An algorithm and ArcGIS tool to create Order-2 Voronoi Diagrams

Francisco Barba

This version: October 2018

Abstract

This paper presents an algorithm to generate Order-2 Voronoi Diagrams, implemented in an ArcGIS tool. Given a set of points in the plane, an Order-2 Voronoi Diagram splits the plane into disjoint areas that are nearest in linear distance to pairs of neighbor points (dyads). The tool is available for download from `https://github.com/FMBarba`.

1. Introduction

This paper presents an algorithm to generate Order-2 Voronoi Diagrams that split the plane into polygons (henceforth: Dyadic Polygons) which is implemented in an ArcGIS tool:`https://github.com/FMBarba/`. For a detailed discussion of Higher-order Delaunay triangulations, see Gudmundsson et al. (2002) [1].

2. Definition

Given a finite set of distinct points $P = \{p_1, p_2, ..., p_n\}$ in the two-dimensional plane $\mathcal{R}^2$, a *Dyadic Polygon* D_{ij} for points p_i and p_j satisfies the condition that the sum of linear distances from p_i and p_j to any point x within polygon D_{ij} is minimal compared to any other pairing of points in the set[1].:

$$D_{ij} = x : \{d(x, p_i) + d(x, p_j) < \ d(x, p_k) + d(x, p_l)\}, \ \forall \ \{i, j\} \neq \{k, l\} \tag{1}$$

where $d(x, p_i)$ denotes the linear distance from point x to point p_i. Therefore, any given point x is part of dyadic polygon D_{ij} if p_i and p_j solve the minimization problem:

$$\min_{p_i, p_j} d(x, p_i) + d(x, p_j) \ \forall i \neq j \tag{2}$$

An algorithm sufficient to find the optimal p_i and p_j for any given x must assign the nearest and second nearest point from vector P to any given x in the plane.

3. Required point comparisons

The following section proves that the number of point comparisons required to define Dyadic Polygons can be reduced by considering exclusively points that are *Thiessen point neighbors*, as defined in continuation. The proof makes use of three properties of Thiessen polygons:

1. Given a set of distinct points p in a plane $P = \{p_1, p_2, ..., p_n\}$ a Thiessen polygon satisfies the condition that each point x within polygon T_i is nearest to p_i than to any other point: $T_i = x : \{d(x, p_i) < \ d(x, p_j)\}, \ \forall j \neq i$

2. Thiessen polygons split the entire plane into disjoint areas (cite).

3. Contiguous Thiessen polygons share a common border (cite).

In continuation, two points p_i and p_j are referred to as *Thiessen point neighbors* if their respective polygons T_i and T_j share a common border as mentioned under 3. above.

[1] Note that possible pairings $\{i, j\} \neq \{k, l\}$ exclude identical pairs, e.g. $\{i = k, \ j = l\}$ but allows for the possibility of overlap in one point of the set, e.g. $\{i = k, \ j \neq l\}$

Lemma. *If x is a point within dyadic polygon D_{ij}, then p_i and p_j are contiguous Thiessen point neighbors.*

Proof by contradiction. *Given a set of distinct points p in a plane $P = \{p_1, p_2, ..., p_n\}$, assume there exists a dyadic polygon D_{ij} whose points p_i and p_j are not Thiessen point neighbors. Any given point x will necessarily fall into the area of a Thiessen polygon of a point $p_i \in P$, given the property of Thiessen polygons to split the plane into disjoint areas (cite). Denote T_i the polygon into which point x falls, and $p_i \in P$ the centroid assigned to T_i. By property of Thiessen polygons, the distance from x to p_i is smaller than to any other point p_j in P. Thus, we can denote p_{i^*} the nearest point to x from set P and substitute into the minimization problem:*

$$\min_{p_j, p_{i^*}} d(x, p_{i^*}) + d(x, p_j) \ \forall j \neq i^* \tag{3}$$

Dropping the constant $d(x, p_{i^*})$ from the minimization problem simplifies to finding the second closest point to x:

$$\min_{p_j} d(x, p_j) \ \forall j \neq i^* \tag{4}$$

Denote T_j the Thiessen polygon of p_j. Since p_i and p_j are not Thiessen point neighbors, polygon T_j and T_{i^*} do not share a common boundary. Denote $\overline{p_j x}$ the line connecting p_j and x. Given T_j is not a contiguous neighbor of T_{i^*}, line $\overline{p_j x}$ will cross the boundary of a neighbor polygon of T_{i^*}. Denote T_n the neighbor polygon of T_{i^*} crossed by line $\overline{p_j x}$, and p_n the centroid of T_n. Denote C the point where line $\overline{p_j x}$ crosses the boundary between T_n and T_{i^*}. Line $\overline{p_j x}$ can be decomposed into lines $\overline{p_j C} + \overline{Cx}$. By property of Thiessen polygons, the distance from p_n to point C is smaller than to any other point[2]. Hence, $\overline{p_n C} < \overline{p_j C}$, and $\overline{p_n C} + \overline{Cx} < \overline{p_j C} + \overline{Cx}$, or $\overline{p_n C} + \overline{Cx} < d(p_j, x)$. But the linear distance from p_n to x is smaller or equal than the distance from p_n via intermediate point C, which gives: $d(p_n, x) \leq \overline{p_n C} + \overline{Cx}$. Hence we have $d(p_n, x) \leq \overline{p_n C} + \overline{Cx} < d(p_j, x)$, or:

