The Death Map

John Burkardt
Interdisciplinary Center for Applied Mathematics &
Information Technology Department
Virginia Tech

....

Presentation to the Mathematics Club
University of Pittsburgh at Greensburg

....

https://people.sc.fsu.edu/∼jburkardt/presentations/...
death_map_2008_upg.pdf

14 November 2008

Casebook: The Giant's Causeway

Casebook: The Giant's Causeway

On the northeast coast of Ireland, there is a "paved" area of 40,000 interlocking (mostly) hexagonal stone pillars, roughly the same size.

Some of the pillars reach up to a cliff, and form a staircase that disappears into the sea.

How can any physical process create such patterns?

Casebook: The Giant's Causeway

Casebook: The Spitting Fish

Casebook: The Spitting Fish

The *Tilapia mossambica* is sometimes called the spitting fish.

1. It raises its young in its mouth, spitting them out when they're ready.
2. It breeds by building a nest on the river bottom, picking up stones and spitting them away

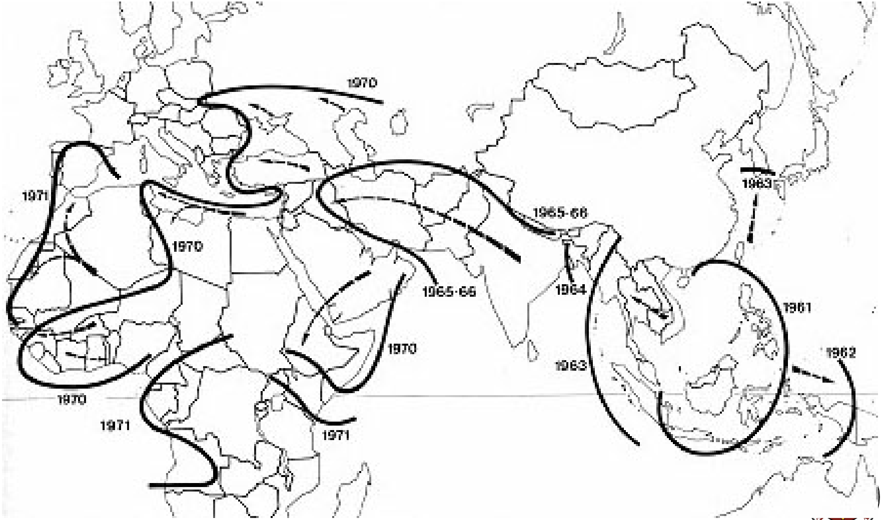
Without taking geometry class, these fish build polygonal networks of nests

Casebook: The Spitting Fish

Casebook: The Spitting Fish

- Each cell is about the same size;
- Each cell is about the same shape;
- The cells are "centered".

As far as we know, the fish do not use a blueprint, nor do they have a planning committee meeting!


1970
1971
1970
1963
1965-66
1965-66
1964
1961
1970
1963
1962
1970
1971
1971

Casebook: The Death Map

In the early nineteenth century, European doctors working in India reported a strange new disease called **cholera**.

Cholera was agonizing and fatal.

No one knew how it was transmitted. No one had any idea how to treat or prevent it.

Yearly reports showed that cholera had begun to move across Asia, and into southern Europe. Soon it showed up in England.

Casebook: The Death Map

Casebook: The Death Map

This picture, illustrating a cholera epidemic, displays one common theory of cholera's transmission: **transmission by miasm**.

A miasm was a hypothetical airborne cloud, smelling terrible, and carrying the disease.

Miasm explained why cholera victims often lived in poor areas full of tanneries, butcher shops, and general dirty conditions.

It also explained why cholera did not simply spread out across the entire population, but seemed to travel, almost like the wind, settling down to devastate a neighborhood, then moving on.

Casebook: The Death Map

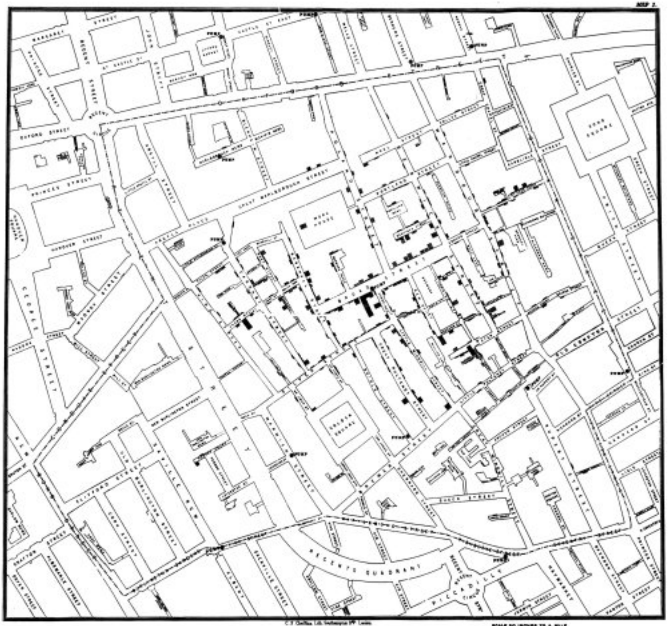
The miasm theory was accepted by doctors and public officials.

They assumed there was nothing they could do, except perhaps to drive out the bad air, using smoke or scented handkerchiefs, and to tell people not to live in low-lying areas near swamps and marshes.

Some people found the miasm theory unscientific. It didn't explain where miasms came from, or how they transmitted disease. The London epidemic was a chance to test the theory,

Was there a better theory to explain the outbreak...
and how to prevent new ones?

