NEIGHBORHOOD COMPUTATIONS FOR LARGE SETS OF DATA POINTS

Kurt Brassel
State University of New York at Buffalo

INTRODUCTION

The growing use of cartographic data banks for mapping calls for an efficient and flexible organization of geographic base files. Traditionally, spatial data are stored by overlaying a grid on the area of interest and recording the properties of the grid cells. In another approach the coordinate values describing the perimeter of an area unit are used for location identification.

Recently Peuker and Chrisman have reported on projects ('Geographic Data Structures' and 'Geograf') that utilize more advanced storage techniques (Peuker and Chrisman, 1975). Peuker stresses the need to assign a list of neighbors to each spatial entity. In such a system data on a statistical unit would include, for example, not only a description of the polygon outline, the center identifiers, and information on the characteristics of the unit, but also a list of identifiers corresponding to the areas contiguous to it. Neighborhood relationships can also be established for point-based data sets by assigning to each data point labels that identify its neighbors. This neighborhood relationship can be illustrated by a triangulation pattern in which each data point is connected with its neighbors by a line. Since there are various ways to define data points as being neighbors of a point A, the selection of neighbors is an ambiguous process. Among various triangulation approaches Delaunay triangles can be used to determine the neighborhood of a data point (Boots, 1974).

Another way of defining a neighbor-based data structure for a set of data points is to attribute an 'area of influence' to each data point and thus to create a set of polygons describing the 'pattern of influence' of the set of data points. Such a pattern can be created by drawing perpendicular bisectors between neighboring data points

Editor's Note: The author has expressed his gratitude to Professor T. K. Peucker who provided funds from his research grant to make this project possible; thanks are also due for his suggestions leading to this algorithm, and general advice. The project has been subsidized by funds of the Swiss National Foundation (Schweizerischer Nationalfonds), the Office of Naval Research, and the Laboratory for Computer Graphics and Spatial Analysis, Harvard University.

The author has also expressed his thanks to R. Fowler, Simon Fraser University, for his testing and revision of the computer program to this algorithm.

as seen in Figure 1. The resulting polygons are called Thiessen polygons. The major problem in generating Thiessen polygons, however, is to specify the Thiessen neighbors, i.e., the data points whose Thiessen polygons are contiguous to the Thiessen polygon of the point in question. Once these neighbors are found, they can be used to generate a triangulation pattern. By definition Delaunay triangles correspond to the connection of Thiessen neighbors. The Thiessen approach, therefore, can be used to generate both types of base files: to store triangulation as well as space allocation models.

Various authors have discussed the generation of Thiessen polygons (Thiessen 1917, Whitney 1929, Keeney 1971), and recently Rhynsburger (1973) has published a method for analytic delineation of Thiessen polygons. Searching for the Thiessen polygon about a data point A, he constructs perpendicular bisectors to all other points of the data set and extracts the innermost polygon formed by these lines about the center. This innermost polygon he calls the 'interior envelope' (Rhynsburger 1973, p. 137). Since this process of generating interior envelopes is repeated N times (where N = number of points in the data set), about N*N*6 perpendicular bisectors are to be generated, which is inefficient for large sets of data points. The basic source of this inefficiency is that Rhynsburger uses a global method to solve a local problem. In order to define the polygon outline of one data point, his algorithm consults the whole population of points, rather than only points in the neighborhood. The problem, of course, is that the neighbors are not known at the beginning of the procedure.

The present paper describes another approach to generating Thiessen polygons which uses local processes and thus is suitable for processing large sets of data points.

