Geometría Computacional

Alex Damián Bonivardo

17 de diciembre de 2017

Índice general

Capítulo 1

¿Qué es la Geometría Computacional?

La Geometría Computacional es una rama de las ciencias computacionales que se encarga del diseño y análisis sistemático de algoritmos y estructuras de datos necesarios para la solución eficiente de problemas que implican como entrada y salida objetos geométricos. Sus orígenes nos pueden remontar en siglos (hay quien dice que el primer algoritmo de Geometría Computacional nace cuando una serie de pasos correctos no ambiguos y con un final resuelven un problema geométrico, el precursor: Euclides), pero a principios de 1970 Michael Shamos comienza un estudio sistematizado de problemas con la idea de crear una nueva disciplina de las ciencias de computación, a la cual llamó Geometría Computacional [1]. Partiendo de la abstracción de problemas de otras áreas, se trata de desarrollar herramientas y técnicas para resolver problemas de naturaleza, principalmente, geométrica, con especial énfasis en el diseño eficiente de algoritmos y estructura de datos.

El principal impulso para el desarrollo de la geometría computacional se lo dio el avance de la computación gráfica y el diseño asistido por ordenador (CAD/CAM), que hacen uso intensivo de las técnicas de esta disciplina. Otras aplicaciones importantes incluyen la robótica (planificación de movimientos y problemas de visualización), los sistemas de información geográfica (localización y búsqueda geométrica, planificación de rutas), diseño de circuitos integrados (diseño geométrico y verifición de CI), ingeniería asistida por computadora (programación de máquinas controladas numéricamente)[2].

En esta monografía vamos a introducirnos especialmente en una aplicación de la Geometría Computacional, los **diagramas de Voronoi**.

[1] Martínez Rodríguez, 2015, p. 1

[2] Extraído de www.wikipedia.org

Capítulo 2

Diagramas de Voronoi

Los diagramas de Voronoi se encuentran entre las más importantes estructuras en geometría computacional. Un diagrama de Voronoi codifica la información de proximidad entre elementos.

Sea P = $\{p_1, p_2, \ldots, p_n\}$ un conjunto de n puntos distintos del plano. Definimos el **diagrama de Voronoi** del conjunto P como una subdivisión del plano en n celdas, una por cada punto de P, con la propiedad de que un punto 'q' pertenece a la celda correspondiente al punto 'p' si y solo si la distancia euclidea de 'q' a 'p' es menor que a cualquier otro punto de P. Observemos un diagrama de Voronoi generado por 8 puntos.

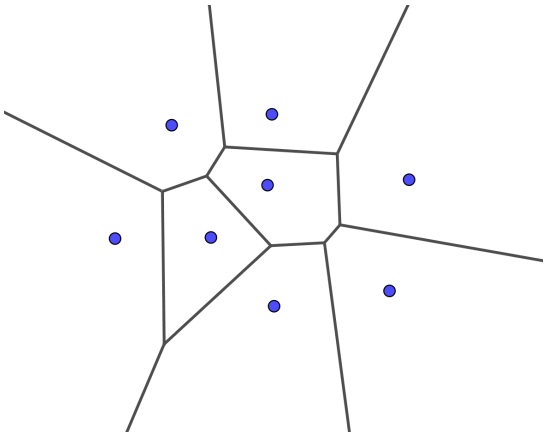


Figura 2.1: Diagrama de Voronoi

Ante la presencia de un nuevo punto, por ejemplo, el rojo en la siguiente figura, es inmediato reconocer cuál, de los 8 originales, es el más cercano a él, el punto generador de la región en la que ha caído.

Observemos la siguiente imagen que da cuenta de ello:

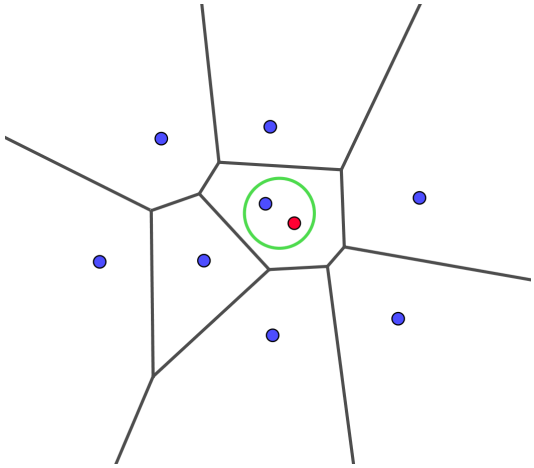


Figura 2.2: Diagrama de Voronoi - Punto Generador

Esta propiedad que poseen los diagramas de Voronoi tienen una importante aplicación en la vida cotidiana.