Golden Square: A Map

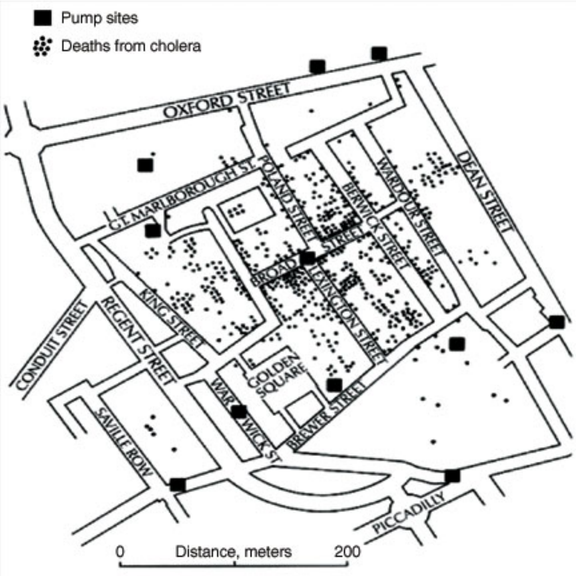


Golden Square: John Snow's Investigation

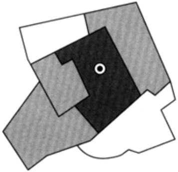
Dr John Snow suspected a particular water pump on Broad Street was the source of the cholera outbreak, but the pump water seemed relatively clean when he examined it.

- He made a map of where victims lived;
- He paced the distance to the nearest pump;
- He circled the houses closest to the Broad Street pump.
- Some victims outside the line nonetheless used the pump.

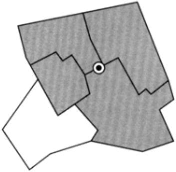
Golden Square: See the Pumps as "Suspects"



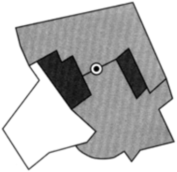
Golden Square: See the Pump Distance as Explanationl



In this aggregation of individual deaths into six areas, the greatest number is concentrated at the Broad Street pump.



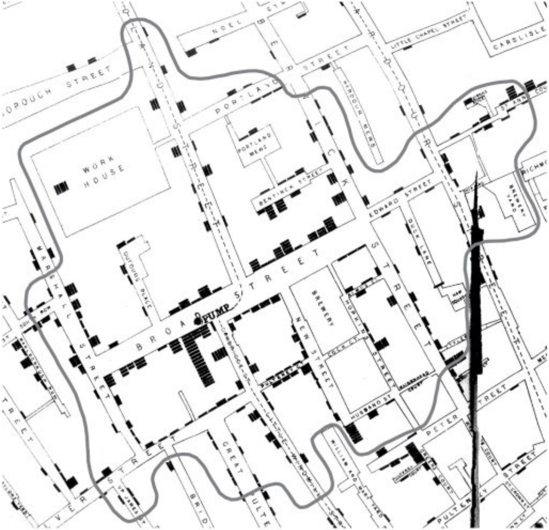
Using different geographic subdivisions, the cholera numbers are nearly the same in four of the five areas.



In this aggregation of the deaths, the two areas with the most deaths do not even include the infected pump!

[18] Mark Monmonier, *How to Lie with Maps* (Chicago, 1991), pp. 142–143.

Golden Square: The Pump and its Neighborhood



Golden Square: Conclusion

Dr Snow's map destroyed the miasm theory.

Snow's map strongly suggested that the people who died were the ones for whom the pump at Golden Square was the closest in terms of walking disance.

The miasmic cloud theory required that the miasm, purely by chance, only settled over people who used the Golden Square pump.

So Dr Snow's map showed **where** deaths occurred,
and also suggested **why**.

Disease could come from contaminated water.

Golden Square: Conclusion

There were two organizing principles in Dr Snow's map:

- the pumps
- the distance to the pumps

Dr Snow imagined every dead person walking back to the pump that was the nearest to them, and found most of them met at Golden Square.

Dr Snow was doing epidemiology, but also mathematics. His map is an interesting example of a **Voronoi diagram**.

Voronoi: Definition

Suppose that

- we have a space $\mathcal{D}$ of objects containing points or objects.
- we choose a finite set $\mathcal{G}$ of points in $\mathcal{D}$, called "generators".
- the distance between points p_1 and p_2 is $d(p_1, p_2)$.

Then the **Voronoi diagram** $\mathcal{V}(\mathcal{G}, d)$ assigns every point in $\mathcal{D}$ to the nearest generator.

In Dr Snow's map, $\mathcal{G}$ was the location of pumps,
$d(p_1, p_2)$ was the walking distance between two points.

Voronoi: Names of Things

We can abstract the ideas from Dr Snow's diagram, and think about a simple situation involving points in a plane.

We can randomly pick a small number of points as our generators, and ask how these generators, plus a distance function, organize all the points in the plane.

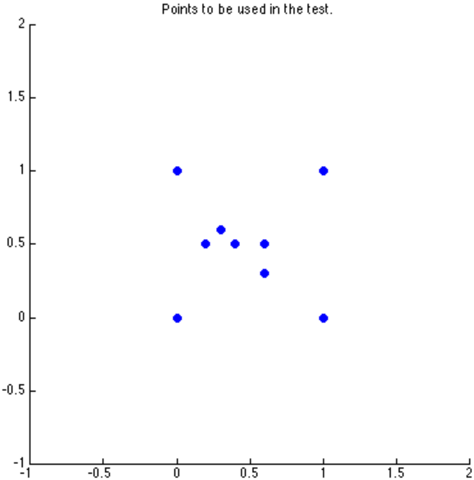
Mathematically, the distance that Dr Snow used was similar to something called the ℓ_1 distance (if streets only ran north-south or east west). We will usually prefer to use the ℓ_2 or "Euclidean" distance.