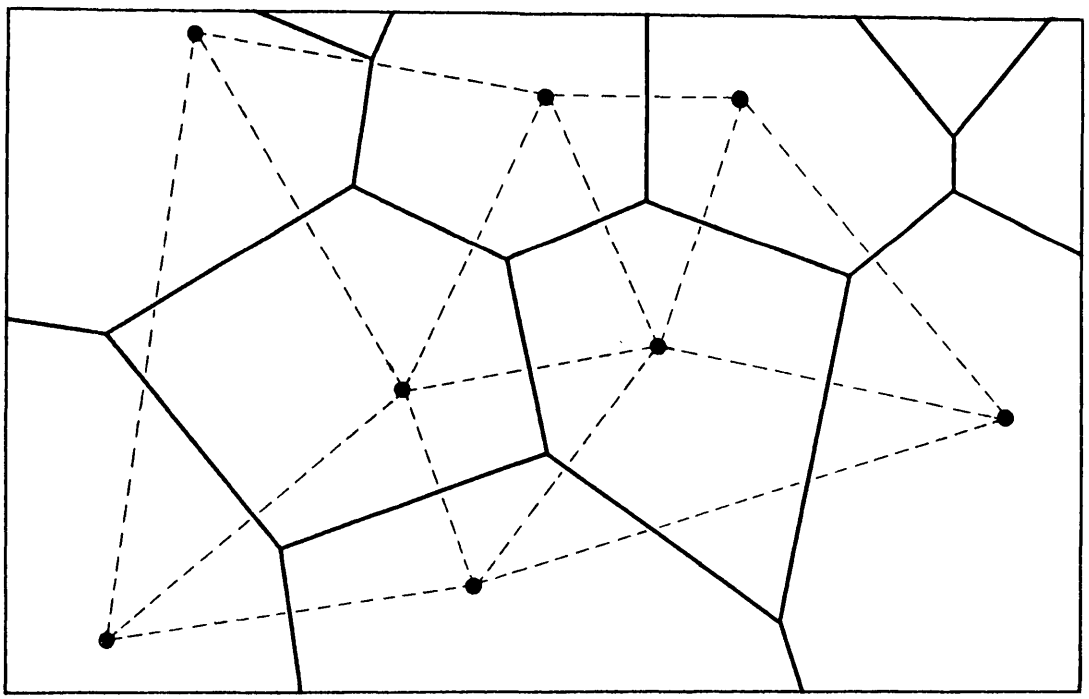


Figure 1. Sample of a Thiessen polygon structure showing the data points, Thiessen polygons (solid lines), and Delaunay triangles (dashed lines).

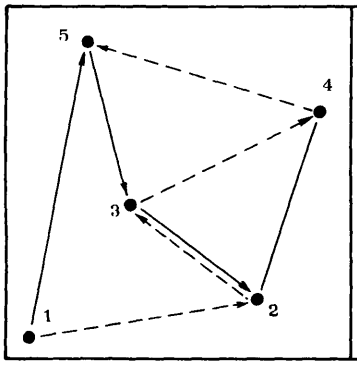
BASIC CONCEPT OF THE PRESENT ALGORITHM

In this discussion a centroid is considered equivalent to a data point. A polygon edge is defined as the locus of all points equidistant from two centroids and not closer to any other point in the data set. Locations equidistant to three centroids (and not closer to any other centroid) are called vertices. Pairs of polygons with a common edge are called neighbors (Thiessen neighbors), and pairs of polygons with a single vertex in common are called half-neighbors. The terms "neighbor" and "half-neighbor" are pertinent to both polygons and data points.

Given a set of data points in the plane, the first step sorts these points in both x- and y- directions, and assigns to each point the following pointers:

-XUP pointer, points to the next centroid in positive x-direction
-XDN pointer, points to the next centroid in negative x-direction
-YUP pointer, points to the next centroid in positive y-direction
-YDN pointer, points to the next centroid in negative y-direction

The XUP and YUP pointers are visualized as arrows in Figure 2.



Point identifier	Coordinates x	y	Pointers: Positive x direction XUP	Negative x direction XDN	Positive y direction YUP	Negative y direction YDN
1	1.0	1.0	5	-	2	-
2	11.0	2.5	4	3	3	1
3	5.5	6.5	2	5	4	2
4	13.5	10.5	-	2	5	3
5	3.5	13.5	3	1	-	4

Figure 2. Pointer structure used for the computation of Thiessen polygons.

