Identifying Unknown Border Lines from Historical Spatiotemporal Data: The DIACHORON Approach

Elias Frentzos, Nikos Pelekis, Nikos Giatrakos and Yannis Theodoridis

Department of Informatics, University of Piraeus,
80 Karaoli-Dimitriou St, GR-18534 Piraeus, Greece
{efrentzo, npelekis, ngiatrak, ytheod}@unipi.gr

Abstract

Recently, a special type of Multimedia Information Systems (MIS) called Cultural Heritage Management Systems (CHMS) has appeared confirming the special interest for applications accessing historical thematic data that are accompanied by spatio-temporal information. Such kind of data can easily be analysed in any Geographical Information System (GIS); however the main inherent shortcomings for enabling highly interactive usage of them is that, usually the spatio-temporal information is either missing, or it is given in a way that it cannot be directly visualized in a map by a GIS in a precise way, or it can not be inferred given certain spatio-temporal query predicates by an end user. For instance, consider an analytical query asking for the possible border between two empires given that certain cities have are occupied by each one of them. Difficulties in answering such kind of queries is not just a limitation of current GIS software, but rather an intrinsic constraint of storing the historical information in a way that does not allow inferences that result in new spatio-temporal data. In this paper we present a methodology for dealing with such kind of queries and propose efficient algorithms that produce meaningful, non-stored border lines between spatial entities, by taking into advantage their thematic characteristics as well as morphological information of the region of interest.

Keywords

Multimedia, Cultural Heritage Information Systems, Spatio-Temporal Query Processing, Voronoi Diagrams, Delaunay Triangulation

1. Introduction

Lately CHMS have gained the interest of researchers from many disciplines like historians, (anthropo) geographers, historic and thematic cartographers as the means, not only to get deeper insight into unrevealed knowledge hidden into vast historical spatio-temporal datasets, but also to provide interactive learning procedures to the general public. Despite the availability of huge spatio-temporal datasets and the advances in GIS software technology, there are certain types of analytical queries that can not be resolved by the functionality offered by state-of-the-art commercial GIS. Example queries include those that do not simply retrieve a proper subset of the database; rather the answer is new, previously unknown data that can be rationally deduced by the actual stored data. Such unseen spatial data may be further visualized and presented to the user providing new knowledge and supporting educational and research purposes.

To illustrate this idea we describe a motivating example where a user studies the historic development of the administrative areas of countries A, B in Figure 1. More specifically, consider the rectangular area where a set of spatial point entities exist. These points correspond to certain historical events and each distinct point may be associated with temporal information either as a time period (i.e. city *Γ* belonged to country *A* during the 5th century B.C.) or as a time point (i.e. a battle took place in place *Δ* that belongs to country *B*, at 323 B.C.). An implicit conclusion from the above discussion is that country *A* (country *B*) may be considered as a thematic attribute of point *Γ* (point *Δ*) events (represented by the orange (magenta) circles in the figure). In the other words, these spatio-temporal events imply an imaginary border between *A* and *B* that also defines the boundaries of these two non overlapping surfaces.

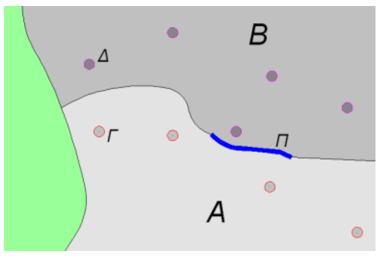


Figure 1 - Determination of administrative boundaries of A and B surfaces using a set of point references and geomorphologic characteristics

Having in hand such kind of knowledge, the aim is the approximate determination of the border line between the two countries. Obviously, the linear spatial entity representing a suitable boundary, will cross the line between points *Δ*, *Γ* and any other pair of points in the aforementioned set distinguished by their thematic characteristics. If no additional knowledge is available, we can only assume that the points of the border line should lie in the middle of each pair. However, in this case additional information can be acquired by geomorphologic knowledge of the underlying ground. In Figure 1, such knowledge comes in the form of a river *Π* and acts as another distinguishing parameter of the initial point set. Similarly, other types of background information implying physical border existence (e.g. mountain chains, lakes, coastlines etc.) may be utilized as additional disjunctive factors.