Voronoi: Names of Things

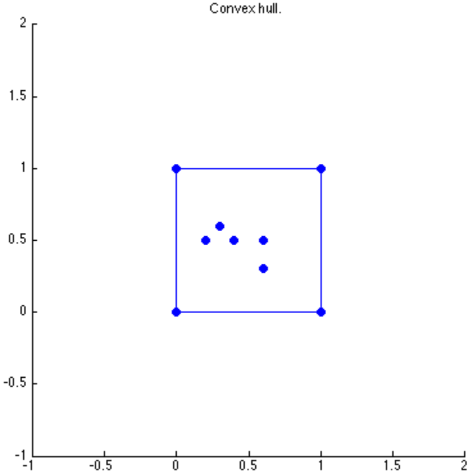
For a set of points or "generators" in the plane, there are three common geometric structures:

- **convex hull** – the smallest polygon containing the points;
- **Delaunay triangulation** – "best" connection of all points;
- **Voronoi diagram** – each generator creates a "country".

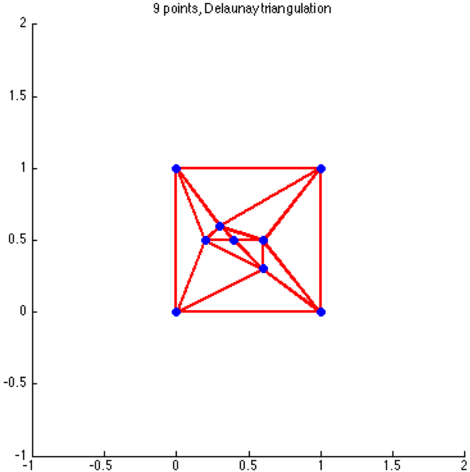
Voronoi: 9 points



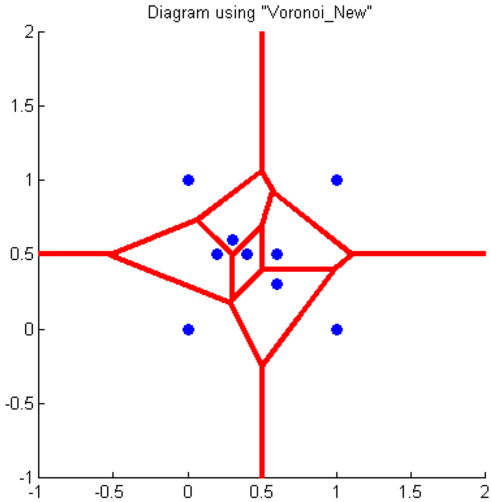
Voronoi: 9 point convex hull



Voronoi: 9 point Delaunay triangulation



Voronoi: 9 point Voronoi diagram



Voronoi: Properties of the Voronoi Regions

Assuming we are using the ℓ_2 distance function,

- All regions are convex (no indentations);
- All regions are polygonal;
- Each region is the intersection of half planes;
- The infinite regions have generators on the convex hull.

Voronoi: Properties of the Edges and Vertices

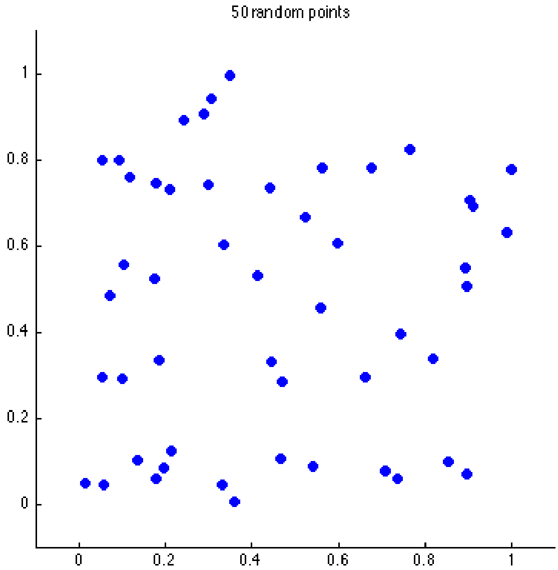
The Voronoi **edges** are the line segments of the boundary;

- Edges are perpendicular bisectors of neighbor lines;
  they contain points equidistant from 2 generators;
- If two generators share an edge, then they are connected in the Delaunay triangulation.

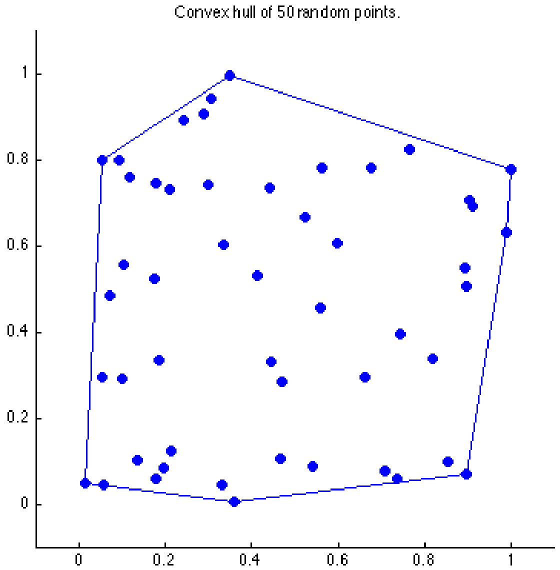
The Voronoi **vertices** are "corners" of the boundary.

- Vertices are the endpoints of 3 (possibly more) edges;
  they are equidistant from 3 (possibly more) generators;
- Each vertex is the center of a circle through 3 generators;
- Each vertex circle is empty (no other generators).