Supongamos que nos hemos quedado sin nafta para nuestro auto y deseamos chequear, a través de un diagrama de Voronoi, en que Región de Voronoi nos encontramos y así poder saber que Estación de Servicio es la más cercana. Recordemos que, al trazar estos diagramas, dividimos el plano en regiones. Dichas regiones son generadas por los Puntos P_i y P_j que, en nuestra situación son las Estaciones de Servicio.

Antes de continuar con nuestra situación, veamos detenidamente como construir los diagramas de Voronoi.

2.1. ¿Como se construyen los polígonos de Voronoi?

Se crean al unir los puntos entre sí, trazando las mediatrices de los segmentos de unión. Las intersecciones de estas mediatrices determinan una serie de polígonos en un espacio bidimensional alrededor de un conjunto de puntos de control, de manera que el perímetro de los polígonos generados sea equidistante a los puntos vecinos y designando su área de influencia.

Dados 2 puntos, P_i y P_j, en un plano T, la perpendicular al segmento P_iP_j en su punto medio divide el plano en dos regiones V_i y V_j; la región V_i contiene todos y solo los puntos cuya distancia a P_i es menor que a P_j y la región V_j contiene el resto.

Observemos la siguiente imagen

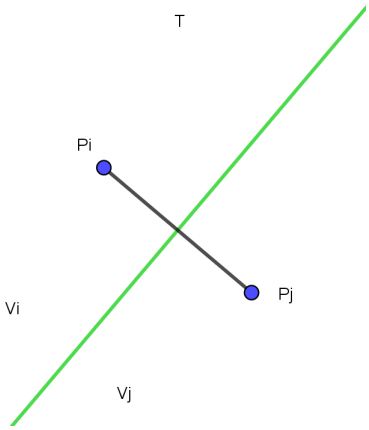


Figura 2.3: Regiones de Voronoi de dos puntos

Si agregamos otro punto para construir nuestro diagrama, el vértice en el cual tres regiones de Voronoi V_i, V_j, y V_k se intersectan, es llamado un vértice de Voronoi y tiene la propiedad de ser equidistante de los puntos P_i, P_j y P_k. Por lo tanto, es el centro del círculo que pasa por estos puntos. Además este círculo no contiene otros puntos generadores en su interior.

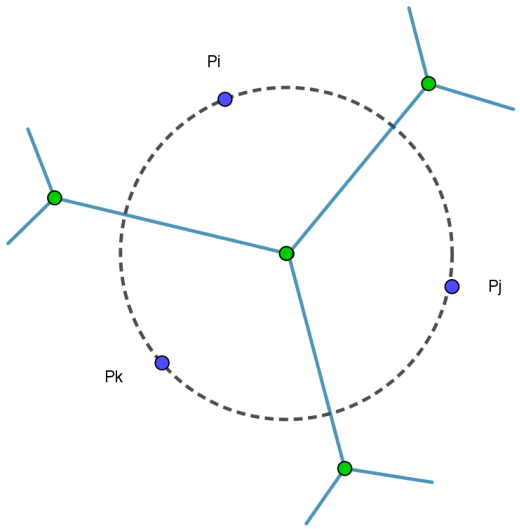


Figura 2.4: Regiones de Voronoi de tres puntos

El concepto se extiende a múltiples puntos P_n de forma que cada uno de ellos se asocia a una región de Voronoi, V_n, que contiene todos los puntos del

plano más próximos. Si se aplica a un dominio cerrado se genera un conjunto de polígonos convexos que divide el plano.

Este método de construcción de diagramas de Voronoi recibe el nombre de **Intersección de semiplanos**. La construcción de **n** semiplanos puede realizarse en un tiempo *O(n Log n)*; y hacer esto para cada generador podría costar un tiempo total de $O(n^2\ Log\ n)$[1].

Este algoritmo tiene algunas desventajas. En primera instancia, el cálculo explícito de los semi-planos y su intersección puede provocar problemas de precisión en la computadora generado, evidentemente, una versión incorrecta de ***Vor(P)***. El segundo inconveniente involucra que no se produce información inmediata y que se pueda aprovechar acerca del vecindario de cada sitio o punto generador.

Ahora retomemos nuestra situación hipotética anterior. ¿Cómo quedarían delimitadas las regiones de Voronoi en torno a las estaciones de Servicio de la zona?