The basic strategy used for the computation of Thiessen polygons is to first find edges, vertices, and neighbors of a first point A (compare Figure 3). Assuming that a first true neighbor X of A is known, we are searching for the next true neighbor of A in clockwise direction and find centroid B and a vertex V_1. This clockwise search is repeated until the originally known point X is found as the next true neighbor in clockwise direction (X→B→C→D→Y→X). At this time the entire polygon about A is known, and the search is continued by finding the polygon about a centroid contiguous to point A, e.g. point B. Already known neighborhood relationships ('B is neighbor of A') and known vertices and edges, however, are not computed again, but reused in this step. The search continues by completing all polygons contiguous to A, and then all polygons contiguous to the polygon about B are computed, etc. Starting at data point A, the search is spreading radially, and it is completed as soon as the farthest polygon is processed.

Given this strategy, the following problems remain to be solved:

1. To develop a procedure which finds the next true neighbor (B) in clockwise direction of an already known true neighbor

(X). Implicitly this problem is identical to finding a vertex V delimiting the perpendicular bisector between A and X.

2. To find a first neighbor X of an area A.

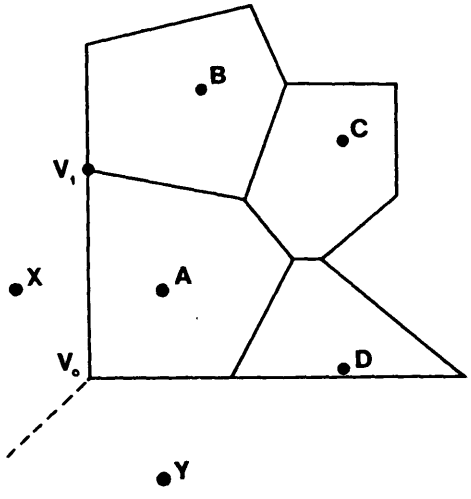


Figure 3

THE SEARCH FOR A THIESSEN NEIGHBOR

The search for the next true neighbor of A in a clockwise direction from B is performed essentially in three steps:

- find a possible neighbor
- find a highly probable neighbor
- find the next true neighbor and the respective vertex.

The first two steps are illustrated by Figure 4 below. Assuming that A is the centroid of interest and point B is known to be a neighbor of A, then we assume the next neighbor (clockwise) will be within the sector defined by the lines x_B, ab, and y_A. We are therefore searching for a highly probable neighbor C' within this sector. This is done by starting at the known points A and B and examining the data points in the sequence of either of the four pointer structures XUP, YUP, XDN, or YDN. Starting at point B we proceed along the XUP (positive

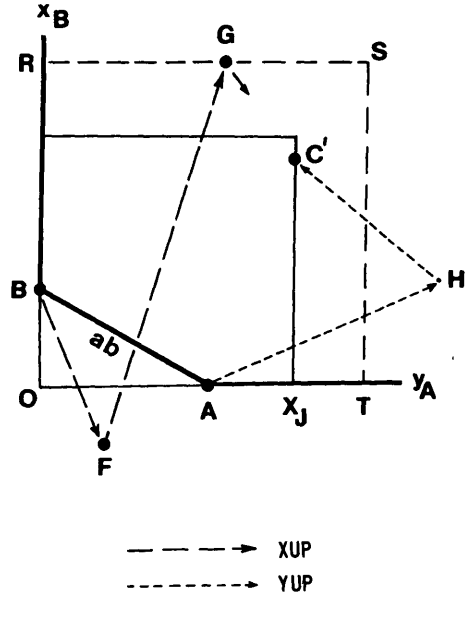


Figure 4. Search for a highly probable neighbor. Point A is the centroid of interest, B is already known as being a neighbor of A. The highly probable next neighbor C' is found by alternately following the pointer sequences in positive x-direction (XUP) and positive y-direction (YUP), and picking points in an 'innermost' truncated square. Depending on the relative position of points A and B the search may be performed in another quandrant and with other pointer sequences from among XUP, XND, YUP, and YDN.

X direction) sequence until we find a point within the quadrant (B→F→G). Point G is a possible neighbor and, in connection with points A, B, and G, implicitly defines a truncated square (ABRSTA) which is used for the search for a highly probable neighbor. Starting from point A and proceeding along the YUP pointer sequence (positive y-direction), a point C' is found within the said truncated square. C' defines a still smaller truncated square, and the search for a point within this new truncated square is continued along the XUP sequence (starting from point G). Switching between these two pointer sequences, the search is repeated until no further point is found in the innermost square. The point creating that square (C') is then called a highly probable neighbor.