Generalizing the problem, consider that some of the thematic properties of the data points may be dynamically changing over time. For instance, a point on a map representing a village may belong to different provinces in different historical periods. Per se, the challenge is to efficiently formulate such time changing borders among regions that include a set of points of interest, by utilizing appropriately underlying geomorphologic data. In this paper we deal with this problem by proposing effective algorithms for the computation of the unknown border lines, between areas that during different time periods are associated with different spatial points. Our approach is based on the incremental Delaunay Triangulation algorithm of the Guibas and Stolfi (1985), which allows us to efficiently adjust the computed border line in user exploratory queries with different parameters.

The rest of this paper is structured as follows: section 2 outlines some basic background knowledge of our approach; the algorithms and results of our approach are described in Section 3, while Section 4 concludes the paper, giving future perspectives of the approach.

2. Background

In this section we present some basic definitions of well-known structures upon which we base our study. The first such structure, is the so-called *Voronoi diagram* (also called set of

Thiessen Polygons), which is a fragmentation of the space using polygons of irregular shape. More formally:

Definition 1 (Voronoi diagram): Given a set of points $S=\{p_i\}$, $i=1,\ldots,n$ in a 2-dimensional space, the Voronoi diagram V of S is the fragmentation of the space into a set of cells V_i such that the distance of any point p in cell V_i is smaller than its distance from any other p_i in S:

$$V = \{V_i = \{p\}: distance(p, p_i) \leq distance(p, p_j),\ \forall\ p_j \in S\}$$

Thiessen Polygons, as the outcome of Voronoi cell computation, are widely used in geosciences in order to define zones of influence for certain data points based on the criterion of spatial proximity. Figure 2 illustrates a set of random data points along with the corresponding Voronoi cells.

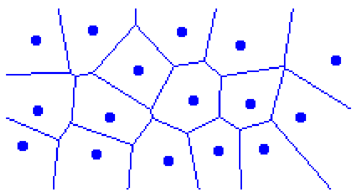


Figure 2 - Voronoi Diagram of a data points set

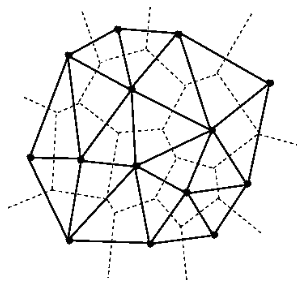


Figure 3 - Duality between Delaunny Triangulation and Voronoi Diagram

For any point $p_i \in S$, the Voronoi cell V_i is unbounded if and only if p_i is on the convex hull of S. If V_i is bounded, then V_i is a convex polygon. Since in our study we consider certain rectangular areas of interest, the Voronoi cells are always bounded. Usually, the construction of the Voronoi diagram is achieved through the computation of its dual, namely the *Delaunay Triangulation*.

Definition 2 (Delaunay Triangulation): The *Delaunay Triangulation* of a set P of points in the plane, is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$.

Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid "sliver" triangles. The triangulation was invented by Boris Delaunay in 1934. The computation of the Voronoi cells via the Delaunay Triangulation is attained by traversing the perpendicular bisectors of the edges of each triangle, and then by performing appropriate cutoffs at the intersection points of the perpendicular lines (Figure 3); moreover, these intersection points can be straightforwardly determined as the center of the circumcircle of each triangle. In the subsequent section we appropriate utilize the above mentioned structures so as to design an effective solution to our problem.