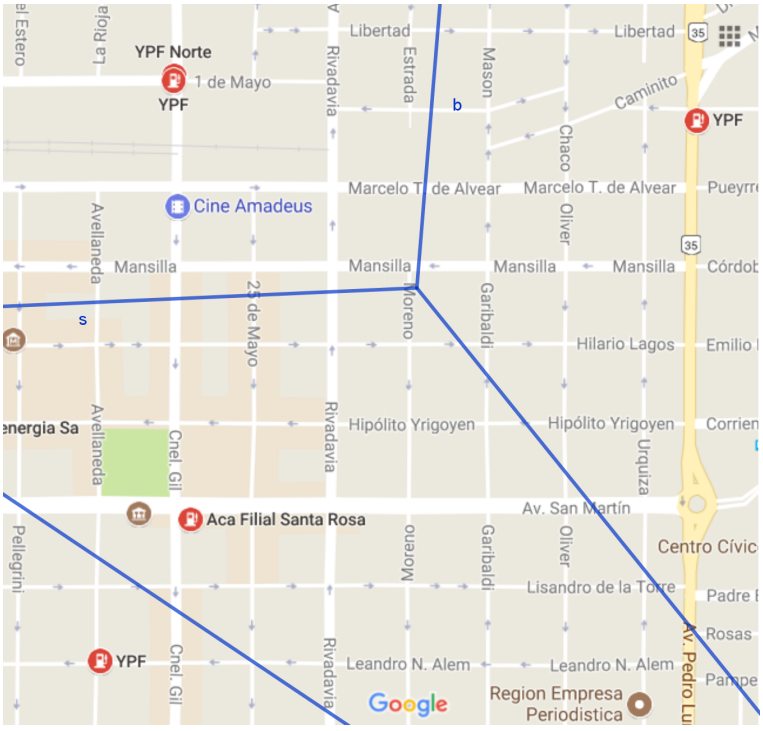


Figura 2.5: Regiones de Voronoi de las Estaciones de Servicio

El anterior diagrama de Voronoi se obtuvo con el software Geogebra,

[1]Complejidad computacional (peor caso, caso promedio y mejor caso) en términos de n, el tamaño de la lista o arreglo. Para esto se usa el concepto de orden de una función y se utiliza la notación $O(n)$. El mejor comportamiento para ordenar es *O(n Log n)*.

tomando una porción del mapa donde habitualmente se encuentran las personas involucradas en la situación-problema.

Una vez delimitadas las regiones de Voronoi en el mapa podremos saber cuál es la Estación de Servicio más Cercana. Si estamos ubicados en la esquina de Mansilla y Rivadavia, sin dudas la más cerca es YPF Norte, pues nos encontramos en la región delimitada por la misma.

2.2. Algunos métodos más

El gran número de aplicaciones del diagrama de Voronoi ha espoleado a numerosos investigadores a desarrollar algoritmos para computarlo. A continuación mencionaremos algunos métodos más.

2.2.1. Algoritmo incremental

Este algoritmo se basa en, ya construido el diagrama para k puntos, construir el diagrama para $k+1$ puntos o sitios. La idea es generar celdas de forma incremental. En lugar de hallar la intersección de cada región o celda con respecto a todas las demás, se llevan a cabo los respectivos cálculos sólo con aquellas que son afectadas por un nuevo punto que se inserte.

Observemos la siguiente imagen donde se construye el diagrama por el algoritmo incremental.

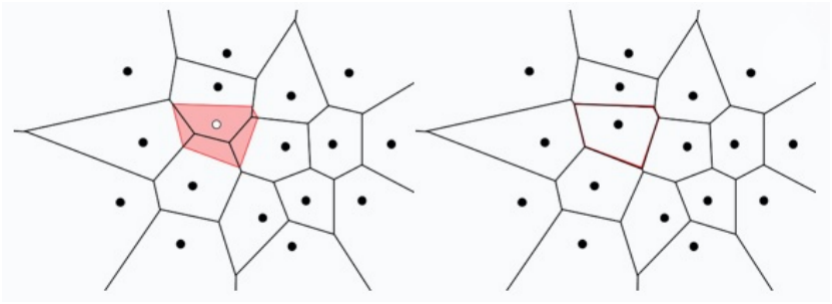


Figura 2.6: Diagrama de Voronoi - Algoritmo incremental

Este algoritmo emplea un tiempo de $O(n)$ en la inserción de cada nuevo punto, con una complejidad total de $O(n^2)$. A pesar de su complejidad cuadrática, este ha sido el método más popular para construir diagramas de Voronoi.

2.2.2. Algoritmo Divide y Vencerás

El objetivo de este algoritmo es dividir un problema en n subproblemas y, una vez halladas las n subsoluciones, fusionarlas para encontrar la solución general.

Consiste en dividir, mediante una línea vertical L, al conjunto de puntos P en dos subconjuntos P_1 y P_2, con aproximadamente el mismo tamaño, de los que se debe encontrar su diagrama de Voronoi independiente. Finalmente, ambos diagramas deben ser unidos para obtener el diagrama de Voronoi de P.

