airspace which has to be partitioned into a number of sectors, each represented by a set of vertices; we must then compute non-overlapping sector boundaries such that each sector boundary contents all its vertices and, as mentioned above, can not be fragmented.

For example in the 2D case, we must determine for each sector a simple polygon which contents all its vertices. We propose a way to compute such sector boundaries for the 2D airspace case. Note that it can be extended to the 3D case. The idea is to partition the plane of airspace with n vertices into n non-overlapping polygons, each polygon contain one and only one vertex. The boundary of a sector defined by a subset of vertices is computed by grouping the polygons corresponding. From this partitioning, a neighboring relative graph is determined and used to ensure the connectivity constraint.

A. Point-based Polygonal Tessellation and Neighboring Relative Graph

Definition: A *Point-based Polygonal Tessellation* of a plane with a set of n points V, denoted by $PT(V)$, is a partitioning of the plane into n non-overlapping polygons, called tiles, such that each polygon $P(v)$ contains exactly one point $v \in V$.

Definition: The *Neighboring Relative Graph* of a point-based polygonal tessellation $PT(V)$, denoted by $NRG(PT(V))$, is the graph constituted by (V,E), where edge (u,v) belongs to E if and only if $P(u)$ and $P(v)$ have at least one common side.

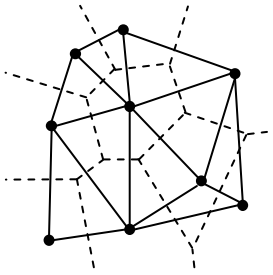

Figure 2: Delauney Triangulation (dashed line) and Voronoi Diagram

Example: The Figure 2 illustrates us an example of the case in which the Delauney Triangulation is the NRG of Voronoi Diagram [4]. The Voronoi diagram (also known as the Direchlet tessellation or Theissen tessellation) is a subdivision of a plane into a number of tiles; each tile has one sample point in its interior called a generating point. All other points inside the polygonal tile are closer to the generating point than to any other. Its dual, the Delauney triangulation (the term *triangulation* is defined in the next), is created by connecting all generating points which share a common tile edge.

Proposition: Let $PT(V)$ be a point-based polygonal tessellation of a set of points V in a plane and the corresponding $NRG(PT(V))$. For all subsets $V_i \subset V$, if a subgraph of $NRG(PT(V))$ corresponding to V_i is connected, then there exists a simple polygon which contains all points of V_i and no point in $V \backslash V_i$.

The proof of this proposition is given by construction of such a polygon: we group the tiles corresponding to all points in V_i and remove all shared sides. Since each tile contains only one point of V by the definition of $PT(V)$, this polygon contains only the points of V_i, but no point in $V \backslash V_i$.

Hence, the boundary of a sector can be obtained by grouping the corresponding tiles of its vertices and a sector V_i is un-fragmented if $NRG(PT(V_i))$ is connected.

B. Triangulation as Neighboring Relative Graph

Our problem is now how to obtain a point-based polygonal tessellation of a set of points in a plane and its neighboring relative graph. In the next, we firstly determine the neighboring relation between the vertices, which is a triangulation, and then construct the corresponding tessellation.

Definition: [5] A *Triangulation* of a set of n points V in the plane, denoted by $T(V)$, is joining the points of V by non-intersecting straight line segments such that all regions are triangles.

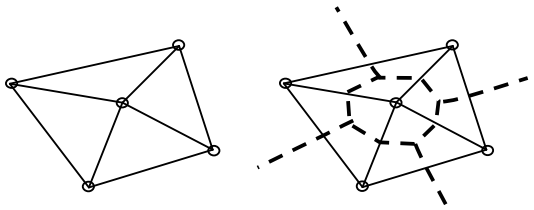

Figure 3: Triangulation of 5 points and a Tessellation (dotted lines)

Observation: For every triangulation $T(V)$, we can always determine a point-based polygonal tessellation $PT(V)$ such that $T(V)=NRG(PT(V))$.

For instance, in Figure 3, we choose a point inside each triangle and the tiles are defined by joining these points and the center points of edges.