3. The proposed algorithms

Rephrasing the aforementioned basic problem, the aim of this paper is to discover the boundaries between spatial polygon entities, using the Voronoi Diagram constructed by a set of points with known thematic properties. Furthermore, in order to refine the border lines and make them more realistic, we adjust them by leveraging morphological knowledge of the underlying ground.

e.Org	The starting point of the edge.
e.Dest	The terminal point of the edge.
e.Sym	The symmetric of the edge (inverse direction).
e.Lnext	The next edge that results after rotating *e* in a clockwise fashion around its terminal point.
e.Rprev	The next edge that results after rotating *e* in counter-clockwise fashion around its terminal point.
e.Onext	The next edge that results after rotating *e* in counter-clockwise fashion around its starting point.
e.Oprev	The next edge that results after rotating *e* in a clockwise fashion around its starting point.

Table 1 – Auxiliary fields of an edge *e* in Delaunay Triangulation

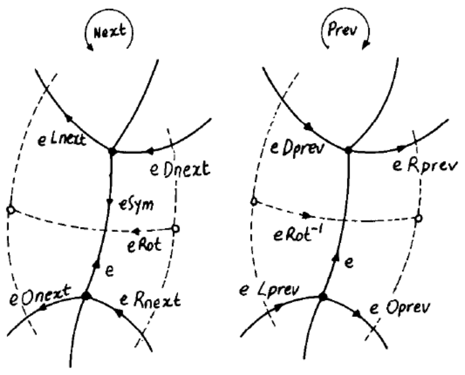


Figure 4 – Edge Functions (Guibas and Stolfi (1985))

In order to achieve this goal, we adopt notion of the Delaunay triangulation, implementing the *incremental algorithm* of Guibas and Stolfi (1985) used to efficiently construct a Delaunay triangulation. More specifically, the structure proposed by Guibas and Stolfi (1985) implements the Delaunay Triangulation as a directed graph. Each directed edge *e* is equipped with a set of seven auxiliary fields that can be used during the navigation in the graph. Table 1 exemplifies the usage of these fields, while Figure 4 describes a number of supporting functions for traversing the graph. The structure of each point participating in the triangulation process consists of the fields illustrated in Table 2

Id	The unique identifier of the point
x	The x coordinate of the point
y	The y coordinate of the point
attrib	Thematic information associated with the point; e.g the province in which the point belongs to

Table 2 - Structure of a point participating in the triangulation

The incremental algorithm of Guibas and Stolfi (1985) allow us to build the triangulation in a dynamic (incremental) way. This means that we can progressively add new points that might not be available in advance. In addition, as we may have ad hoc deletions similarly to ad hoc insertions, we adopt the algorithm proposed by Devillerst (1999), for the dynamic reorganization of the resulted triangulation after the deletion of expired (in terms of time range selection) points.

```
Algorithm Create_Polygon (Points Collection, PolyLines Collection, att string)
// Create a new Delaunay Triangulation and add all Points
TIN = NEW Delaunay_Triangulation;
FOR EACH Point IN Points
  TIN.Add Point;
NEXT;
// Determine the triangulation edges with the attrib of their starting point
   = att and the attrib of their terminal point <> att. Put the edges in the
   ColAttrib Collection
FOR i=1 TO TIN.count
  IF TIN.Edges(i).Org.attrib=att AND TIN.Edges(i).Dest.attrib<>att THEN
    ColAttrib.Add TIN.Edges(i);
  ENDIF;
NEXT;
// Cycle through all edges in the collection
DO UNTIL ColAttrib.Count=0
  First=ColAttrib(1); // First, Current and Previous are Delaunay Edges
  DO UNTIL Current=First
    Previous = Current; PreviousInterPoly=InterPoly;
// Choose the next edge among two edges on the right side of the Current one.
    IF Current.Rprev.Dest.Attrib=att THEN
      Current=Current.Dnext;
    ELSE
      Current=Current.Oprev;
    ENDIF;
    ColAttrib.Remove Current;
// Check whether Current intersects a PolyLine contained in PolyLines
    IF Intersects(Current,PolyLines) THEN
      InterPoly=Intersection(Current,PolyLines);
// Check whether Previous and Current intersect the same Polyline
      IF InterPoly=PreviousInterPoly THEN
// If so, add in the boundary the part of the polyline contained between the
   Previous and the Current edge.
        BoundaryLines.Add Part_Between_Edges(
                              Polylines(InterPoly),Previous,Current);
      ELSE
// Otherwise, add in the boundary the second half of the part of the polyline
contained between the previous and the current edge.
        BoundaryLines.Add Last_Half_Between_Edges(Polylines(InterPoly),
                                                  Previous,Current);
      ENDIF;
    ELSE
// Otherwise, check whether the previous edge intersects a polyline
      IF PreviousInterPoly<>0 THEN
// If so, add in the boundary the first half of the part of the polyline
   contained between the previous and the current edge.
        BoundaryLines.Add First_Half_Between_Edges(
                              Polylines(PreviousInterPoly), Previous,Current);
      ELSE
// Otherwise the perpendicular bisector is the boundary
        BoundaryLines.Add Perpendicular_Bisector(Lines(i))
      ENDIF
    ENDIF
  LOOP
LOOP
RETURN BoundaryLines
```