In most case the highly probable neighbor C' is the next true neighbor (Figure 5). Even though point C' defines a smaller square with respect to A and B, point C

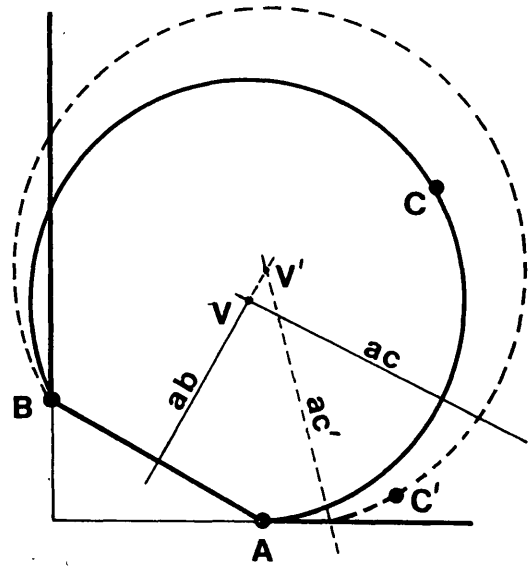


Figure 5. Search for a true next neighbor (circle test). Given a highly probable neighbor C' with respect to points A and B, a vertex V' (circumcenter of the triangle ABC') and a circle is defined. If there is no other data point within this circle, then C' is a true next neighbor, otherwise such a data point (C) is the new highly probable neighbor, and the circle test is repeated.

is the next true neighbor in clockwise direction. V' is the polygon vertex as defined by points A, B, and C', and V is defined by A, B, and point C. By definition V' cannot be a Thiessen polygon vertex because a point C is closer to V' than the three equidistant points A, B, and C'.

To replace the highly probable neighbor C' by the true next neighbor C, the circle through points A, B, and C' (centered in V') is drawn and a search for centroids within that circle is performed. In this search -- which we call the 'circle test' -- the discussed pointer structure is used in a similar fashion as before. If a point is found within the circle, a new (smaller) circle is constructed and the circle test is repeated. Point C is a true next neighbor if there is no centroid within the circle about A, B, and C.

If point B lies in another quadrant with respect to point A, the search is conducted in an appropriate sector using an appropriate pair of pointer sequences (XUP/YDN, XDN/YDN or XDN/YUP).