$$d(p_n, x) < d(p_j, x) \tag{5}$$

But this contradicts the assumption p_j was part of D_{ij} if p_i and p_j are not Thiessen point neighbors. Hence, if x is a point in dyadic polygon D_{ij}, then p_i and p_j must be contiguous Thiessen point neighbors.

4. The algorithm

The result from the Lemma can be used to reduce the number of eligible points that form a Dyadic polygon to points which are *Thiessen point neighbors*, e.g. their Thiessen polygons share a common boundary. Dyadic Polygons can

[2] Note that point C is shared between T_n and T_{i^*}, since the all points on the boundary line have minimal distance to both T_n and T_{i^*}.

be constructed by iteratively overlapping Thiessen Polygons and merging the resulting neighboring objects, as shown in the graphical example section 5. The algorithm to define Dyadic Polygons can be described as follows:

```
import point list P
generate Thiessen polygons for points in list P
        output: thiessen polygons T
create empty list  L[]

for each point in list P
        select one point p
        select the polygon with centroid p from objects T
                selected object: thiessen polygon t
        inverse selection: select all points other than p
                selected objects: Pj
        generate Thiessen polygons for points in list Pj
                output: thiessen polygons Tj
        spatially overlap polygon t with polygons T
        keep common overlap
        fraction t into objects disected by T
                output: polygons d inside t
                save object IDs of d:
                        i: source polygon t
                        j: neighbor polygon from Tj
        add     polygons d to empty list  L[]

merge polygon list L
for each polygon d in L:
        select one polygon d
        if {source IDi = neighbor IDj
            and neighbor IDi = source IDj}
                merge source and neighbor polygons
                delete merged polygons from list L
```

5. Example

This section shows a graphical example of the algorithm. A simple representation of Dyadic Polygons is the case where input points create Thiessen polygons which are regular hexagons.

The algorithm begins by generating Thiessen polygons from all input points, as shown in the first panel of Figure1. In a second step, operations are iterated for each point in the list. The first iteration is shown in the second of Figure1. First, point A is selected from the point list and the Thiessen polygon with centroid A is selected. Second, Thiessen polygons are generated for all points other than A. In a third step, the spatial overlap between the Thiessen polygon of Point A and the Thiessen polygons for all points except A is created. This fractions the inside of the polygon of point A. Repeat the same operations for the next point in the list, here: B. The merging of neighboring parts of the interior fractions into Dyadic Polygons is shown in Figure2. Panel one of Figure2 shows the merge operation for the iterations of points A and B from Figure1, while panel two of Figure2 shows the remaining merges for all other points in the list. The sub-figure in the third column and third row of last of Figure2 shows the resulting Dyadic Polygons. Note that for present special case where points generate regular hexagon Thiessen polygons, Dyadic Polygons can be created by connecting the centroids of the polygons with its vertices. With irregular distributions of points in space, the shape of Dyadic Polygons become more uneven, as shown for a sample points in Figure3.