Figure 5 - The *Create_Polygon* algorithm

Having described the basic notions used in this work, we can proceed to the core of the paper; the algorithm illustrated in Figure 5 was developed so as to construct a Voronoi diagram generalization of a set of points that are available during a given time period, taking also into consideration other subsidiary linear spatial entities. In particular, the algorithm takes as arguments the set of *Points* that are available for the given time period (in terms of their coordinates, along with their thematic attribute), a set of *Polylines* representing natural boundaries, such as rivers e.t.c., and a thematic property *att* used to group points (and subsequently, regions of space) together. The algorithm initially builds a new Delaunay Triangulation (called *TIN*) by adding each one of the *Points* initially provided (lines 2-5). It subsequently determines the subset of the triangulation edges, such that the thematic property of their origin matches the one requested (*att*), being at the same time different with the corresponding property of their destination (lines 7-11). This is due to the fact that the actual border between an area including points with common attribute (same as the one requested), and all other spatial regions, passes through these edges. In the sequel, this set of edges is exhaustively examined in order to determine the actual boundary.

The algorithm tries to detect regions including points with common thematic property, being equal with the one requested. Thus, more than one, non overlapping polygons might be returned describing the area that satisfies a desired attribute. During each iteration the algorithm determines the next edge to be traversed as illustrated in Figure 6 (Lines 18-22). In particular, Figure 6 illustrates an instance during the algorithm's execution, where the *Current* edge has Origin (Destination) thematic property *A* (*B*). Thus, there are two possible cases regarding the vertex being at the right of the current edge (which is also the third vertex of the triangle): either the point has a thematic property with value *A* (Figure 6 (a)) or the property's value is *B≠A* (Figure 6 (b)). As also shown in the figure, in the first case the boundary will cross edge *Dnext*, while in the latter *Oprev* edge will be crossed.

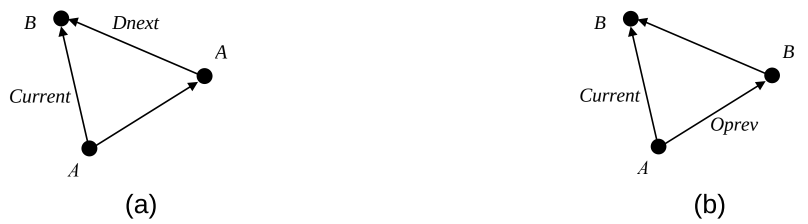
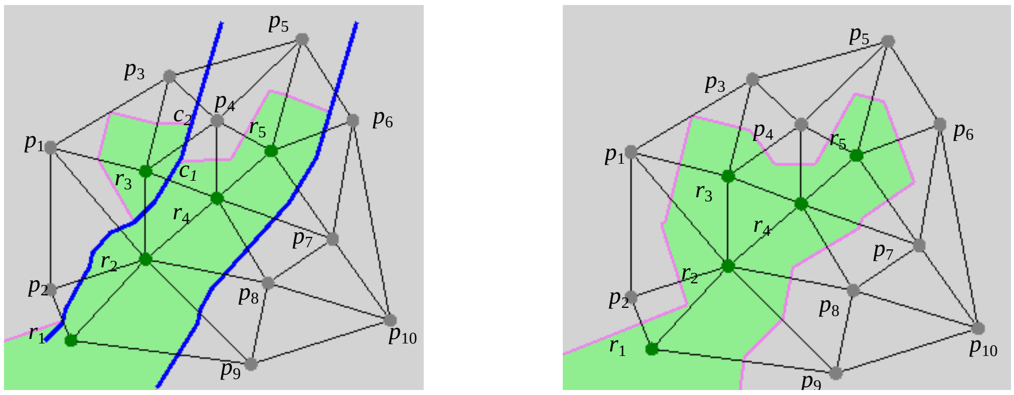


