UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE CIENCIAS

GEOMETRIA COMPUTACIONAL: EL CASO DE LOS DIAGRAMAS DE VORONOI

T E S I S

QUE, PARA OBTENER EL TITULO DE

A C T U A R I O

P R E S E N T A :

SERGIO AGUIRRE RIOS

MEXICO, D. F. 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Indice

I. Introducción

a) Antecedentes e Historia

El impresionante auge que las computadoras han alcanzado en las últimas décadas se ha debido en parte a la constante preocupación de algunas personas por hacerlas cada vez más eficientes y accesibles. Han incursionado con un rotundo éxito en los negocios, la industria y la investigación científica. Sin embargo, no dejan de ser simples máquinas incapaces de razonar, que sólo pueden hacer lo que el mismo ser humano les indica. Es por esto que de nada sirve tener computadoras muy sofisticadas, si no se cuenta también con algoritmos eficientes que se encarguen de proporcionar la lógica necesaria para efectuar las tareas deseadas. Ya no es suficiente con encontrar una solución a un problema, sino que hay que encontrar una que pueda optimizar el tiempo de ejecución, los recursos que ocupa, etc.

Cuando se habla de problemas geométricos esto se vuelve más evidente ya que las caracterizaciones clásicas de los objetos geométricos frecuentemente no llevan al diseño de "buenos" algoritmos, por lo que primero se necesita identificar los conceptos que nos pueden ser útiles y luego establecer sus propiedades.

La Geometría Computacional se dedica precisamente a esto, es decir a buscar y mejorar los procedimientos computacionales que se pueden aplicar para resolver problemas inherentemente geométricos tales como el problema (euclideano) del agente viajero, el del árbol de peso mínimo, el de líneas ocultas y algunos de programación lineal entre muchos otros. Tal y como se conoce hoy en día, se puede decir que es el producto del esfuerzo de entre otros M.I Shamos [1,2,3] por tener algo más que una simple colección aislada de resultados.

Estos problemas provienen de una amplia variedad de áreas aplicadas como la graficación, la robótica, el diseño y la manufactura asistidos por computadora (CAD y CAM), etc; y a pesar de que desde el siglo pasado existían estudios algoritmicos sobre ellos, no es sino hasta ahora que los algoritmos geométricos están siendo estudiados en forma sistemática, mediante el análisis de temas relacionados con la representación de los objetos.

La Geometría Computacional ha tenido que dar respuesta a preguntas tales como ¿cómo modelar una curva, superficie o sólido?, ¿cómo presentarle al usuario la información geométrica?, ¿cómo hacer para que el usuario interactúe de una forma natural con el modelo geométrico?, ¿cómo saber si un algoritmo es óptimo o si todavía se puede mejorar?

Obviamente estas preguntas no son sencillas, por lo que para poder responderlas se necesita de toda la ayuda posible, la cual puede provenir de áreas de la matemática tales como topología, combinatoria, álgebra, probabilidad y por supuesto de la geometría, así como de las relativas a la ciencia de la computación como algoritmos de gráficas, estructuras de datos y optimización.

En este trabajo nos ocuparemos de estudiar dos estructuras muy importantes llamadas el diagrama de Voronoi y la triangulación de Delaunay y que bajo el campo de estudio de la Geometría se conocen simplemente como teselaciones.

Desafortunadamente, el estudio de las teselaciones como parte de la matemática se relegó hasta fines del siglo pasado, quedando como excepción el trabajo realizado por el célebre astrónomo alemán Johannes Kepler (1571-1630) al respecto [6]. En la actualidad se debe a los

químicos y cristalógrafos una buena parte del conocimiento sobre este tópico.

A pesar de que los diagramas de Voronoi pueden definirse para cualquier espacio R^n con cualquier métrica L_p [4,5,6], aquí nos restringiremos al caso del plano y la distancia euclideana por ser las más comunes.

b) Definiciones

Básicamente un tejado o teselación es un conjunto de celdas que cubren un determinado espacio sin intersectarse. Como se había mencionado, aquí hablaremos de teselaciones del plano. Estas presentan la propiedad de que si las celdas que las componen no incluyen su frontera, entonces todo punto en el plano cae en una y sólo una celda, o bien en un lado de ella.

Pensemos en el problema de acomodar k polígonos regulares alrededor de un punto en el plano, de tal forma que un vértice de cada polígono toque al punto. Dichos polígonos se dice que forman un conjunto cristalográfico que denotaremos como $(l_1,\ldots,l_k)$ con l_i como el número de lados del iésimo polígono y donde se debe satisfacer la ecuación $\sum_{i=1}^{k}(l_i - 2) / l_i = 2$ (esto debido a que cada ángulo interior en un eneágono regular mide $\Pi (n - 2) / n$) y que en términos comúnes simplemente significa que el punto debe estar "completamente rodeado" por los polígonos. Los l_i's se agrupan en el menor orden lexicográfico posible de entre todas las ordenaciones cíclicas y las reversiones con el fin de lograr una representación única.

Aún con todas las combinaciones posibles que se podrían hacer en una agrupación de éstas, sólo existen 21 conjuntos cristalográficos diferentes [6] y sólo 11 de ellos dan origen a una teselación que puede ser regular (si los elementos de dicho conjunto son todos iguales) o semi-regular (si no lo son) y se llaman tejados Arquimedeanos.

Todas estas teselaciones tienen la particularidad de que los puntos que les dan origen presentan una cierta distribución conocida. Cuando los puntos estan distribuidos en forma aleatoria la teselación que resulta es el llamado diagrama de Voronoi.

Este último está compuesto de una serie de polígonos convexos (uno alrededor de cada punto del conjunto) que reciben el nombre de polígonos de Voronoi (tambien llamados de Dirichlet, de Wigner-Seitz, de Theissen o en forma de "S"). Los lados que los conforman recibirán el nombre de lados de Voronoi. Asimismo, cuando dos de los polígonos compartan un mismo lado, a los puntos que los originan les llamaremos vecinos de Voronoi.