Sea $\boldsymbol{M}$ el subgrafo de Voronoi del conjunto P, tal que los lados de M están determinados por parejas de puntos $\{p_i, p_j\}$ tales que $p_i \in V(P_1)$ y $p_j \in V(P_2)$; M es el grafo frontera.

Veamos en la siguiente imagen como construir un diagrama de Voronoi mediante el algoritmo Divide y Vencerás.

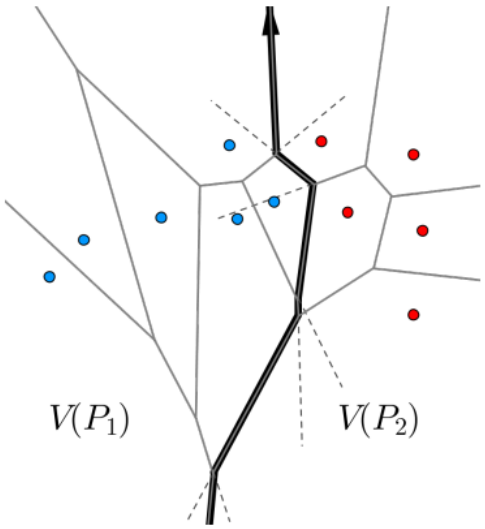


Figura 2.7: Diagrama de Voronoi - Algoritmo Divide y Vencerás

Los diagramas de Voronoi construidos con la utilización del algoritmo Divide y Vencerás requieren un tiempo *O(n Log n)*. Esta complejidad es asintóticamente óptima, pero el algoritmo resulta bastante difícil de implementar.

2.2.3. Algoritmo de Fortune (barrido del plano)

Hasta mediados de los ochenta, la mayoría de las implementaciones para computar el diagrama de Voronoi usaban el algoritmo incremental cuadrático, admitiendo su mayor lentitud para evitar la complejidad del método divide y vencerás. En 1985 Steven Fortune inventó un inteligente algoritmo de barrido plano.

Fortune consideró que, con el método tradicional de barrido del plano, no alcanzaba y por ello utilizó un método 'distorsionado'. El problema al barrer el plano con una línea es que la parte del diagrama que cae por encima de la línea puede modificarse por puntos que están debajo. Para evitar este problema se utiliza la *línea de playa* que es el conjunto de puntos que equidistan de la *línea de barrido* y algún p_i por encima de línea de barrido. Para que los sitios que se encuentran por debajo de la línea de barrido no modifiquen el diagrama que se encuentra por encima de ella, solo mantendremos la parte del diagrama que no puede ser afectada por algún sitio que esté debajo de la línea de barrido.

Veamos, mediante la siguiente imagen, como se comporta el algoritmo de Fortune.

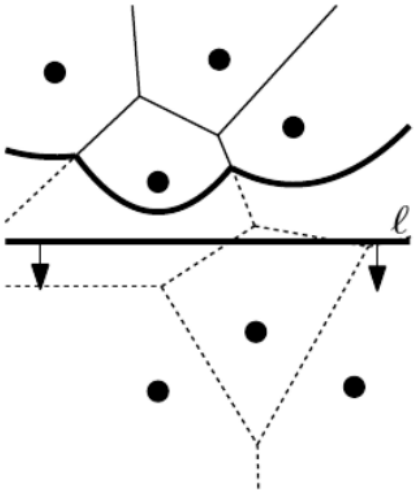


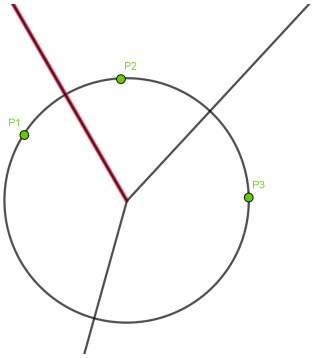
Figura 2.8: Diagrama de Voronoi - Algoritmo de Fortune

El algoritmo de Fortune es tan simple de aplicar como el algoritmo incremental pero posee una complejidad *O(n Log n)*.

2.3. Algunas propiedades del Diagrama de Voronoi

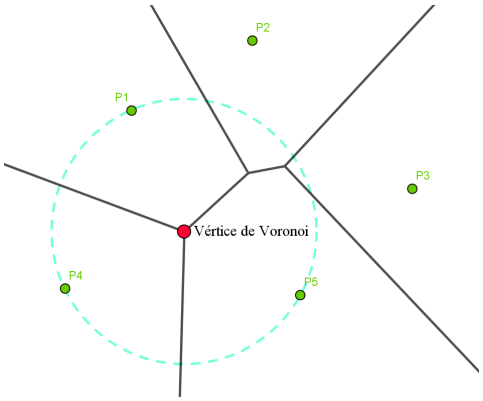
Los diagramas de Voronoi poseen algunas propiedades que mencionaremos a continuación:

1. Dos puntos p_i y p_j son vecinos si comparten una arista. Una arista es la mediatriz perpendicular del segmento $p_i p_j$.