Figure 1: Algorithm Example Part I

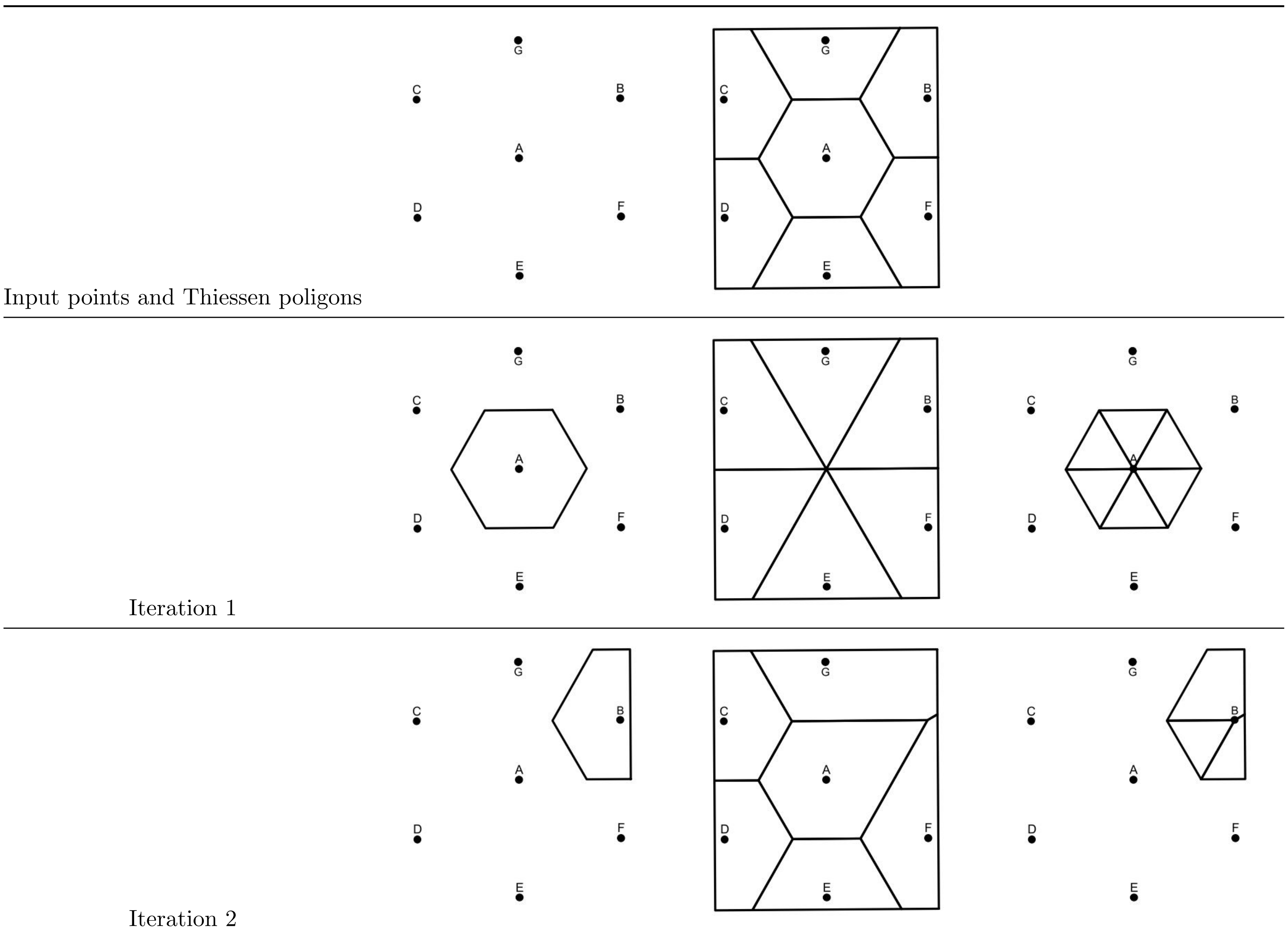
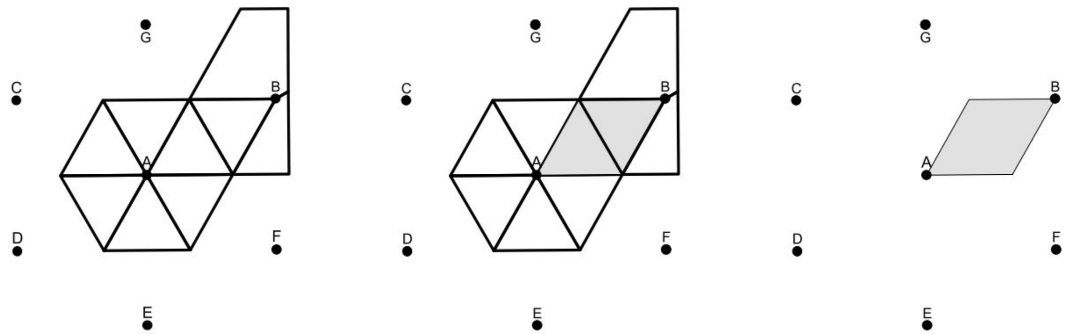
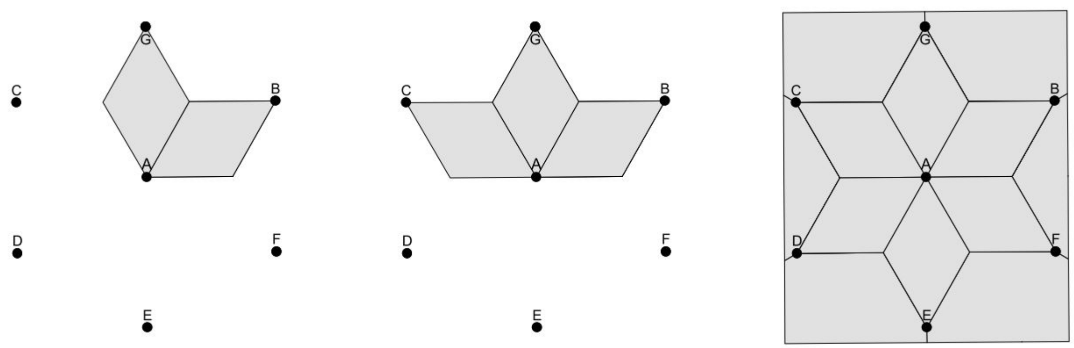