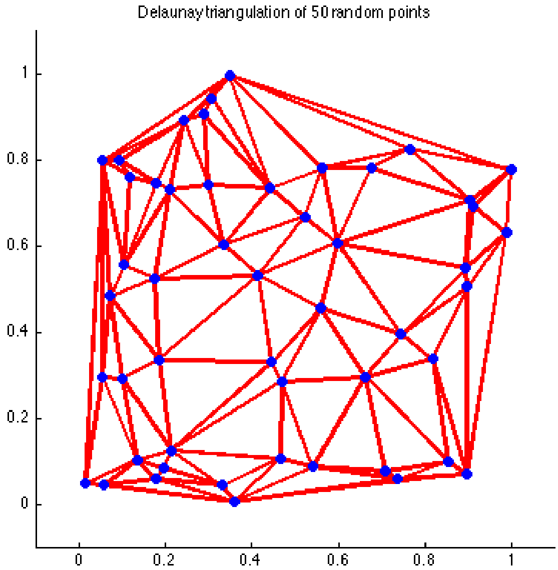
Voronoi: 50 points



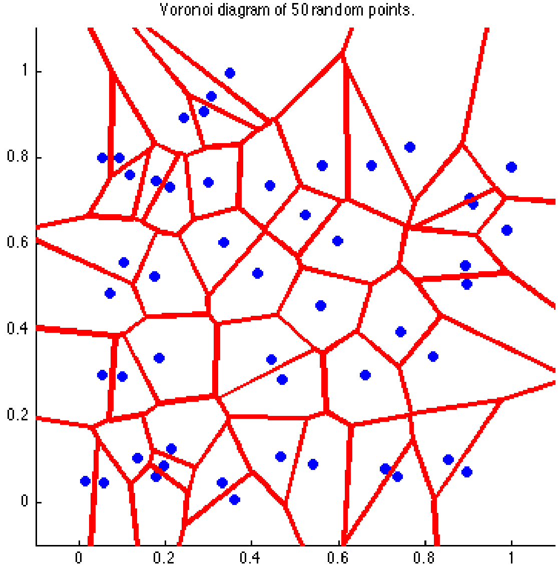
Voronoi: 50 points convex hull



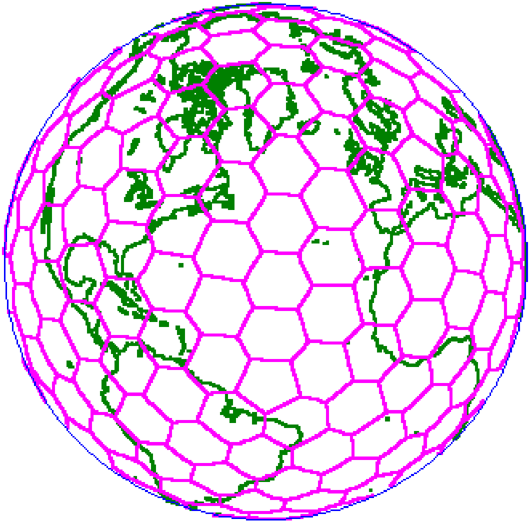
Voronoi: 50 points Delaunay triangulation



Voronoi: 50 points Voronoi diagram



Voronoi: Spheres and Other Shapes



Euler's Formula: Counting Faces, Edges, Vertices

If we only know the **number** of generators involved, that's not enough to tell us how complicated the Voronoi diagram will be.

If we think about storing the information in a computer, we would probably need to know the number of vertices and edges, **before we try to compute the diagram**, so that we have enough storage.

Since these numbers also depend on the **arrangement** of the generators, the best we can hope is that we might be able to compute an overestimate for the number of edges and vertices, so we are guaranteed to have enough storage.

Euler's Formula: Counting Faces, Edges, Vertices

Leonhard Euler developed a formula that related the number of faces, edges and vertices in a 3-dimensional polyhedron.

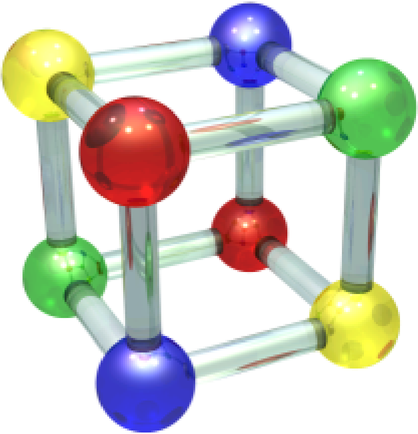
This formula can estimate the "size" of a Voronoi diagram, that is, the number of edges and faces for a given number of points.

Euler's Formula: for 3D Polyhedrons

The formula relates **F**aces, **V**ertices, and **E**dges of a polyhedron.

$$F + V = E + 2$$

$$6 + 8 = 12 + 2$$



Euler's Formula: Adapted to 2D

We can apply Euler's Formula to a 2D figure in the plane

Just imagine the surface is stretchable. Puncture the surface at one point and flatten it out.

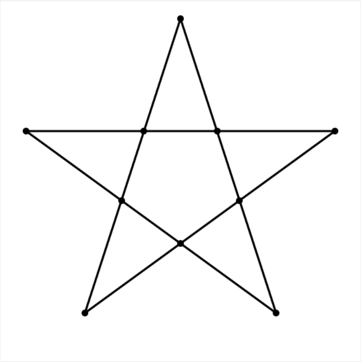
Our puncture will make one face disappear. Things will add up correctly in Euler's formula if we increase the number of faces we can see by 1, to account for the face we destroyed.

Euler's Formula: for Bounded 2D Figures

In 2D, and a bounded figure, add one infinite face.

$$(F + 1) + V = E + 2$$

$$(6 + 1) + 10 = 15 + 2$$



Euler's Formula: Unbounded 2D Figures

We can also apply Euler's Formula to a Voronoi diagram.

A Voronoi diagram is unbounded, some edges go to infinity.

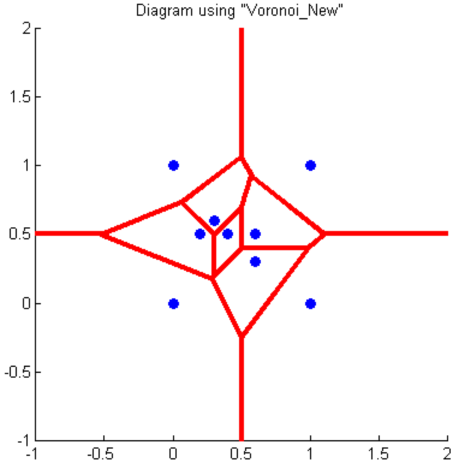
We can think of the infinite regions as just "very big" faces;
we can imagine the infinite edges all meet at a vertex at infinity.