   En nuestra situación de las Estaciones de Servicio, el punto **ACA Filial Santa Rosa** es vecino de los tres puntos restantes, ya que comparte aristas con todos.

2. Un vértice de Voronoi es un punto equidistante a tres generadores (si lo es a más de tres hablamos de casos degenerados) y es la intersección de tres aristas.

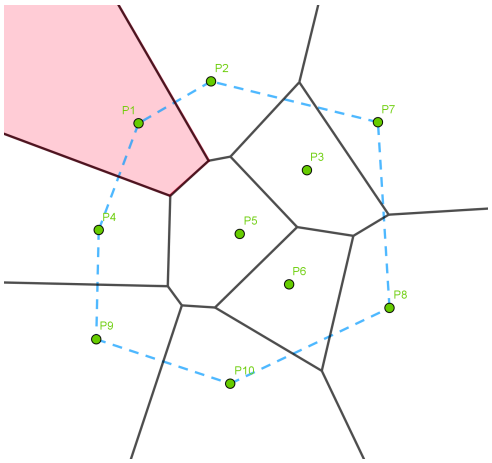


   El vértice de Voronoi que se encuentra en la calle Moreno casi Mansilla es generado por **ACA Filial Santa Rosa**, **YPF** e **YPF Norte**.

3. Dentro del círculo con centro en un vértice de Voronoi y que pasa por 3 puntos generadores no puede existir ningún otro punto generador.

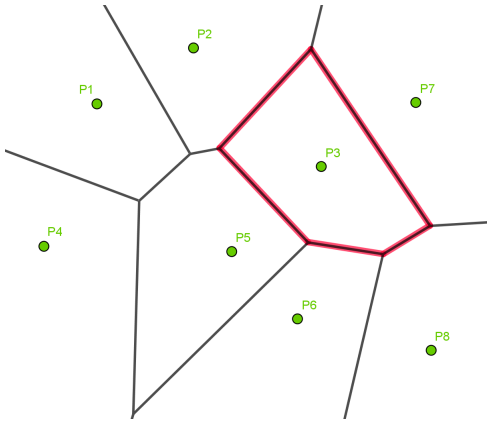
4. Una región de Voronoi es no acotada si su punto generador pertenece a la envolvente convexa de la nube de puntos[2].

   Por ejemplo, la región generada por el punto P_1 es no acotada pues su punto pertenece a la envolvente convexa de nube de puntos.



   En la situación de las Estaciones de Servicio, las regiones de Voronoi generadas por ambas **YPF** e **YPF Norte** son no acotadas.

5. Una región de Voronoi es un polígono convexo o es una región no acotada.



   En la situación de las Estaciones de Servicio, la única región que es un polígono convexo es la generada por **ACA Filial Santa Rosa**.

6. El diagrama de Voronoi de un conjunto de puntos es único.

[2]Envolvente Convexa: polígono convexo que contiene a todos los puntos del diagrama. Nube de puntos: aquellos puntos que forman la envolvente convexa

Capítulo 3

Diagramas de Voronoi en Mathematica

Como sucede con otras temáticas, y los Diagramas de Voronoi no son la excepción, siempre existen software que facilitan el trabajo de calculo y/o construcciones geométricas.

En Mathematica[1], también se pueden evidenciar los Diagramas de Voronoi de variadas maneras. Veamos, en este capítulo, algunas de ellas.

Una opción muy sencilla, y con pocos comandos, de generar estos diagramas es utilizar primero la instrucción `RandomReal` para generar números reales aleatoriamente y luego `VoronoiMesh`.

In[1]:= `Puntos=RandomReal[{-20, 1}, {25, 2}];`
In[2]:= `VoronoiMesh[Puntos]`
Out[3]:=

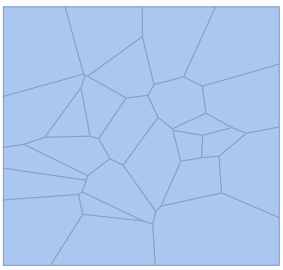


Figura 3.1: Diagrama de Voronoi con Mathematica

[1]Programa utilizado en áreas científicas, de ingeniería, matemática y áreas computacionales. Originalmente fue concebido por Stephen Wolfram. Comúnmente considerado como un sistema de álgebra computacional, también es un poderoso lenguaje de programación de propósito general

El comando `RandomReal`, en este caso, generará 25 números aleatorios, de a pares, entre -20 y 1. Por su parte, `VoronoiMesh` producirá el diagrama de acuerdo a los puntos generados anteriormente. Y, si queremos mostrar también los puntos generadores de cada región utilizamos el comando `Show` que sirve para visualizar varias gráficas juntas, en este caso incorpora con el comando `Graphics` al conjunto de puntos.