Figure 4: Edge throughs three sectors

But in our problem, to avoid the case in Figure 4 where an edge can through three sectors, two vertices of an edge in the graph representing airspace must be considered as neighbors each other. It means that this edge belongs to the set of edges of NRG. What we need is then a *constrained*

triangulation.

Definition: [5] Given a planar graph *G=(V,E)*, a *constrained triangulation*, denoted by *CT(G)*, with respect to *G* is a triangulation *T(V)* such that all edges of *E* are edges of *T(V)*.

Figure 4 gives an example of a constrained triangulation.

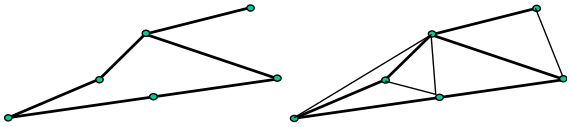

Figure 5: A constrained triangulation (right) of a given graph (left)

A constrained triangulation can be obtained by adding edges that do not intersect any of existing edges, till no more new edges can be added. This technique has a poor complexity of $O(n^4)$. In [5], an $O(n \log n)$ constrained triangulation algorithm is described.

Now let us come back to our problem: given an airspace represented by *G=(V,E)*, we construct a constrained triangulation *CT(G)* with respect to *G*. The airspace then will be sectorized into a number of subsets V_i such that, for all V_i, the subgraph of *CT(G)* corresponding to V_i is connected. From this constrained triangulation, we determine the corresponding point-based polygonal tessellation, and the boundary for each sector is constructed by grouping the tiles of its vertices.

Note that for a given graph, we can obtain several constrained triangulations. It is difficult and outside the scope of this paper to determine which one is the ``best".

C. Connectivity Constraint Propagation

We firstly recall a brief overview of the principles of Constraint Programming (CP). Constraint Programming is a paradigm aimed at solving Constraint Satisfaction Problems (CSP). An instance of the CSP is described by a set of variables $X=\{x_1, x_2, .., x_n\}$, each variable x_i has a set $D(x_i)$ of possible values (domain of variable), and a set of constraints between the variables. A *solution* of the CSP is an instantiation (assignment of values for all variables), such that all constraints are satisfied.

CSPs can be solved as follows. A tree search is created and the variables are instantiated sequentially. Each node of the tree search represents a partial solution (partial assignment) and the algorithm attempts to extend it to a full solution by assigning a value to an uninstantiated variable. Whenever a partial solution violates any of constraints, backtracking is performed.

The key idea of CP is *Constraint Propagation:* when a variable is instantiated, the constraints are not only used to check the validity of solution, but they are also used to deduce new constraints, to remove *inconsistent values* of uninstantiated variables, so to reduce the search space.

In [2], the Airspace Sectorization Problem is modeled as an instance of the CSP by introducing *n* variables $x_i \in \{1..k\}$, where *k* is the number of sectors and $x_i = j$ means that the vertex v_i is in the subset V_j representing the sector *j*.

Now to ensure the connectivity constraint, we define a *global constraint* concerning all variables and the following propagation algorithm is applied for each partition *p*, whenever a variable is instantiated:

- We first compute the set of vertices of which the corresponding variables can take the value *p*:
$$V_p=\{v_i \mid p \in D(x_i)\}$$
- Determine the connected components of the sub graph of *CT(G)* corresponding to V_p:
$$V_{p1} \cup V_{p2} \cup \ldots \cup V_{pk} = V_p$$
- No more than one of these subsets belongs to sector *p* hence, if a variable x_i in a subset V_{pl} takes already the value *p*, we can then deduce that *p* can be removed from the domain of the variables of all other subsets.

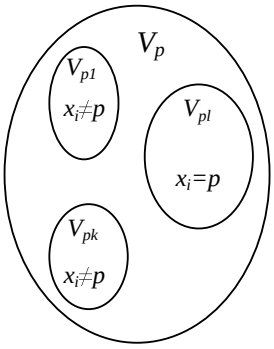

Figure 6: Connectivity Constraint Propagation. Only the variables of V_{pl} can take the value *p*

With *n* and *m* are respectively the number of vertices and the number of edges of the graph representing the airspace, the first step of the algorithm can be performed in $O(n)$; the second one is in $O(m)$ by the Tarjan depth-first search algorithm [6] and the last one in $O(n)$. We check the connectivity for all *k* partitions, the algorithm is then in $O(k.m)$.