So the original 3D polyhedron formula will work for us, as long as we add one fictitious vertex at infinity.

Euler's Formula: Unbounded 2D Figures

In 2D, and unbounded diagram, add vertex at infinity.

$$\begin{aligned} F + (V+1) &= E + 2 \\ 9 + (11+1) &= 19 + 2 \end{aligned}$$



Euler's Formula: Estimating Voronoi Size

Now we use our formula $F + V = E + 1$ to bound the maximum sizes of E and V.

First, we can figure out a limit for V if we know E.

Each edge is defined by the 2 vertices that are its endpoints, and each vertex belongs to (at least) 3 edges.

Therefore:

$$3V \leq 2E$$

Substituting for E in Euler's equation gives:

$$\begin{aligned} F + V &= E + 1 \\ 2F + 2V &= 2E + 2 \\ 2F + 2V &\geq 3V + 2 \\ 2(F-1) &\geq V \end{aligned}$$

Now we seek a bound on E:

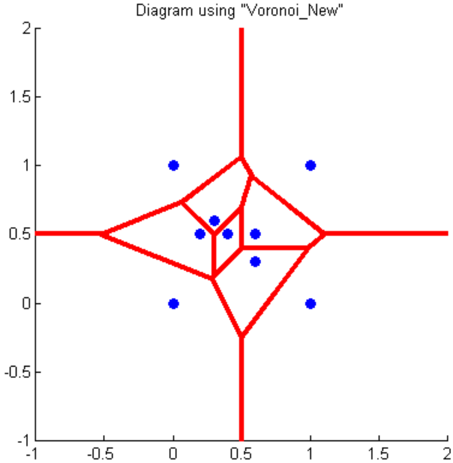
$$\begin{aligned} F + V &= E + 1 \\ F + 2(F-1) &\geq E + 1 \\ 3(F-1) &\geq E \end{aligned}$$

And thus we can bound V and E if we only know F.

Euler's Formula: Unbounded 2D Figures

For our example Voronoi diagram, $F = 9$, $V = 11$, $E = 19$, so

$$\begin{aligned} 2(F-1) &\geq V \\ 3(F-1) &\geq E \end{aligned}$$

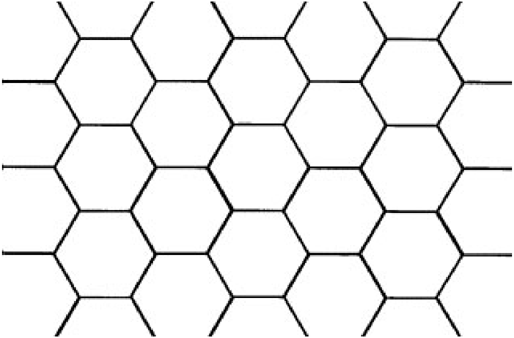


Euler's Formula: Estimating Voronoi Size

The limit on edges gives us one more interesting fact.
Each edge separates two neighboring faces.

If the number of edges is roughly its maximum $3F$, then the average number of neighbors would be about 6.

This corresponds nicely with an optimal hexagonal tiling pattern!



Voronoi Computation

In some ways, a Voronoi diagram is just a **picture**; in other ways, it is a **data structure** which stores geometric information about boundaries, edges, nearest neighbors, and so on.

How would you compute and store this information in a computer?

It's almost a bunch of polygons, but of different shapes and sizes and orders - and some polygons are "infinite".

Euler's formula lets us know in advance a limit of the number of edges and vertices.

A convex hull calculation lets us know how many infinite regions we will have.

Voronoi Computation - A Picture Calculation

If you just want a picture of the Voronoi diagram, then you will be satisfied with a list of the line segments that make up the edges.

The semi-infinite edges just have to be drawn to the edges of the picture.

For the picture option, we don't store any extra information.

MATLAB command: **voronoi (xVector, yVector)** ;
Mathematica command: **DiagramPlot[xyList]** .

Voronoi Computation - A Geometric Calculation

A more detailed calculation would need to compute lists:

- the location of the vertices;
- the pairs of vertices that form an edge;
- the sequence of edges that form the boundary of a region.
- some way to indicate whether the region is finite or infinite.

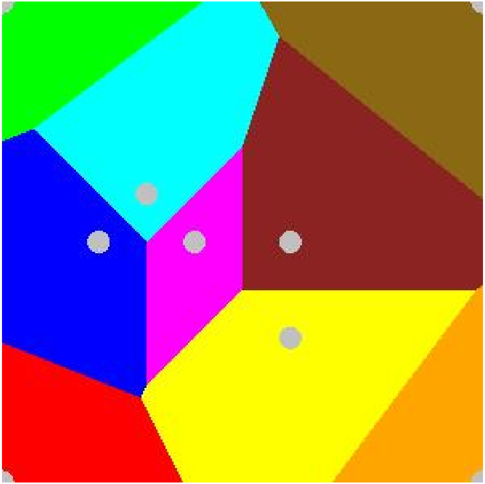
MATLAB command: **[v,c] = voronoin (xyArray)** ;
Mathematica command: **VoronoiDiagram[xyList]** .