Figure 6 - Searching for the polygonal border of a region of interest

Having determined the next edge to be examined, the algorithm checks whether it intersects any of the *Polylines* in the input collection that correspond to additional geomorphologic data, such as rivers e.t.c. (Line 25). In case such an intersection exists, the algorithm checks whether this particular *Polyline* also intersects the previously examined edge (*Previous* in Line 28); if so, the algorithm adds in the boundary the entire part of the *Polyline* being between the *Previous* and the *Current* edge (Line 30); otherwise, the previous boundary point is located on the middle of the *Previous* edge, and the algorithm determines the second half of the part of the *Polyline* being between the *Previous* and the *Current* edge, and adds it in the output *BoundaryLines*. A similar procedure is also executed in the case where the *Current* edge intersects no *Polyline* (Lines 37-44), checking whether there is an intersection between the *Previous* edge and any member of the *Polylines*, and acting accordingly by adding the first half of the part of the *Polyline* being between the *Previous* and the *Current* edge; otherwise, the perpendicular bisector of the edge is added in the *BoundaryLines*. This procedure terminates after all edges in the collection have been examined, or after making a full cycle reaching the starting edge.



(a) (b)

Figure 7 - Constructing regions of interest with (a) and without (b) the usage of additional resources

The above algorithm can be further exemplified with the usage of the example illustrated in Figure 7 (a). In particular, Figure 7 (a) presents two sets of points {p_1, p_2,..p_{10}} and {r_1, r_2,..r_5} along with two rivers (illustrated with blue color); both sets share a common value for their attribute (e.g., the country on which they belong), which is represented using their color: gray regarding the first set and green for the second. Consider now the boundary line that crosses the Delaunay edge (p_4, r_3) connecting points p_4 and r_3, and, without loss of generality, assume that the algorithm approaches the edge from point r_4, e.g., the *Previous* is the edge (p_4, r_4). In this case, the algorithm comes from a normal Voronoi boundary (e.g. the boundary is the perpendicular bisector between p_4 and r_4); as such the previous boundary point is the middle of edge (p_4, r_4), and the boundary must find the point c_1 from which the *second half* of the blue *Polyline* (that crosses the *Current* edge) begins. In the sequel, edge (p_3, r_3) is examined, and given that it does not intersect any *Polyline*, the algorithm first determines point c_2 which is the middle of the blue *Polyline* (that intersected the previous edge) inside the triangle (p_3, r_3, p_4), and then, adds the middle of (p_3, r_3) as the new boundary point, and so on.

Figure 7 (b) also illustrates the result of the same procedure without taking into account the additional information provided (e.g., the blue rivers); in this case the algorithm produces significantly different and less realistic boundaries than the ones calculated by our proposal.