In[4]:= `Show[VoronoiMesh[Puntos],Graphics[{Red, Point[Puntos]}]]`
Out[5]:=

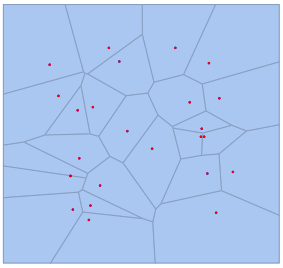


Figura 3.2: Diagrama de Voronoi con Mathematica

Otra forma de generar un Diagrama de Voronoi es a través de la siguiente secuencia de comandos:

In[6]:= `n=250;`
In[7]:= `g=Image[Map[Boole[# >0.001]&, RandomReal[{0,1},{n,n}],{2}]];`
Out[8]:= `i=DistanceTransform[g] //ImageAdjust //ImageData;`
Out[8]:= `mask=Image[WatershedComponents[Image[i]]]`

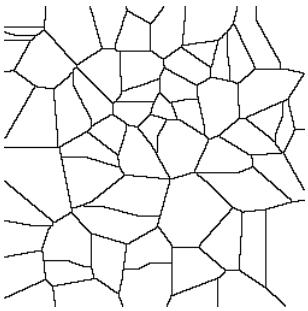


Figura 3.3: Diagrama de Voronoi con Mathematica

En la anterior lista de comandos, un poco más complejos y variados, se trata de generar un diagrama a través de una serie de rásters o píxeles,

analizando cada uno de ellos y, a través de respuestas afirmativas o no, ir generando el diagrama correspondiente.

No es el objetivo de este capítulo introducirnos tanto en el manejo de Mathematica, pero si exhibir algunos resultados que nos brinda este software. Diversas son las maneras de producir diagramas de Voronoi recurriendo a los comandos que nos brinda el programa. A continuación veamos algunos resultados obtenidos con Mathematica que se pueden visualizar con los códigos que encontraremos posteriormente en el Anexo.

- Utilizando gama de colores para identificar las celdas según su área.

Figura 3.4: Diagrama de Voronoi con Mathematica - Gama de colores

- También podemos realizar diagramas de Voronoi sobre una esfera.

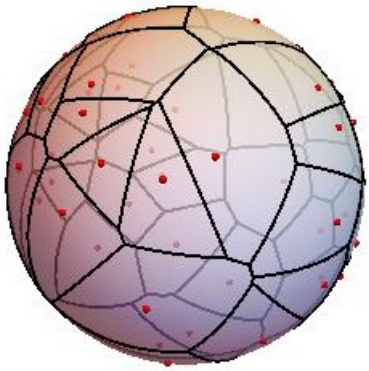


Figura 3.5: Diagrama de Voronoi con Mathematica - Esfera

- O realizar diagramas de Voronoi sobre cualquier polígono.

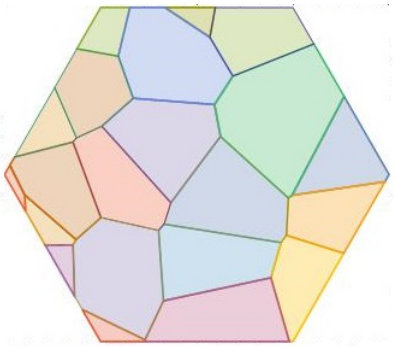


Figura 3.6: Diagrama de Voronoi con Mathematica - Polígono

Capítulo 4

Aplicaciones analizadas

Así como construimos un diagrama de Voronoi, a partir de una simple y cotidiana situación con las Estaciones de Servicio, los mismos se encuentran presentes en numerables situaciones de la vida cotidiana, en la naturaleza y en el mundo. Veamos a continuación algunas de ellas.

4.1. Brote de Cólera en Londres

"En los años 1853 y 1854, Londres enfrentó una tercera epidemia de cólera. Por aquél entonces los habitantes de ciertos distritos del sur de la ciudad extraían agua del Támesis o la obtenían de bombas de uso público. Snow sostenía que la gente, al beber agua contaminada extraída del río, ingería materia insana y de esta manera contraía el cólera.