Voronoi Computation: Algorithms

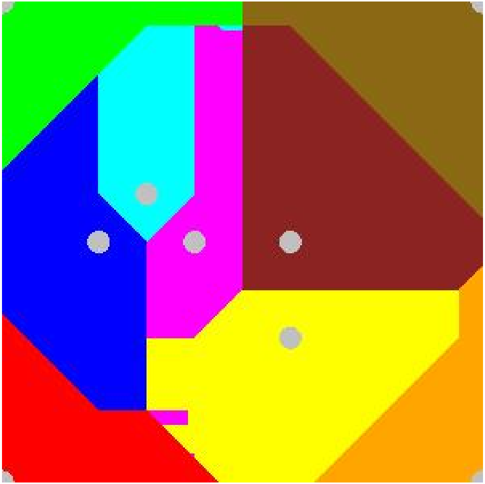
Algorithms for computing a Voronoi diagram include:

1. **The Method of Half Planes** - add one generator at a time;
2. **Byer's Method** - bisect the "neighbor" lines;
3. **Fortune's Method** - scans the data from left to right, building the diagram in stages;
4. **Grid or Pixel method** - identify nearest generator for a discrete set of points;
5. **Contour method** - draw contours of distance function.

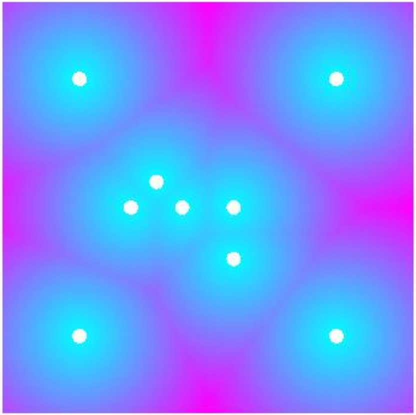
Voronoi Computation: Pixel Method (Euclidean distance)



Voronoi Computation: Pixel Method (L1 distance)



Voronoi Computation: Distance Contour Plot



Voronoi Computation: Distance Surface Plot

A side view, using a different “mountain chain”.

Centered Systems: Natural Patterns?

The Voronoi diagram seems to create some order out of scattered points.

We might compare the Giant's Causeway, or the spots on leopards or giraffes, or the "spitting fish"" nests.

However, the generators are not "centered" and the shape and size of the regions seems to vary more than in some natural patterns. Can we explain or control this?

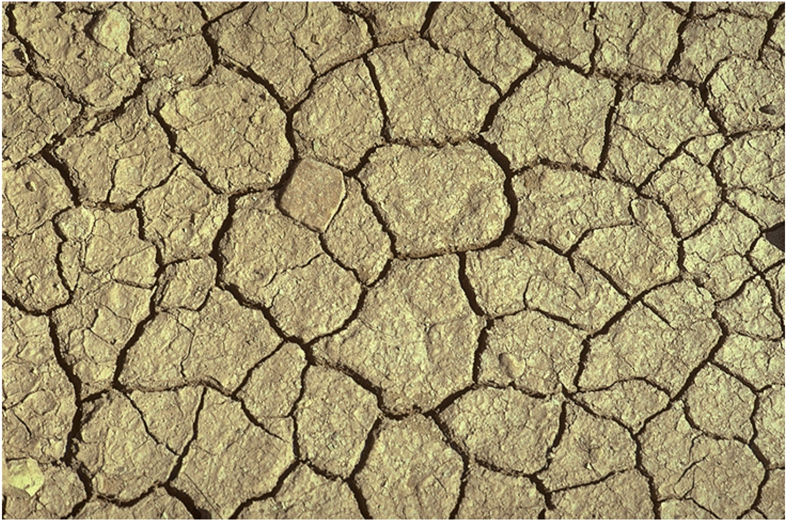
Centered Systems: Giraffe Patches

Centered Systems: More Realism?

Physical examples differ from our mathematics.

- Instead of the infinite plane, we have a finite area or surface.
- The regions are roughly the same size.
- The regions are roughly the same shape.

Voronoi: Drying Mud



Centered Systems: Center the Generators

Suppose, once we compute a Voronoi region, we allow the generator to move towards the center.

This is like relocating the capital city of a state to the center of the state (this happened in Iowa!)

One effect: probably reduces more distances to the generator.

Another effect: some points may suddenly "belong" to a different generator.

So what happens if we recompute the Voronoi diagram?

Centered Systems: Algorithm

To carry out this process, we simply replace each generator point by the centroid (C_x, C_y) of the region R.

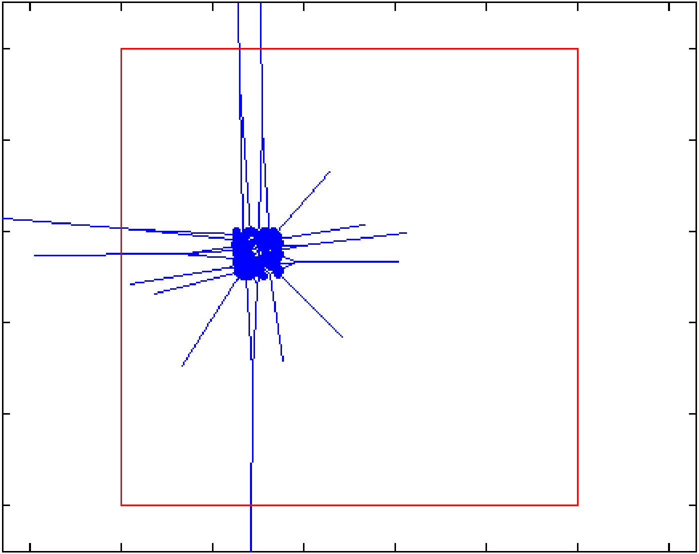
If the region is a polygon, then we can use geometry to figure out the centroid.