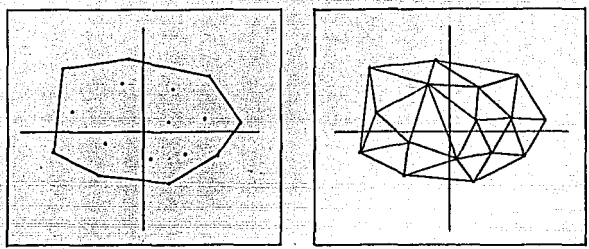
Para poder entrar de lleno al estudio de estas estructuras necesitamos conocer las siguientes definiciones.

Definición 1.- La envoltura convexa de un conjunto de puntos P en R^n es la frontera del dominio convexo más pequeño en R^n que contiene a P (ver Figura 1.2.1(a)).

Definición 2.- Una gráfica del plano es una gráfica G = (V,L) (V es el conjunto de vértices, L es el conjunto de lados) que se puede trazar en el plano sin ninguna intersección (excepto en los vértices).

Definición 3.- Una triangulación de un conjunto finito P de puntos es una gráfica del plano de P con el mayor número de lados posible (esto es equivalente a decir que la triangulación de P se obtiene al unir los puntos del conjunto por segmentos de linea rectos que no se intersectan, de tal forma que cada región interna a la envoltura convexa de P es un triángulo),(ver Figura 1.2.1(b)).

Aquí podriamos definir tambien, sin mucha formalidad, la complejidad de un algoritmo como una función que depende de uno o varios parámetros y que nos da una idea del tamaño de un problema a resolver. Cuando se habla de ella, se usa una notación muy conocida y que a continuación se presenta.



(a) (b)

Figura 1.2.1 (a) La envoltura convexa de un conjunto de puntos en R^2; (b) Una triangulación del mismo conjunto.

- $O(f(n))$ denota el conjunto de todas las funciones $g(n)$ tales que existen constantes positivas C y n_0 las que cumplen con $|g(n)| \leq C\, f(n)$ para todas las $n \geq n_0$ (se usa para indicar funciones que describen cotas superiores).
- $\Omega(f(n))$ denota el conjunto de todas las funciones $g(n)$ tales que existen constantes positivas C y n_0 las que cumplen con $g(n) \geq C\, f(n)$ para todas las $n \geq n_0$ (usado para las funciones que denotan cotas inferiores).
- $\theta(f(n))$ denota el conjunto de todas las funciones $g(n)$ tales que existen constantes positivas C_1, C_2 y n_0, las que cumplen con la siguiente desigualdad: $C_1\, f(n) \leq g(n) \leq C_2\, f(n)$ para todo $n \geq n_0$ (se usa para indicar funciones que son del mismo orden que $f(n)$, es decir para hablar de algoritmos óptimos).

Por ejemplo, si tenemos una función $g(n) = 3n^2 - n + 5$ entonces es fácil concluir que es del orden de $O(n^2)$ ya que si $c = 3$ y $n_0 = 5$ $|g(n)| = |3n^2 - n + 5| \leq 3n^2 = C\, f(n)$ para toda $n \geq n_0$.

c) Algunas aplicaciones de los diagramas de Voronoi

1) Procesamiento de Patrones de Puntos [6]: Al procesar información visual a veces tenemos que tratar no sólo con imágenes, sino también con patrones de puntos. Si estos fueron generados por algún proceso conocido, entonces el análisis se facilita mucho; pero si no es el caso, entonces lo primero es tratar de ver que características presenta su estructura. Para esto lo que se necesita es estudiar las posiciones relativas de los puntos vecinos.

Obviamente, el concepto de vecinos que aquí nos interesa es el de Voronoi. Consideraremos que la vecindad de un punto es la región encerrada por su correspondiente polígono, ya que en la forma y tamaño de este se reflejará el ambiente local del punto.

En el problema de "segmentación por análisis de grupos" [7] lo que se necesita precisamente es formar grupos de puntos, es decir, encontrar una partición de tal forma que los miembros de un mismo grupo sean "similares" según alguna regla de decisión. Como suponemos que estamos en el plano, podemos basar esa similitud en las vecindades de los puntos.

Nos fijamos en las características que esas vecindades de Voronoi tienen, por ejemplo, su área y perímetro, su alargamiento o compactamiento, la dirección del eje principal (la diagonal mayor), las variaciones en las longitudes de sus lados, etc, que al unirse con algunas propiedades conjuntas como la distancia entre vecinos forman una serie de atributos que pueden ser los componentes de un vector multidimensional. A este se le puede entonces aplicar una medida de semejanza basada en la distancia euclideana en varias dimensiones, para así poder decidir que puntos deben conformar los grupos. Se puede descubrir si estos presentan mayor densidad según la orientación, si existe alguna curva o "cuello", si presentan forma globular o no, etc.

Por otro lado el problema de la "extracción de la frontera perceptible" [16] también se puede analizar bajo el mismo enfoque de vecindad. Una frontera perceptible es la frontera "que se nota intuitivamente" en un patrón de puntos debido a la localización relativa de estos, sin ninguna interpretación semántica o cognoscitiva. Se empieza uniendo todos los puntos que están en el borde y después se usa una regla sencilla para cuando se desee agregar algún punto que forme una concavidad. Así, para añadir un punto p_j a dicha frontera, entre dos puntos p_i y p_k que ya pertenecen a ella, nos fijamos si la línea que une a estos puntos "atraviesa" el polígono de Voronoi de p_j. En caso afirmativo se acepta al punto, si no se rechaza.

2) Cálculo de la Precipitación Pluvial en una Area [8]: En la hidrología forestal se usa al diagrama de Voronoi para calcular la precipitación diaria, mensual, anual o de una tormenta en una área.

Dado un río y sus posibles afluentes nos debemos fijar en su respectiva cuenca, ya que ésta va a ser el área a la cual le deseamos calcular su precipitación. Dentro o cerca de ésta área se tendrán algunas estaciones que medirán la cantidad de lluvia que cayó durante el período considerado y que estarán representadas por los puntos a los cuales se les trazarán sus correspondientes polígonos de Voronoi.