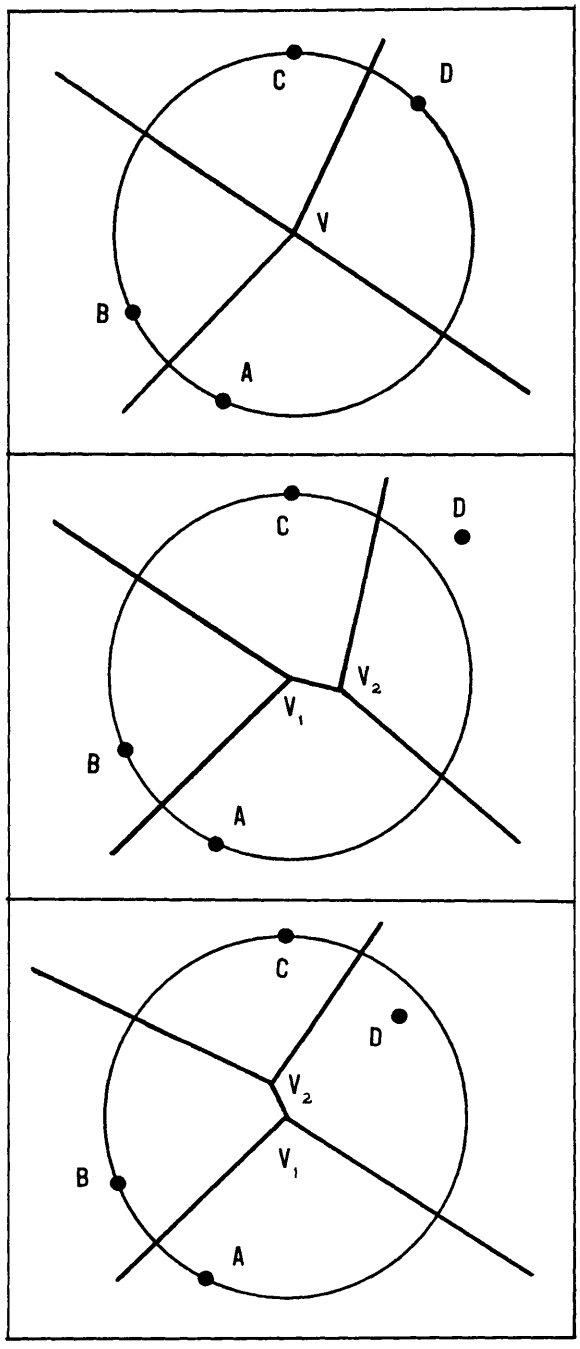


Figure 6. Four data points located on the same circle about the circumcenter V are pairwise halfneighbors (a). By shifting one data point (D) by an infinite small distance the halfneighbor relationships are eliminated (b,c).

If a point D (Figure 6) is located on the outline of the circle as defined by A, B, and C (and assuming that no point is within the circle), then the polygons about A, B, C, and D have one point in common, and two of these centroids are pairwise half-neighbors each. In this situation the new point D is shifted by a very small amount so that it lies either within or outside the circle (Figure 6, b, and c).

So far we have assumed that we can find at least one data point in a search sector as shown in Figure 4. A further assumption was that at least one neighborhood relationship is known as we start our procedure.

The basis of these two assumptions is demonstrated by the procedure discussed in the next section.

BORDER PROBLEMS AND INITIALIZATION

With respect to Thiessen polygon generation, a set of data points in the plane can be divided into two subsets: one type of data points generates closed polygons (Polygons 4-9 and 12 in Figure 7), the other subset generates open polygons (1-3, 10, 11, 13-15). For all points generating open polygons there will be at least one sector as defined in Figure 4 in which no next neighbor will be found. In order to allow the search procedure as outlined in the previous paragraph to be used for both cases, the following solution has been found:

- The set of N Thiessen polygons is assumed to be bounded by a rectangular border line (parallel to the rectangular coordinate system used). The four border lines can be arbitrarily defined.
- Four dummy data points are added to the data set, one across each border line.

The rectangular border line and the location of the four dummy points are illustrated in Figure 8. Assuming that the polygon about point A is searched for, the four dummy points are situated across each four border lines such that the line connecting point A with any dummy point is perpendicular to the respective border, and divided in half by it. In other words, if point A originally generates an open polygon, at least one of the four dummy points will be found as a true next neighbor, and the perpendicular bisector between point A and this dummy point represents exactly the predefined border outline.

After the polygon about A has been computed, the dummy points have to be shifted to a position across the newly processed centroid B. This implies a change of both the coordinate values of the dummy points and their location in the pointer structure. If a delimitation of the set other than by a rectangle is desired, then the arbitrary borders have to be positioned an adequate distance from the data points and the polygon structure has to be clipped in a subsequent step (Figure 7).

This concept of using a predefined border and dummy points allows for initializing the search procedure as well. A corner of the bounding rectangle by definition is an element of the centroid closest to it. Assuming that A in Figure 3 is closest to the corner point V_o, then V_o is a Thiessen vertex generated by A and its two dummy points X and Y. X then can be used to find the true next neighbor in clockwise direction (B).

CONCLUSION

This paper has presented an algorithm for finding Thiessen polygons and Thiessen neighbors for a set of N data points. Given four border lines delimiting the set of data points, the process picks out the data point closest to the lower lefthand corner, finds its neighbors and computes its Thiessen polygon. This is done point by point in clockwise order. The search for one neighbor point includes a step to find a highly probable neighbor within a predefined sector and, as another step, the 'circle test', which finds the true next neighbor. In both these steps only data points in the neighborhood of the data point in question are consulted. For this search four additional dummy data points are added to the data set. These points are located in a way that open polygons are delimited by border lines as defined above. Once the polygon closest to the lower lefthand corner is generated, the polygon contiguous to it is processed. Ideally the processing spreads radially and is completed as the polygon farthest to the lower lefthand corner is determined. Since this algorithm finds Thiessen neighbors and polygons in a local process, it can be applied for large sets of data points.