If the region is more complicated, we must use calculus:

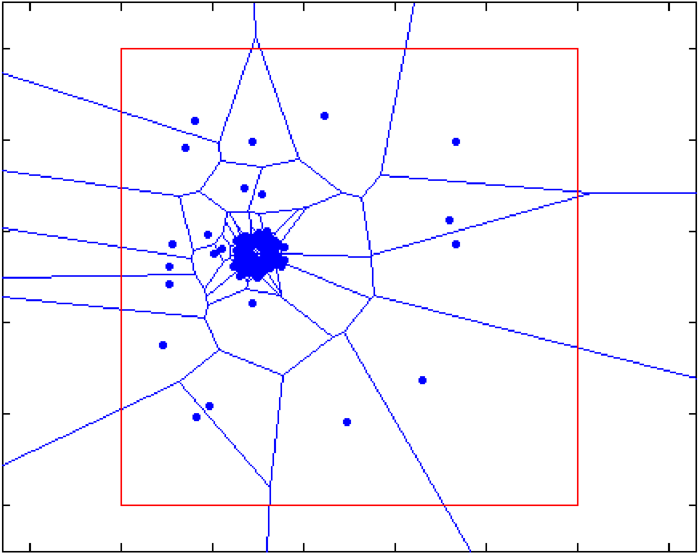
$$C_x = \frac{\int_R x \, dx \, dy}{\int_R dx \, dy}, C_y = \frac{\int_R y \, dx \, dy}{\int_R dx \, dy}$$