Para efectuar la estimación lo único que se hace es ponderar la cantidad de lluvia que cayó en una estación determinada, por el porcentaje de la cuenca que está más cerca de dicha estación que de cualquier otra. La suma de todos estos valores es precisamente la estimación para esa área y tiene la ventaja de que es mucho más precisa que si se hubiera obtenido a partir de un simple promedio.

3) Análisis del crecimiento y la mortalidad de plantas individuales como función del "área disponible" [9,10]: En Ecología se utilizan los diagramas de Voronoi para ver como se comportan ciertas plantas

como resultado de la competencia que se genera entre individuos de la misma especie.

Lo primero que se hace es trazar un mapa de la distribución exacta que guarda alguna población de plantas en un momento determinado (generalmente poco después de la aparición de éstas). A cada uno de estos organismos se le traza su correspondiente polígono de Voronoi, para que cuando se tenga el diagrama completo se pueda obtener la distribución de frecuencia de las áreas de dichos polígonos. Pasado un tiempo se repite el proceso con los individuos que hayan sobrevivido, con el fin de poder realizar comparaciones.

Gracias a esto, se han podido concluir algunos resultados, tales como: i) las plantas que tienen vecinos "muy cercanos" (es decir con polígonos más pequeños) tienen mayor posibilidad de morir que las demás, ii) de entre los sobrevivientes, los que habían tenido vecinos "muy cercanos" eran los más pequeños y iii) a pesar de que las distribuciones de las áreas al principio estaban fuertemente sesgadas (muchos individuos pequeños y pocos grandes) después tendían a volverse simétricas, ya que los sujetos débiles (pequeños) mueren.

Resultados similares se pueden obtener con otro tipo de poblaciones (algunos crustáceos, por ejemplo), lo que se debe a una cualidad de estos diagramas y es que los polígonos de Voronoi de un patrón de puntos con distribución no uniforme tienen una área decreciente conforme uno se dirige a la zona de mayor densidad.

4) Simulación Discreta de la Filotaxia [17,18]: La Filotaxia es una rama de la Biología que estudia la estructura geométrica de las plantas, por ejemplo, la disposición de las hojas en el tallo y de los pétalos en la flor, la distribución de las areólas, etc.

En la Filotaxia aparece a menudo la serie de Fibonacci. En particular, en el centro de algunas flores, como la margarita y el girasol los organelos están distribuidos en espirales logarítmicas y

siempre el número de espirales que se abren a la derecha y el número de espirales que se abren hacia la izquierda son términos consecutivos de dicha serie.

Esto se observa también en las piñas, tanto en las tropicales como en las de las pináceas. Una propuesta para reconstruir ésta filotaxia está en el trabajo de Rivier *et al*, que a continuación se discute someramente.

Se propone un mecanismo con simetría polar para generar los centros de las celdas, dado por la siguiente pareja de relaciones

$$r(l) = a\, l^{1/2} \quad y \quad \theta(l) = 2\pi\lambda\, l \qquad (1)$$

En ésta relación $l = 1,\ldots,n$ etiqueta las celdas individuales procediendo hacia afuera a partir del centro de simetría ("a" es la típica dimensión lineal de las celdas). λ es proporcional a l y r aumenta conforme lo hace $l^{1/2}$, para así poder conseguir uniformidad y homogeneidad en las densidades de las celdas. El algoritmo (1) introduce regularmente las celdas en una espiral, en el orden inverso en que hojas sucesivas brotan del tallo de una planta. Una vez que se han especificado los centros, las celdas se construyen mediante una partición de Voronoi del espacio con respecto a ellos. La única información estructural que es cifrada es el ángulo λ entre areólas (brotes jóvenes que eventualmente dan lugar a hojas o espinas de una planta), y el hecho de que areólas sucesivas aparecen en intervalos regulares de tiempo. La forma de cada areóla (hexágono, pentágono, etc) no se proporciona, sino que resulta una simple consecuencia de como se "llena" una área.

Cuando, en particular, se elige al parámetro estructural λ como $1/\tau$ (siendo λ la proporción áurea) se obtiene, mediante el mecanismo descrito anteriormente, la siguiente figura:

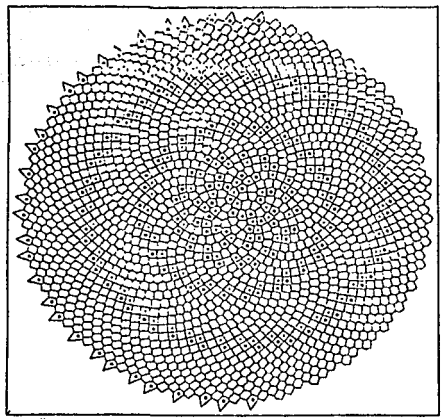


Figura 1.3.1 Reconstrucción de una filotaxia real según el modelo proporcionado por Rivier *et al.*

5) Iteraciones de Voronoi con los Centros de Masa: Hace poco tiempo en un coloquio de matemáticas sobre "Geometría Intuitiva", M. N. Bleicher del Departamento de Matemáticas de la Universidad de Wisconsin, en Madison, planteó el siguiente problema:

Dados n puntos en una región acotada consideraremos la siguiente operación

1) División .- La región es dividida en celdas tales que cada uno

los puntos se encuentra en la celda que incluye a todos aquellos puntos que están más cerca de el que de cualquier otro.

2) Centralización .- Cada punto es movido al centroide o centro de masa de la celda correspondiente.

Estos dos pasos se van a iterar y las preguntas que surgen son ¿existe una posición limitante o es posible un comportamiento cíclico? ¿cómo se ven las posiciones limitantes (si es que existen) para casos especiales?

Inmediatamente reconocemos que en el paso uno lo que se necesita es precisamente el diagrama de Voronoi del conjunto de n puntos, razón por la cual este sistema resulta una valiosa herramienta de ayuda para los investigadores interesados en "atacar" este problema abierto. Así se pueden efectuar las iteraciones en la computadora, pudiendo visualizar varios diagramas juntos (encimados) para realizar comparaciones y ver que clase de comportamiento se está presentando. Obviamente, el número de diagramas que podrían guardarse al mismo tiempo dependerá de la cantidad de memoria disponible y del tamaño del conjunto de puntos. Sin embargo, al proporcionar el sistema la opción de poder "salvar" los centros de masa y los vértices de Voronoi de un diagrama esto no resulta tan grave.