Figure 2: Algorithm Example Part II

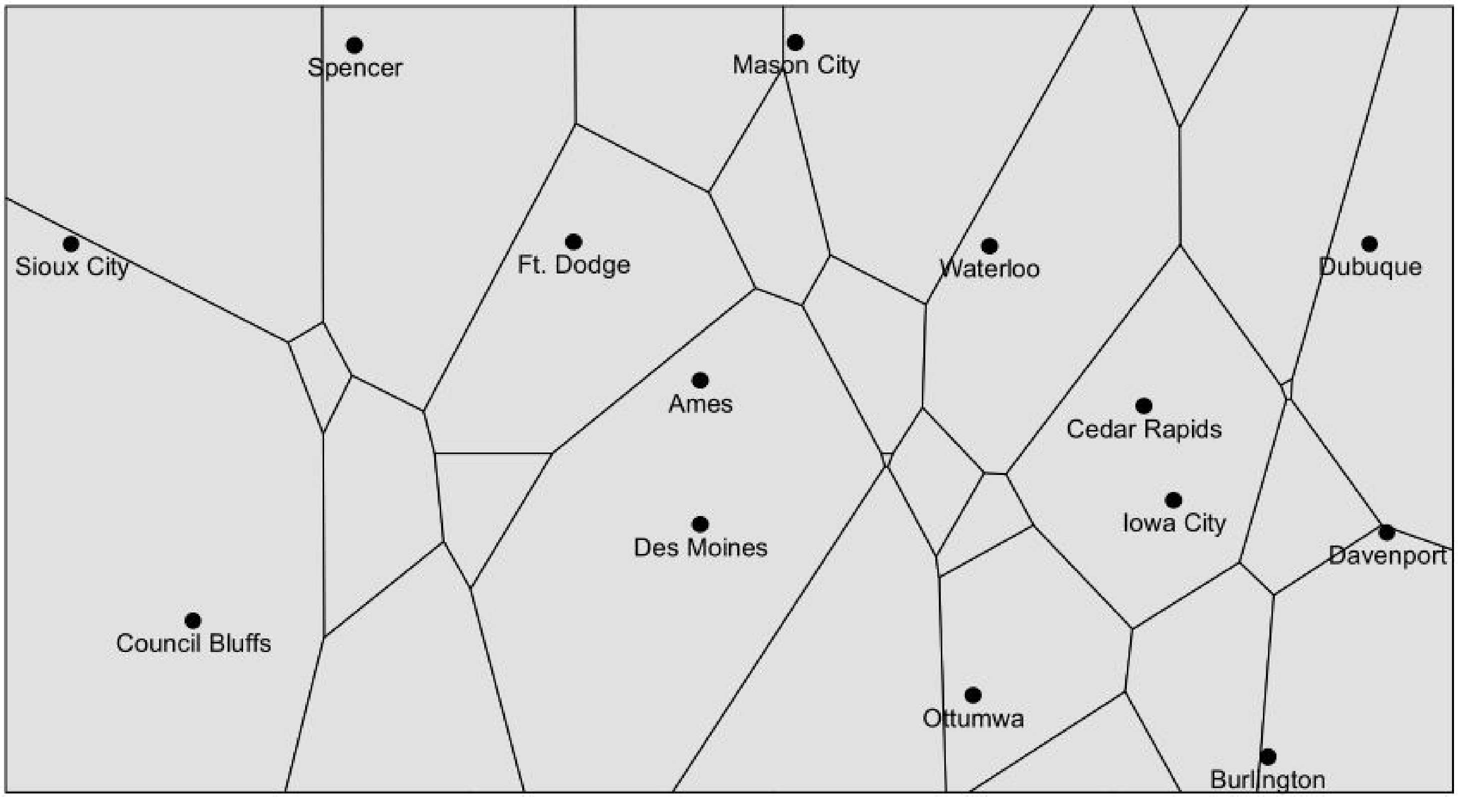


Merge neighbor polygons



Iterate over all neighbors

Figure 3: Example Dyadic Polygons for cities from Iowa, USA



6. Bibliography

[1] J. Gudmundsson, M. Hammar, M. van Kreveld, Higher order delaunay triangulations, Comput. Geom. 23 (1) (2002) 85–98.