(but if **that's** *too hard, we can use a sampling trick!)*

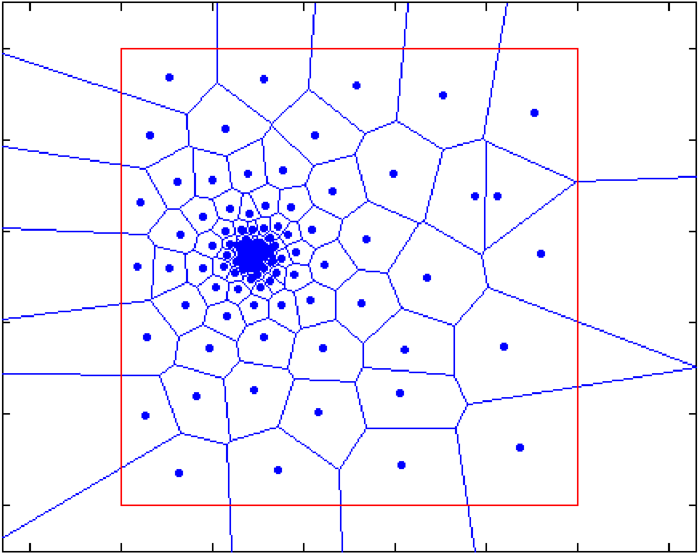
Centered Systems: Step 001



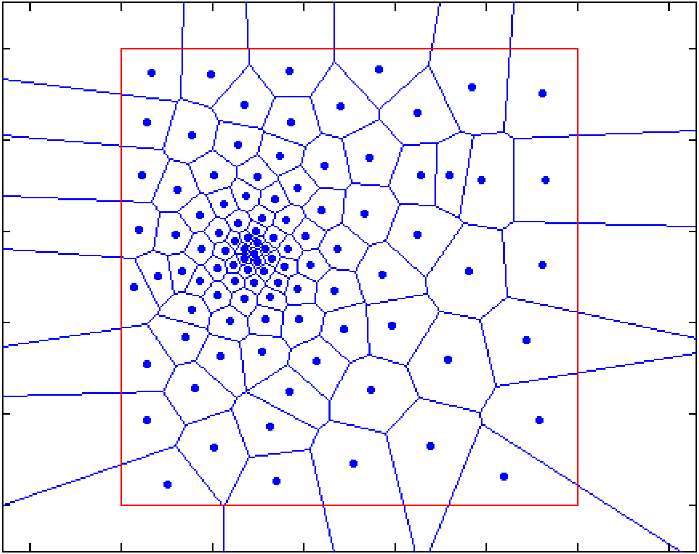
Centered Systems: Step 002



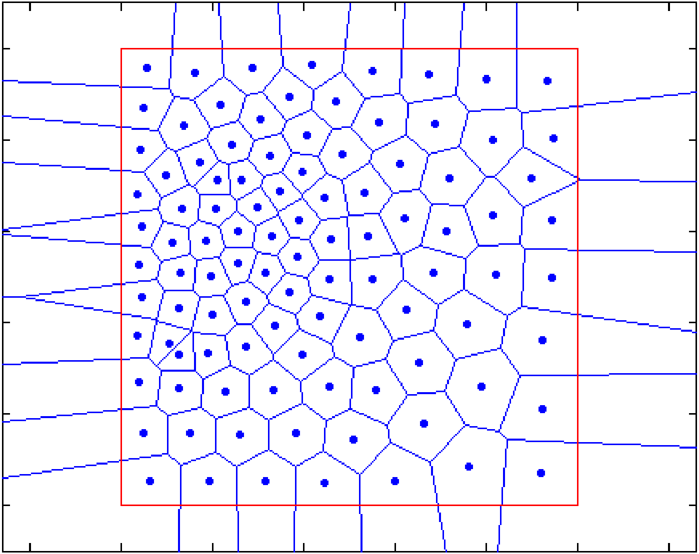
Centered Systems: Step 010



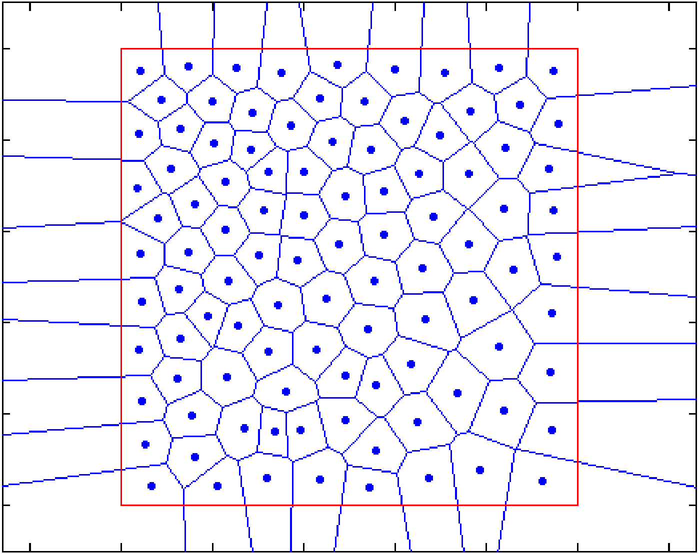
Centered Systems: Step 020



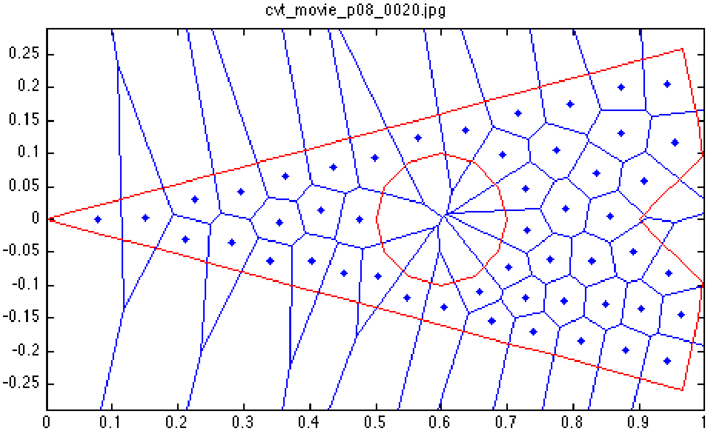
Centered Systems: Step 040



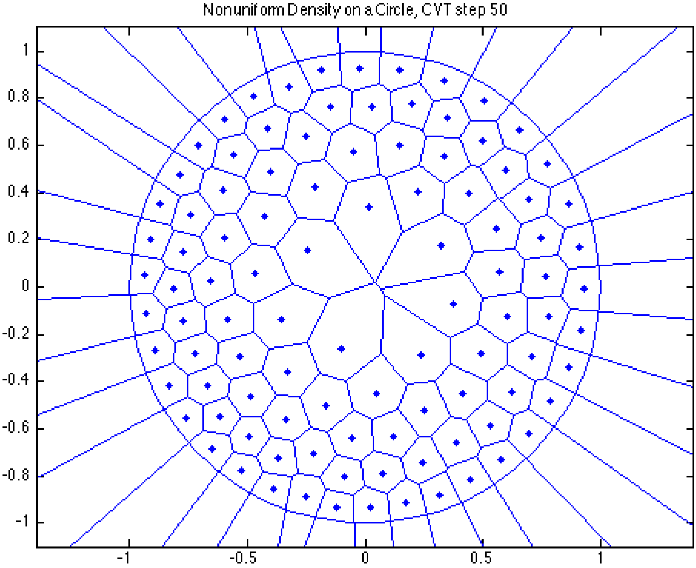
Centered Systems: Step 080



Centered Systems: Regions with Holes and Corners



Centered Systems: Nonuniform Density, Curved Region

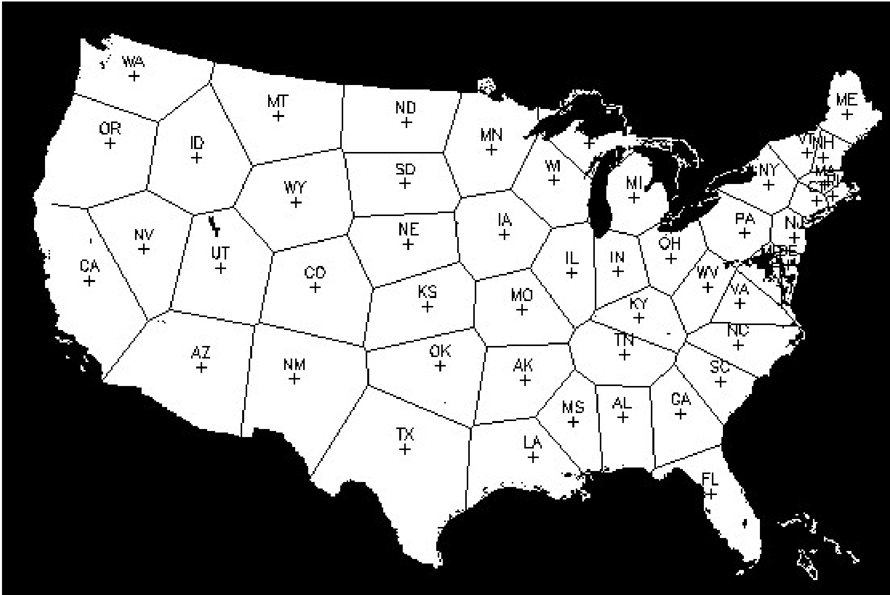


Conclusion: Voronoi Diagrams

The Voronoi diagram is a mathematical tool for analyzing geometric processes with many local centers.

- disease outbreaks
- patterns on animal skin or sea shells
- territories of ant colonies, prairie dogs
- how can an attacking plane best avoid all enemy bases?
- if we have already chosen capitals for the states, where should state boundaries be?

Conclusion: The Voronoi States of America



Conclusion: Centroidal Voronoi Diagrams

The Centroidal Voronoi diagram helps us to simulate, model, and sometimes to understand cell structures in which the generator tends to be in the center, or to move there:

- how do fish adjust their nesting sites?
- how do rotating cells form in a cooling liquid?
- placement of mailboxes
- arrangement of sonar receivers on ocean floor
- what arrangement of bases makes it hardest for planes to attack?

Conclusion: Computational Geometry

Computational Geometry: geometric algorithms for:

- distances, areas, volumes, angles of complicated shapes;
- nearest neighbors;
- shortest paths, and best arrangements of points;
- hulls, triangulations, Voronoi diagrams;
- meshes, decomposition of shapes into triangles or tetrahedrons;
- motion of an object through a field of obstacles.

CGAL is a computational geometry library;
Mathematica includes a computational geometry package;
MATLAB has many computational geometry routines.

Conclusion: The John Snow Memorial