II. Las Teselaciones de Voronoi y de Delaunay

a) Definiciones

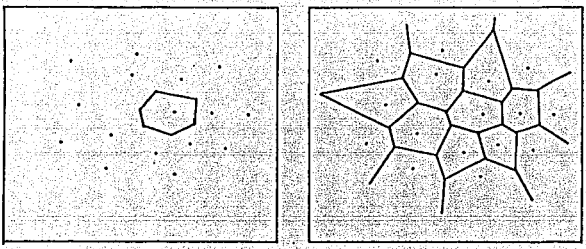
El diagrama de Voronoi es una teselación cuya estructura tan particular la convierte en una herramienta útil dispuesta al servicio de las ciencias. Para su construcción existen varios algoritmos que se ha demostrado son óptimos cuando se aplican sobre un conjunto de n puntos en el plano y que bien implementados darían origen a un producto práctico y posiblemente no muy costoso (en cuanto a recursos requeridos para dicha construcción y posterior aprovechamiento).

En teoría, su construcción es relativamente sencilla. Se parte de un conjunto inicial de puntos, llamémosle $P = \{p_1,\ldots,p_n\}$ en el cual $n \geq 3$. Se hacen las suposiciones de que no existan cuatro puntos que sean cocirculares y que no todos los puntos del conjunto sean colineales (esto se explicará más adelante).

Empezamos construyendo lo que se conoce como el polígono de Voronoi de cada punto en el conjunto. Si $p_i \in P$, entonces dicho polígono, denotado por $DV(p_i)$ se define como el conjunto de puntos en el plano que están más cerca de p_i que de cualquier otro punto de P, con lo cual si $H(p_i,p_j)$ es el semiplano que contiene a todos los puntos que están más cerca de p_i que de p_j es claro que $DV(p_i) = \cap_{j\neq i}^{n} H(p_i,p_j)$ y es una región poligonal convexa, que a lo más podría llegar a tener (n-1) lados (ver Figura 2.1.1(a)).

Con esto queda definido el diagrama de Voronoi del conjunto P como $DV(P) = \cup_{i=1}^{n} DV(p_i)$ (ver Figura 2.1.1(b)).

Una interpretación práctica de un diagrama de Voronoi se produce cuando se habla del problema de las oficinas postales. Ahí los puntos



(a) (b)

Figura 2.1.1 (a) El polígono de Voronoi de un punto p_i; (b) El diagrama de Voronoi.

de P se interpretan como oficinas de correo y la pregunta que se plantea es que dado un nuevo punto (x,y) que podría, por ejemplo, representar a un nuevo residente, ¿cuál es la oficina que le queda más cerca?

Es claro que una solución inicial obvia sería sacar todas las distancias entre el punto (x,y) y los puntos de P y ver cuál es la menor, sin embargo es bastante factible pensar que no sólo vamos a formular esa pregunta para un solo punto, razón por la cual en la práctica este proceso resulta obsoleto e incosteable (cada vez habría que estar calculando n distancias, lo cual resultaría muy tardado). Es aquí donde el problema inicial se transforma en el problema de obtener el diagrama de Voronoi correspondiente, ya que a fin de cuentas todo se reduce a saber en que polígono cae el punto en cuestión. A cada uno de estos polígonos se le interpretará precisamente como la "zona de acción" de la oficina postal correspondiente.

En lo referente a las propiedades que esta teselación tiene, algunas

de ellas podrían entenderse mejor si primero vemos como es que un proceso de crecimiento puede generar dicho diagrama.

Pensemos en los puntos de P como si fueran los núcleos de unas celdas que se van a desarrollar. Todas ellas se empiezan a expander al mismo tiempo (a partir de su núcleo) en forma circular y en una proporción uniforme. Cuando el borde de una celda "choca" con el de otra, ese primer punto de contacto (que es el punto medio entre sus núcleos) se irá alargando, creando una recta que crecerá indefinidamente a menos que tope con una tercera celda. De ese conjunto de rectas es de lo que finalmente se compone el diagrama de Voronoi.

De lo anterior se desprenden, aunque sea en forma intuitiva, dos hechos muy importantes. Uno es que cada recta de las que se acaban de mencionar no es más que un segmento del bisector perpendicular de un par de puntos de P y el otro es que el diagrama se compone tanto de polígonos cerrados como abiertos (estos originados por las rectas que crecen sin "topar" con otra línea en expansión).

Ya con esto podemos entrar de lleno a las siguientes propiedades:
(las demostraciones de las más importantes se encuentran en el Apéndice 1).

Propiedad 1. Cada vértice del diagrama de Voronoi es la intersección común de exactamente tres lados del diagrama.
Lo que nos indica esto es que los vértices de Voronoi son los centros de círculos definidos por tres puntos del conjunto. Si, para un vértice v, se denota como C(v) a dicho círculo entonces es fácil checar lo siguiente:
Propiedad 2. Para cada vértice v del diagrama de Voronoi de P, el círculo C(v) no contiene ningún otro punto de P.
La importancia de esto estriba en que nos proporciona una regla de

decisión que se puede aplicar para construir el diagrama. En este trabajo sin embargo, no se usa de forma directa, es decir, se utiliza solamente como "una ayuda intermedia" para construir cierta estructura a partir de la cual se puedan derivar finalmente los polígonos de Voronoi.

Propiedad 3. El polígono $DV(p_i)$ es no acotado si y sólo si p_i es un punto que está sobre la frontera de la envoltura convexa del conjunto P.

Sin embargo, la propiedad más interesante del diagrama todavía no se ha mencionado, y es que cuando se une con una línea recta a todo par de vecinos de Voronoi, el área comprendida dentro de la envoltura convexa queda completamente dividida por una serie de triángulos, dando origen a lo que se conoce como el dual del diagrama; la triangulación de Delaunay.