A principios de septiembre de 1854, el sector Golden Square fue escenario de un brote epidémico de gran intensidad. Snow era vecino del área y sabía que la mayoría extraía agua de una bomba en la calle Broad Street, por ello registró las direcciones de 83 personas fallecidas en el área. Se constató que la mayoría de los personas fallecidas se abastecían de la bomba mencionada, dado que calculó la distancia entre la residencia de cada víctima y la bomba más cercana.

Para saber si en verdad la bomba de Broad Street es la más cercana al domicilio de las muertes por cólera se dibujó el diagrama de Voronoi tomando como puntos generadores cada bomba en el mapa. Los puntos representan los domicilios de las victimas de cólera y las cruces representan las bombas, destacando con una cruz roja la bomba de Broad Street"[1].

[1]Martínez Rodríguez, 2015, p. 85

Figura 4.1: Diagrama de Voronoi del análisis de brote de cólera en Londres

4.2. La jirafa

"Las jirafas tienen un sistema de vasos sanguíneos que pasan por debajo de las manchas rojizas de sus pelajes. Un vaso sanguíneo más grande rodea cada mancha y envía vasos más pequeños al centro de la mancha.

En el centro de cada mancha, los vasos liberan calor desde el cuerpo, regulando la temperatura corporal del animal. Sus manchas sirven para deshacerse del calor ya que ellas no pueden sudar, el centro de las manchas es por donde emana la temperatura más caliente del cuerpo.

Para asegurarnos de que la piel de una jirafa sigue un diseño de diagrama de Voronoi, se trataron de ubicar los puntos generadores a partir del color más obscuro dentro de las regiones; es decir, se intentaron hallar los vasos que liberan el calor"[2].

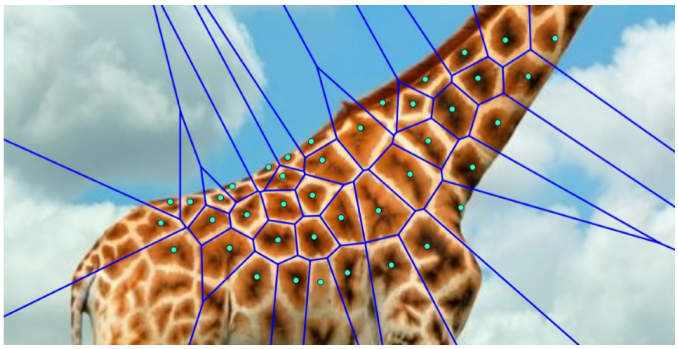


Figura 4.2: Aproximación del diagrama de Voronoi en la piel de una Jirafa

[2]Martínez Rodríguez, 2015, p. 90

4.3. Epidemia de Diarrea en Santa Rosa

"La directora de Epidemiología de La Pampa advirtió que la población está en riesgo a partir de los desbordes cloacales que se incrementaron en la capital Santa Rosa, ciudad que se encuentra en emergencia sanitaria con 4.309 casos de diarrea.

Bertone destacó que la población está en riesgo producto de estos derrames cloacales y recordó que el ambiente es un determinante para la enfermedad de las personas.

Veamos el siguiente mapa de una parte de la Ciudad de Santa Rosa donde se puede observar una aproximación del diagrama de Voronoi teniendo en cuenta como puntos generadores los centros de los barrios. Los barrios más afectados del mapa son Butaló I y II, Villa Las Camelias y Villa Parque"[3].