A PL/1 program of this algorithm is being developed, and a first test run for 300 data points has resulted in an execution time of about 10 seconds on an IBM 370/165 (Figure 9). This program is presently being further tested and revised by R. Fowler, Simon Fraser University.

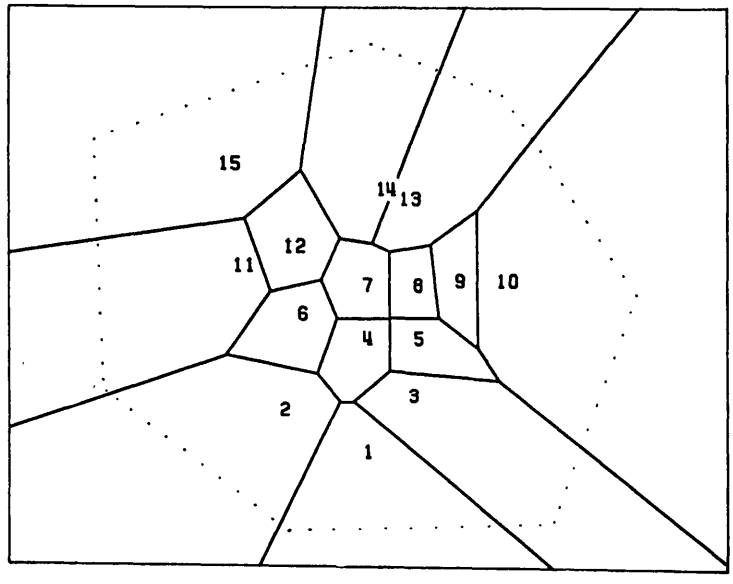


Figure 7. Closed and open Thiessen polygons. The possibility of polygon clipping is indicated by a dotted outline.