En cuanto a las propiedades que tiene podemos empezar con las siguientes:

Propiedad 4. Dado un conjunto $P = \{p_1,\ldots,p_n\}$ de puntos, cualquier lado (p_i,p_j) es un lado de Delaunay de DT(P) si y sólo si existe un punto x tal que el círculo centrado en x y que pasa por p_i y p_j no contiene en su interior otro punto de P.

Propiedad 5. Dado un conjunto $P = \{p_1,\ldots,p_n\}$ de puntos, el lado (p_i,p_j) en la frontera de la envoltura convexa de P es un lado de Delaunay.

Es claro que esta propiedad resulta una simple consecuencia de la anterior y se recurrirá a ella en la siguiente sección, cuando se trate de construir la triangulación.

Propiedad 6. Dado un conjunto $P = \{p_1,\ldots,p_n\}$ de puntos, el triángulo $p_ip_jp_k$ es un triángulo de Delaunay de DT(P) si y sólo si su circuncírculo no contiene otro punto de P en su interior.

Es claro que si se tiene el diagrama de Voronoi de un conjunto de puntos, a partir de él se puede construir sin mucha dificultad la triangulación de Delaunay y viceversa. Sin embargo si este no es el caso, la triangulación se puede obtener directamente mediante la aplicación de alguno de los varios criterios que existen para ese efecto.

A la propiedad 6 se le llama precisamente el criterio del círculo y su validez radica en la definición misma del dual y en la segunda propiedad del diagrama de Voronoi. Otro criterio (que se puede ver es equivalente) es el del ángulo max-min en donde lo que se busca es evitar en la medida de lo posible los ángulos "pequeños", es decir, se maximiza el menor ángulo de todos los triángulos pertenecientes a la triangulación.

Para comprobar que este criterio también nos lleva a la triangulación de Delaunay nos ayudaremos de un Procedimiento de Optimización Local (POL) que funciona de la siguiente manera: tomamos un lado l que no forme parte de la envoltura convexa, llamamos Q al cuadrilátero que tiene a l como una de sus diagonales y consideramos el circuncírculo que pasa por uno de los dos triángulos que forman Q. Si el cuarto vértice del cuadrilátero cae dentro del círculo, cambiamos a l por la otra diagonal, de lo contrario no se toma ninguna acción. Así un lado de la triangulación se dirá que es localmente óptimo si al aplicar el POL no hay intercambio de diagonal.

Necesitamos ahora crear un orden para todas las triangulaciones posibles de un conjunto. Para ello nos ayudaremos de la siguiente propiedad que, además de establecer como una constante al número de triángulos en una triangulación, nos da la pauta para que posteriormente calculemos la cantidad de espacio que la estructura, a utilizar para las construcciones, necesita.

Propiedad 7. Dado un conjunto P de n puntos, cualquier triangulación T(P) tiene el mismo número de triángulos $n_t = 2(n-1) - n_e$ y el mismo número de lados $n_l = 3(n-1) - n_e$, donde n_e es el número de puntos en la envoltura convexa de P.

Supóngase que tenemos una situación como la establecida en esta propiedad y que a cada triángulo le asignamos un valor que va a ser la medida de su ángulo más pequeño. Reunimos estos valores asociados de todos los triángulos y los ordenamos en forma no decreciente. Ya ordenados, estos valores van a ser los componentes de un vector que siempre tendrá n_t dimensiones, con lo que definiremos que si T y T' son dos triangulaciones sobre el mismo conjunto, entonces $T < T'$ si y sólo si el vector asociado de T es lexicográficamente menor que el de T'.

Con esto es fácil checar la veracidad de las siguientes propiedades:

Propiedad 8. Dada una triangulación T, si una aplicación del POL a un lado l produce un intercambio con un lado l' originando una nueva triangulación T' entonces $T < T'$.

Propiedad 9. Todos los lados internos de una triangulación T de un conjunto finito P son localmente óptimos si y sólo si ningún punto de P cae dentro del circuncírculo de un triángulo de T.

Propiedad 10. Una triangulación T(P) es una triangulación de Delaunay si y sólo si su vector indicador es lexicográficamente máximo, es decir, ninguna triangulación le sigue en el ordenamiento lineal.

Tenemos finalmente que como consecuencia inmediata de esta última propiedad el criterio del ángulo max-min sirve para construir la triangulación de Delaunay que es lo que queríamos probar.

Debe resultar clara la importancia de las suposiciones iniciales, ya que si no se respetan tendremos, además de otras consecuencias, que esta triangulación no sería única y podría ser degenerada.

b) Métodos Propuestos

En la sección anterior definimos a la triangulación de Delaunay con base en el diagrama de Voronoi. Sin embargo, mencionamos que su construcción no necesariamente debía llevarse a cabo bajo el mismo enfoque, sino que podía realizarse en forma directa con la ayuda de algunos criterios. Esto último es precisamente lo que se aplica en este trabajo.

El algoritmo destinado a obtener esta triangulación y que aquí se emplea se debe a Lee y Schachter [6,11] y se basa en el conocido principio de "divide y vencerás" (concepto que se utiliza mucho en el campo de la Geometría Computacional). Básicamente, consiste en dividir el conjunto de puntos en dos partes, construir en forma recursiva la triangulación de cada una de ellas y después unir ambas para obtener el resultado final.

La estructura que se utiliza requiere que a cada punto en el conjunto se le asigne una lista de adyacencia (compuesta por puntos) doblemente ligada y circular. Una lista es una estructura de datos en donde a cada nodo de información se le agrega un campo (de tipo apuntador) que contenga la dirección del siguiente nodo. El último nodo tiene en dicho campo un valor especial que se denomina "marca de fin de lista". Para trabajar con una lista se requiere de una variable externa que contenga la dirección del primer elemento. Si la lista es circular entonces el último nodo "apunta" al primer elemento en vez de a la señal de terminación y si además tiene doble liga esto nos indica que en cada registro se incluye también la dirección del elemento anterior.