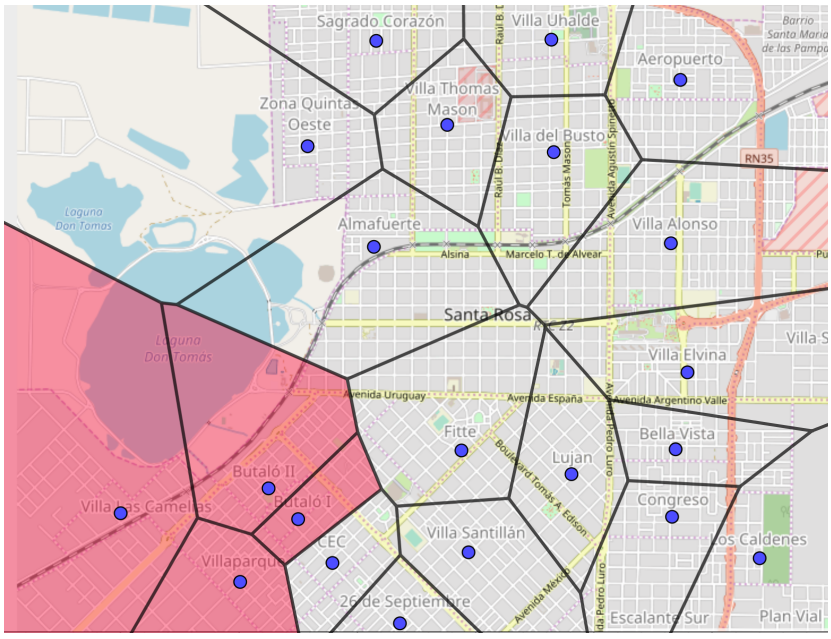


Figura 4.3: Aproximación del diagrama de Voronoi en barrios de Santa Rosa

Gracias al diagrama de Voronoi trazado sobre el mapa de los barrios de Santa Rosa, podemos observar cuáles son las zonas más afectadas refiriéndonos a las proximidades de los centros de esos barrios. Recordemos que hemos tomado como puntos generadores los centros de cada barrio, entonces calificarán como zonas más afectadas (según la proximidad) aquellas que se encuentren en las regiones de Voronoi de Butaló I y II, Villa Las Camelias y Villa Parque.

[3]Extraído de: Aire de Santa Fe DIGITAL *La Pampa en alerta por epidemia de diarrea*

Apéndice A

Anexo

En este anexo veremos los códigos que fueron utilizados para generar los diagramas de Voronoi visualizados en este trabajo.

- Diagrama de Voronoi - Gama de colores

```
pts = RandomReal[{-1, 1}, {25, 2}];
v = VoronoiMesh[pts];
mp = MeshPrimitives[v, 2];
a = Rescale[Area /@ mp];
Legended[Graphics[
  MapThread[{ColorData["Rainbow"][#1], #2} &, {a, mp}]],
 BarLegend["Rainbow"]]
```

- Diagrama de Voronoi - Esfera

```
(* http://mathematica.stackexchange.com/a/10994 *)
arc[center_?VectorQ, {start_?VectorQ, end_?VectorQ}] := Module[{ang, co, r},
    ang = VectorAngle[start - center, end - center];
    co = Cos[ang/2]; r = EuclideanDistance[center, start];
    BSplineCurve[{start, center + r/co Normalize[(start + end)/2 - center], end},
                 SplineDegree -> 2, SplineKnots -> {0, 0, 0, 1, 1, 1},
                 SplineWeights -> {1, co, 1}]]

BlockRandom[SeedRandom[0, Method -> "MersenneTwister"]; (* for reproducibility *)
            points = {2 π #1, ArcCos[2 #2 - 1]} & @@@ RandomReal[1, {50, 2}];]

sp = Append[Sin[#2] Through[{Cos, Sin}[#1]], Cos[#2]] & @@@ points;
ch = ConvexHullMesh[sp];
verts = MeshCoordinates[ch]; polys = First /@ MeshCells[ch, 2];

voro = Normalize[Cross[verts[[#2]] - verts[[#1]], verts[[#3]] - verts[[#1]]]] & @@@ polys;
edges = arc[{0, 0, 0}, voro[[##]]] & /@
        Select[Subsets[Range[Length[polys]], {2}],
               Length[Intersection @@ polys[[#]]] >= 2 &];

Graphics3D[{{Opacity[.75], Sphere[]}, {AbsoluteThickness[2], edges},
            {Red, Sphere[sp, 0.02]}}, Boxed -> False]
```

- Diagrama de Voronoi - Polígono

```
m = VoronoiMesh @ RandomReal[{-1, 1}, {25, 2}];
p = Table[{Cos[i], Sin[i]}, {i, Pi/3., 2 Pi, Pi/3}] // Polygon // DiscretizeGraphics;

DeleteCases[
  RegionIntersection[DiscretizeGraphics@#, p] & /@  MeshPrimitives[m, 2],
  _RegionIntersection
 ] // RegionPlot[#, AspectRatio -> Automatic] &
```

Bibliografía

[1] Rodríguez Gavilán, Ginés (s/f).

Representación gráfica de superficies mediantes mapas térmicos.

Universidad Politécnica de Madrid.

[2] Grima, Clara (2012).

Cada uno en su región y Voronoi en la de todos. naukas.com:

Recuperado de http://naukas.com/2011/12/23/cada-uno-en-su-region-y-voronoi-en-la-de-todos/.

[3] Anónimo (2012).

Animating a Voronoi Diagram.

Recuperado de https://mathematica.stackexchange.com/questions/3963/animating-a-voronoi-diagram.

[4] Rodríguez, Ulises Martínez (2015).

Aplicación de la Geometría Computacional en la Reconstrucción 3D Basada en Diagramas de Voronoi.

Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias Físico Matemáticas.

[5] Diagramas de Voronoi construidos con software GeoGebra disponible en www.geogebra.org.

[6] Télam. 18 de Noviembre de 2017.

La Pampa en alerta por epidemia de diarrea.

Aire de Santa Fe DIGITAL.