Having developed the *Create_Polygon* algorithm, we are able to proceed to the time focused construction of the boundaries that contain a set of points of interest sharing a common value for their thematic attributes. The *Create_Polygon_Period* algorithm, illustrated in Figure 8, begins by retrieving the set of valid points for the querying time period and by storing their identifiers in the *GeoPoints* list. The next step involves the retrieval of the available geomorphologic information stored in the *GeoLinks* list (line 7), as well as the spatial components of all points contained in the *GeoPoints* list (line 6). Finally, *Ceate_Polyon* algorithm is invoked with arguments the *GeoPoints* and *GeoLinks* lists along with the requested attribute (denoted as *att*), and outputs the *BoundaryLines* created by the respective *Create_Polygon* algorithm. The following class diagram of Figure 9 depicts the developed algorithm modules.

```
Algorithm Create_Polygon_Period(DB Database,AGIS GIS,
          t time_period, att string)
 1. // Retrieve from the database the set of points
       that are valid candidates Given the time
       period parameter.  Corresponding Ids are
       stored in the GeoPoints list
 2. GeoPoints = DB.Execute ("SELECT GEO_OBJECTS.*
                FROM GEO_OBJECTS, SPATIO_TEMP,
                TIME_STAMP, TIME_PERIOD WHERE ..")
 3. // Retrieve the set of geomorphologic data (from
       the database or selected files) Corresponding
       Ids are stored in the GeoLinks list
 4. GeoLinks = DB.Execute ("SELECT * FROM
               GEO_OBJECTS, SPATIO_TEMP,
               THEM_CATEGORIES WHERE ...")
 5. // Retrieve the geometries (in terms of the
       underlying GIS system) of the points using the
       GeoPoints list
 6. GeoPoints = AGIS.RetrieveGeometries(GeoPoints)
 7. // Retrieve the geometry (in terms of the
       underlying GIS system) of the geomorphologic
       data using the GeoLinks list
 8. GeoLinks = AGIS.RetrieveGeometries(GeoLinks)
 9. // Invoke the Create Polygon Algorithm using
       GeoPoints, GeoLinks, att as input parameters
10. BoundaryLines = Create_Polygon(GeoPoints,
                    GeoLinks, att)
11. RETURN BoundaryLines
```

Figure 8 - The *Create_Polygon_Period* algorithm

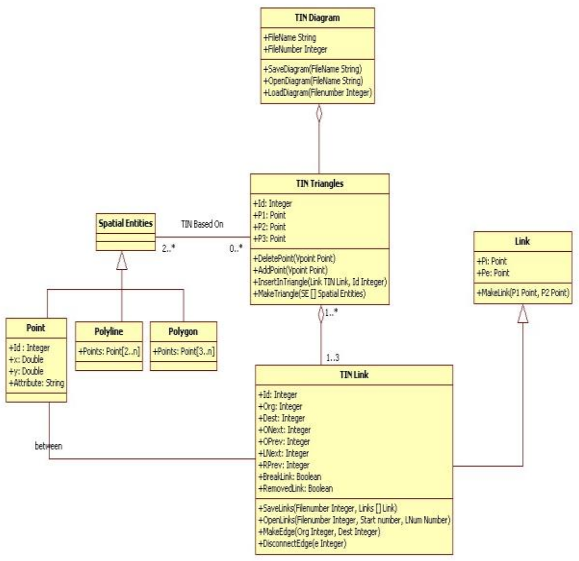


Figure 9 – Implementation Classes of the *Create_Polygon_Period* Algorithm.

4. Conclusions

In this paper we proposed a methodology for supporting a special kind of query that determines the spatial border lines between regions that are distinguished via different thematic properties, as well as, additional morphological information of the region of interest. The contribution of this work is straightforward as, to the best of our knowledge such queries are not supported by current GIS software. As future work we plan to further improve the efficiency of the processing mechanism as to be able to provide the resolution of successive queries as real-time streaming video to end users.

5. Acknowledgement

This research work is supported by the Diachoron project, founded by the Greek Ministry of Development, General Secretariat for Research and Technology, co-funded by the European Union. URL: www.diachoron.gr.

6. References

Guibas, L., Stolfi, J. (1985) "*Primitives for the Manipulation of General Subdivisions and Computation of Voronoi Diagrams*". ACM Trans. Graph. 4(2): 74-123.

Devillerst, O. (1999) "*On Deletion in Delaunay Triangulations*", Symposium on Computational Geometry.