Escribiremos la lista de adyacencia de p_i como $p_{i1},\ldots,p_{ik}$ siempre

que cada segmento $p_i p_{ij}$, $j = 1,\ldots,k$ sea un lado de Delaunay y deberá estar ordenada de forma tal que los puntos que la componen guarden el mismo orden que como se encuentran físicamente los lados a que representan en la triangulación.

Este algoritmo hace uso de las siguientes rutinas:

Pred(p_i,p_{ij}) – está función regresa el punto que se encuentra en la lista de adyacencia de p_i inmediatamente después de p_{ij} si se recorre esta en la sentido de las manecillas del reloj.

Suce(p_i,p_{ij}) – es análoga a la anterior, pero usando el sentido contrario a las manecillas del reloj.

Primero(p_i) – regresa el punto p_{i1} que está en la lista de adyacencia de p_i y que es la "cabeza" de dicha lista. Si p_i está en la envoltura convexa de una triangulación, entonces p_{i1} representa al punto que le sigue a p_i si se recorre dicha envoltura en el sentido contrario a las manecillas del reloj.

Inserta(p_i,p_j) – este procedimiento se encarga de insertar al punto p_j en la lista de adyacencia de p_i (en la posición correcta) y viceversa.

Borra(p_i,p_j) – aquí se borra a p_j de la lista de adyacencia de p_i y viceversa.

PruebaC(p_h,p_i,p_j,p_m) – esta es una función booleana que regresa verdadero si el circuncírculo del triángulo $p_h p_i p_j$ no contiene a p_m en su interior, de otro modo regresa falso.

Existen otras dos rutinas necesarias pero, a diferencia de las anteriores, no reciben puntos como parámetros sino que su entrada consiste de una envoltura convexa:

MD(P) – es una función que regresa el punto que está más a la derecha en la envoltura convexa de un conjunto P.

MI(P) – análoga a la anterior, pero que regresa el punto más a la izquierda.

Para poder empezar a aplicar el algoritmo antes que nada debemos ordenar a los puntos del conjunto P en orden lexicográfico ascendente, es decir que

$$(x_i,y_i) = p_i < p_j \quad \Leftrightarrow \quad \begin{cases} x_i < x_j \\ \text{ó} \\ x_i = x_j \ \text{ y } \ y_i < y_j. \end{cases}$$

Enseguida renombramos a los índices de tal forma que $p_1 < \ldots < p_n$. Después procedemos a dividir al conjunto en dos partes $P_I = \{p_1,\ldots,p_{\lfloor n/2 \rfloor}\}$ y $P_D = \{p_{\lfloor n/2 \rfloor+1},\ldots,p_n\}$ a las cuales se les construirá recursivamente su triangulación correspondiente, o sea $DT(P_I)$ y $DT(P_D)$, que finalmente se unirán para conseguir $DT(P_I \cup P_D)$.

De la sección anterior, sabemos que la envoltura convexa de un conjunto forma parte de la triangulación de Delaunay correspondiente, por lo cual para poder unir dos triangulaciones resulta bastante razonable que el primer paso sea el encontrar la envoltura convexa de $P = P_I \cup P_D$. Como ya tenemos las envolturas de cada una de las dos "mitades", sólo hay que unirlas; lo que se logra hallando la tangente común superior e inferior.

La siguiente subrutina se encarga de hallar la tangente común inferior:

- SUBRUTINA ENVOLTURA
- Esta subrutina recibe como entrada dos envolturas convexas.
- L(X,Y) denota la línea dirigida del punto X al Y.

```
BEGIN Tangente Inferior
      I  ← MD(P_I);  D ← MI(P_D)
      DS ← Primero(D); AUX ← Primero(I); IS ← Pred(I,AUX)
 A:   IF (DS está_a_la_derecha_de L(I,D)) THEN
         AUX ← DS
         DS  ← Suce(DS,D)
         D   ← AUX
      ELSE
         IF (IS está_a_la_derecha_de L(I,D)) THEN
            AUX ← IS
            IS  ← Pred(IS,I)
            I   ← AUX
         ELSE
            RETURN (I,D)
         ENDIF
      ENDIF
      GOTO A
END Tangente Inferior
```

La explicación de este procedimiento es muy simple. La idea es empezar con la recta que tiene por extremos al punto que está más a la derecha de P_I y al que está más a la izquierda de P_D, denotados I y D respectivamente, e irla bajando hasta que ningún punto de los dos conjuntos quede a la derecha de ella (ver Figura 2.2.1). Es así que IS es el punto que le sigue a I si se recorre la envoltura convexa de P_I en el sentido de las manecillas del reloj y DS el que le sigue a D en la otra envoltura en el sentido opuesto, y que son precisamente los puntos que sirven para ver cuando es que se ha alcanzado la tangente deseada.

En cuanto a la tangente común superior, ésta se puede encontrar

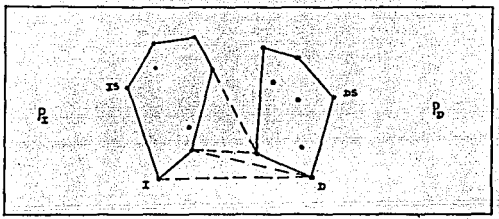


Figura 2.2.1 Generación de la tangente común inferior de un par de triangulaciones.

mediante un proceso análogo. Hecho esto, entonces se llama a la función que combina las dos triangulaciones:

- SUBRUTINA COMBINAR
- Esta subrutina recibe como parámetros dos triangulaciones y sus
- tangentes comunes inferior y superior.

```
BEGIN Combinar Triangulaciones
    TI ← Tangente Común Inferior
    TS ← Tangente Común Superior
    I  ← Extremo Izquierdo de TI
    D  ← Extremo Derecho de TI
    DO UNTIL (TI Igual_a TS)
       A ← B ← FALSE
       Inserta(I,D)
       D1 ← Pred(D,I)
       IF (D1 está_a_la_izquierda_de L(I,D)) THEN
          D2 ← Pred(D,D1)
          DO UNTIL (PruebaC(D1,L,D,D2))
             Borra(D,D1)
             D1 ← D2
```