III. Result and Perspective

This approach for sector boundary computation and the constraint propagation algorithm are implemented and integrated to complete the works in [2]. It has been tested with the generated data. Figure and give us respectively a view of a 100-vertices graph sectorized into 8 and a 500-vertices graph sectorized into 40.

As mention above, with a given airspace, we can have several constrained triangulation, and the choice arbitrary one of them may restrain the search space, hence the better solution. One of perspectives is then to build dynamically the constrained triangulation progressively with the instantiation procedure. Another work to do is the smoothing of the sector boundary, in terms of the number of its segments.

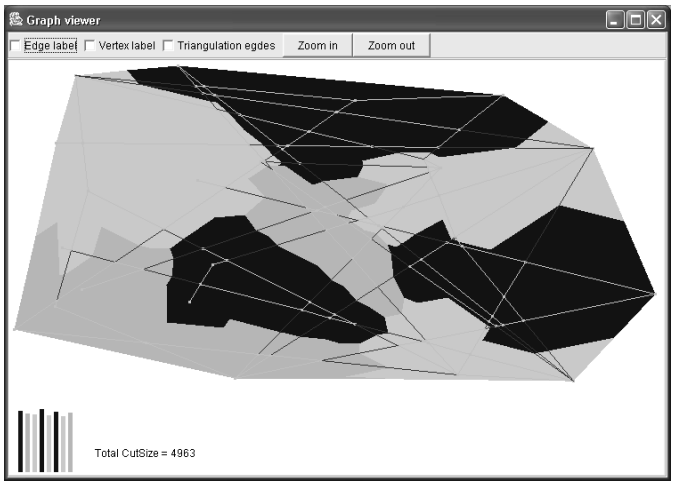

Figure 7: A 100-Vertices Graph Sectorized Into 8

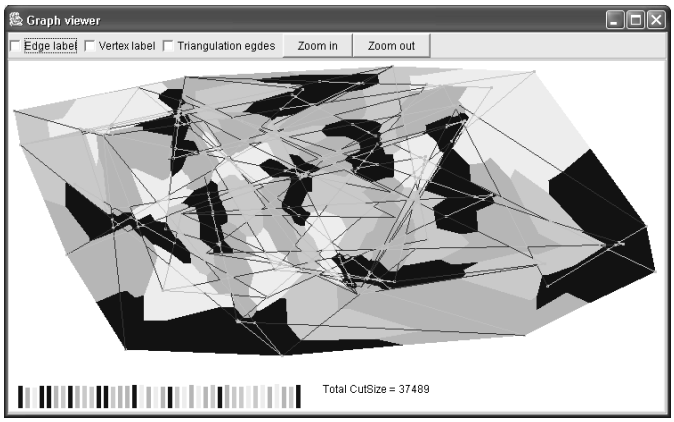

Figure 8: A 500-Vertices Graph Sectorized into 40

REFERENCES

[1] Bartak R.,1998. *Online Guide to Constraint Programming.* http://kti.mff.cuni.cz/~bartak

[2] TranDac H., Baptiste P., Duong V., 2002. *A Constraint Programming Formulation for Dynamic AirSpace Sectorization.* Proc. of 21st Digital Avionics Systems Conference.

[3] Delahaye D., M. Schoenauer and J. M. Alliot, 1998. *Airspace Sectoring by Evolutionary Computation.* IEEE International Congress on Evolutionary Computation.

[4] Okabe A. and al., 1992. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* New York: Wiley.

[5] Chen J., 1996. *Computational Geometry: Methods and Applications*. Computer Science Department, Texas A&M University.

[6] R. E. Tarjan, 1972. *Depth-first Search and Linear Graph Algorithms.* SIAM Journal on Computing, 1:146-160.