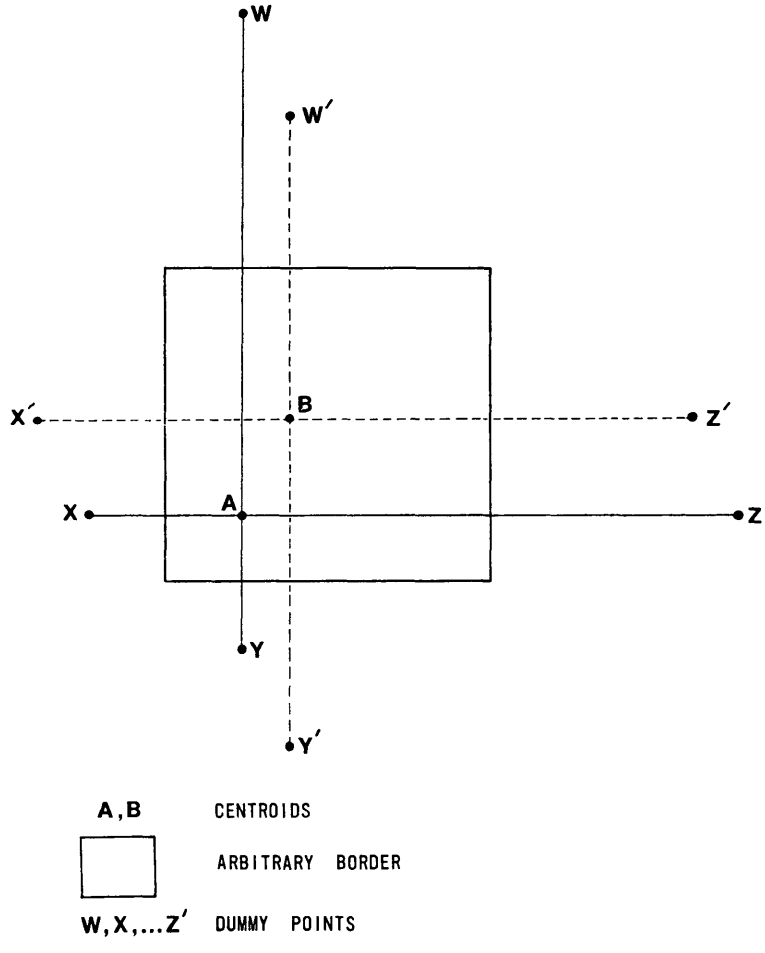


Figure 8

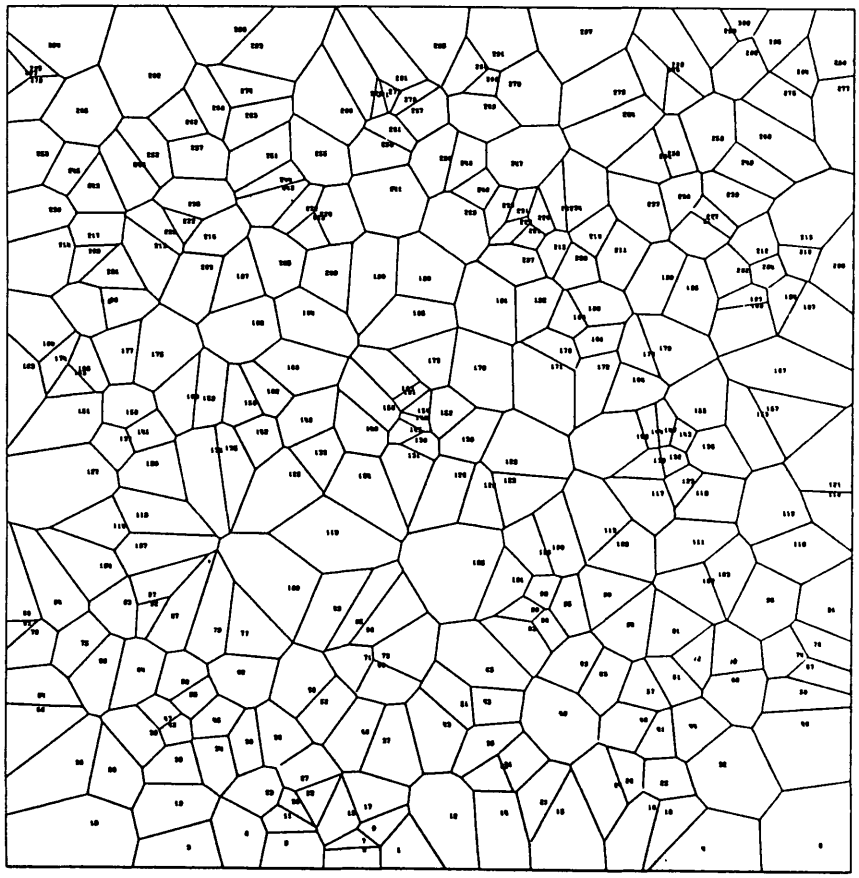


Figure 9. Sample plot of Thiessen polygons (300 randomly generated data points).

REFERENCES

1. Boots, B.N., "Delaunay Triangles: An Alternative Approach to Point Pattern Analysis," Proceedings of the American Association of Geographers, Vol. 6 (1974), pp. 26-29.

2. Keeney, R.L., "A Method for Districting Among Facilities," Operations Research, Vol. 20, No. 3 (1972), pp. 613-618.

3. Peucker, T.K., "Geographical Data Structures Report After Year One," Simon Fraser University, 1974.

4. Peucker, T.K., and N. Chrisman, "Cartographic Data Structures," The American Cartographer, Vol. 2, No. 1 (1975), pp. 55-69.

5. Rhynsburger, D., "Analytic Delineation of Thiessen Polygons," Geographical Analysis, Vol. 5 (1973), pp. 133-144.

6. Thiessen, A.H., "Precipitation Averages for Large Areas," Monthly Weather Review, 39 (1911), pp. 1082-1084.

7. Whitney, E.N., "Areal Rainfall Estimates," Monthly Weather Review, 57, (1929), pp. 462-463.