(continúa)

```
            D2 ← Pred(D,D1)
         END DO UNTIL
      ELSE
         A ← TRUE
      ENDIF
      I1 ← Suce(I,D)
      IF (I1 está_a_la_derecha_de L(D,I)) THEN
         I2 ← Suce(I,I1)
         DO UNTIL (PruebaC(I,D,I1,I2))
            Borra(I,I1)
            I1 ← I2
            I2 ← Suce(I,I1)
         END DO UNTIL
      ELSE
         B ← TRUE
      ENDIF
      IF (A) THEN
         I ← I1
      ELSE
         IF (B) THEN
            D ← D1
         ELSE
            IF (PruebaC(I,D,D1,I1)) THEN
               D ← D1
            ELSE
               I ← I1
            ENDIF
         ENDIF
      ENDIF
      TI ← L(I,D)
   END DO UNTIL
```

(continúa)

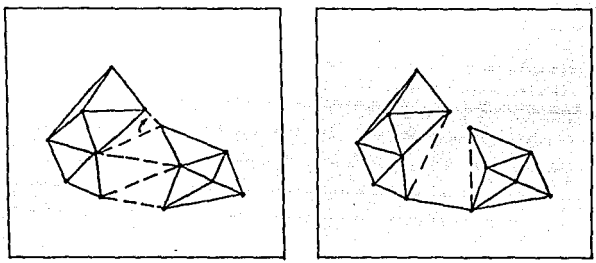
Inserta(I,D)
END Combinar Triangulaciones

Resulta evidente que esta subrutina no es tan simple como la anterior, por lo que una explicación más detallada se vuelve prácticamente una obligación.

La idea esencial consiste en ir insertando sucesivamente las líneas que unirán a las dos triangulaciones. Empezamos con la tangente común inferior y vamos subiendo hasta "topar" con la tangente superior.

Después de haber insertado la primera tangente tenemos dos opciones en cuanto a como colocar la siguiente línea. Podemos unir el extremo izquierdo de la tangente (en P_I) con un punto adyacente al extremo derecho (en P_D) o viceversa. Nótese que esto significa que lo que hay que escoger, es que extremo se va a quedar fijo y cual va a "subir", para así dar origen a un nuevo lado de la triangulación. Esta decisión puede tomarse en algunos casos en forma inmediata, si la nueva línea no está en posibilidad de subir por alguna de las dos triangulaciones (si no hay un punto más arriba de la línea por ese lado), (ver Figura 2.2.2 (a)) y hay que decidirse por el extremo opuesto. En el algoritmo se checa ésta posibilidad para el lado derecho (P_D) en donde se encuentra el primer "IF". Si se ve, sin embargo, que esto no se da, entonces se procede a borrar todo lado de P_D que no cumpla con el criterio del círculo y que por lo tanto no pueda estar en la unión de las dos triangulaciones (por ejemplo, los lados que estában en las envolturas convexas de cada una de las dos partes, pero que no están presentes en la envoltura de P pueden llegar a ser borrados), (ver Figura 2.2.2 (b)). En seguida se checan de nuevo estas posibilidades, pero ahora para el lado izquierdo (localizar el segundo "IF"), y así pasamos a la última parte del algoritmo, que es donde se crea

explícitamente la nueva línea. Si se había visto que uno de los dos extremos ya no podía subir, entonces se hace la asignación correspondiente, si no, hay que aplicar de nuevo el criterio del círculo para así tomar una decisión final. Por cuestiones de simplificación en el algoritmo se interpreta a la nueva línea como si fuera ahora la tangente común inferior (aunque en realidad no lo es) y así sucesivamente hasta alcanzar a la superior.



(a) (b)

Figura 2.2.2 (a) Unica posibilidad de nueva línea; (b) Borrado de líneas que no pertenecen a la unión de dos triangulaciones.

Este algoritmo corre en un tiempo del O(n log(n)) y como se puede demostrar [1] que el construir cualquier triangulación sobre n puntos requiere un tiempo de Ω(n logn) de aquí se concluye que es óptimo, por lo que resulta muy atractivo aún para conjuntos "grandes" de datos.

Ahora, con respecto al método usado aquí para obtener el diagrama de Voronoi, tenemos que se basa casi en su totalidad en la definición misma del diagrama. Vimos que cada polígono de los que lo conforman era el resultado de hacer la intersección de (n-1) semiplanos. Sin embargo,

cada polígono se puede construir totalmente usando en promedio a sólo seis de ellos. El problema obvio es saber de antemano cuales van a ser los semiplanos que se deben considerar para cada punto, y es aquí donde se utiliza la triangulación de Delaunay previamente calculada.

Como a cada lado de la triangulación le corresponde un solo lado del diagrama de Voronoi y viceversa, tenemos que fijarnos en cuales son las líneas de Delaunay que tienen como a uno de sus extremos al punto al que se le desea construir su polígono; pero esto no es más que fijarse en los puntos que conforman su lista de adyacencia. Así, fácilmente, se sabe que semiplanos van a dar origen a las paredes del polígono, y como el diagrama no es más que la unión de estos entonces se tiene ya un método para su elaboración a partir de su dual.

c) Otros Métodos

En este momento podría pensarse que el problema de la elaboración de las teselaciones de Voronoi y de Delaunay está ya óptimamente resuelto. Después de todo, usamos un buen algoritmo para construir la triangulación y el diagrama de Voronoi entonces resulta una consecuencia casi inmediata. Hay, sin embargo, algunas cuestiones que no se han tratado todavía. Por ejemplo, que pasaría si ya que tenemos alguna de las teselaciones deseamos de repente añadir un punto al conjunto de datos. Por la forma en la que se operó, esto significaría que habría que desechar la teselación y volver a construirla desde el principio, es decir que no habría forma de aprovechar la información anterior. En algunas aplicaciones esto puede ser un grave inconveniente, ya que si se espera que dichas adiciones se hagan constantemente entonces es obvio que debería pensarse en un algoritmo de otro tipo que consideráse esto.

En esta sección, vamos precisamente a ver algunos algoritmos para construir tanto la triangulación como el diagrama de Voronoi, pero con otros métodos y bajo otros enfoques.

Primero, vamos a revisar un algoritmo para obtener una triangulación, basado en una idea propuesta por Lawson [6,12,13]. Este, a diferencia del usado en este trabajo, es iterativo.

- ALGORITMO DE TRIANGULACION POR INTERCAMBIO
- paso 1 : Dado un conjunto de n puntos dentro de un rectángulo, remover cualquier punto que caiga en los vértices del rectángulo.
- paso 2 : Partir el rectángulo en aproximadamente $n^{1/2}$ "cajones" (regiones rectangulares más pequeñas).
- paso 3 : Reordenar los puntos por cajones empezando en algún cajón y procediendo hacia cajones vecinos.

- paso 4 : Poner el primer punto en el rectángulo. Conectarlo con las cuatro esquinas del rectángulo para producir una triangulación inicial.
- paso 5 : Añadir el siguiente punto a la triangulación existente. Conectar este punto con los vértices del triángulo que lo circunda.
- paso 6 : El paso 5 puede producir hasta cuatro cuadriláteros estrictamente convexos (cuatro cuadriláteros sólo ocurren cuando un nuevo punto introducido cae en un lado de la triangulación). Cada uno de estos cuadriláteros tiene una diagonal alterna. Intercambiar una diagonal con su alterna, si eso se requiere para satisfacer el criterio del ángulo max-min dentro del cuadrilátero (es decir, usar el POL dentro de éste).
- paso 7 : Cada intercambio realizado en el paso 6 puede resultar en dos nuevos cuadriláteros que necesitan ser examinados. Si uno de estos cuadriláteros no satisface el criterio del ángulo max-min, intercambiar su diagonal con su alterna.
- paso 8 : Este proceso de intercambio se puede propagar hacia afuera. Lawson ha demostrado que este proceso siempre termina.
- paso 9 : Si todos los puntos de V han sido usados, entonces terminar; de otro modo ir al paso 5.

En este algoritmo lo primero que llama la atención es que el conjunto debe estar en el interior de un rectángulo. La finalidad de esto es conocer de antemano la envoltura convexa del conjunto, dentro de la cual sabemos se forma toda la triangulación y, así, evitar perder tiempo teniendo que calcularla. Si el conjunto original no contiene los vértices de dicho rectángulo estos se añaden implícitamente. Después, como parte de la "inicialización" viene el proceso de la reordenación por cajones y lo de la triangulación inicial.

La parte o ciclo principal del algoritmo viene a partir del paso 5. Se van a adiendo los puntos de uno en uno y se va actualizando la teselación cada vez. En toda esta parte, lo único que tal vez necesitáse una explicación más detallada es lo de que se generan cuadriláteros. Tenemos, que al unir un nuevo punto con los vértices del triángulo que lo circunda, ésto nos genera tres o cuatro lados (de los cuales puede ser que dos de ellos ya estuviéran en la triangulación) (ver Figura 2.3.1). De estos, nos fijamos en todo par de lados adyacentes, ya que cada par va a estar unido por un lado de la triangulación, el cual si no está sobre la envoltura convexa del conjunto se interpreta como la diagonal del cuadrilátero que se ha generado. Aquí cabe hacer la aclaración de que ese cuadrilátero podría no ser estrictamente convexo, en cuyo caso no habría posibilidad de intercambio de diagonal.

Este algoritmo a costa de ser mucho más simple de comprender y de implementar tiene un tiempo de corrida no óptimo del $O(n^2)$ por lo que sólo se recomienda para conjuntos "relativamente pequeños". Su utilidad radica en que con algunas pequeñas modificaciones podemos obtener un procedimiento muy apropiado para cuando se necesita añadir un punto a una triangulación cualquiera.

Si lo que se desea actualizar no es la triangulación, sino el diagrama de Voronoi, podemos usar otro algoritmo también iterativo y que se debe a Green y Sibson [14,15].

Supóngase que ya tenemos el diagrama de un conjunto inicial de puntos p_i's y que se desea agregar un nuevo punto p_q. Lo que se debe hacer es:

- paso 1 : Determinar el polígono de Voronoi que contiene a p_q. Sea este $DV(p_a)$.
- paso 2 : Construir el bisector perpendicular del segmento de p_a

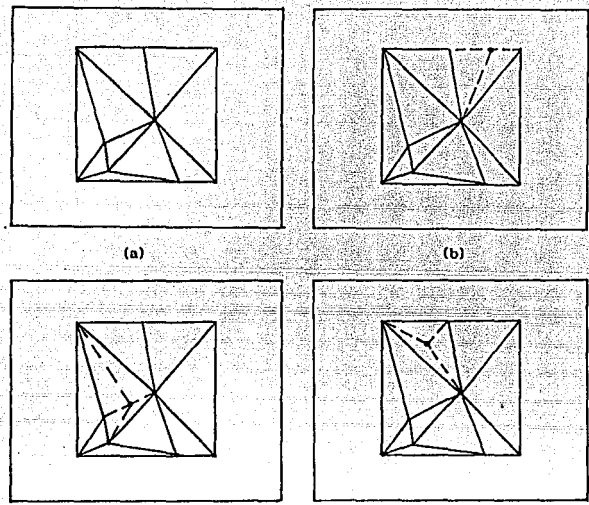


Figura 2.3.1 (a) Triangulación original; (b) Adición de un punto generándose un lado nuevo; (c) Adición con dos lados nuevos; (d) Adición con tres lados nuevos.

a p_q y hallar su intersección en el sentido de las manecillas del reloj con el polígono $DV(p_a)$. Este punto es un nuevo vértice de la teselación modificada, es decir del polígono $DV(p_q)$.

- paso 3 : Determinar el polígono adyacente $DV(p_b)$ y por lo tanto el punto p_b (ver Figura 2.3.2).

- paso 4 : Repetir los pasos 2 y 3 en el sentido de las manecillas del reloj hasta que el nuevo polígono adyacente sea el polígono inicial $DV(p_a)$.
- paso 5 : Remover todo lado y vértice de la antigua teselación, dentro del nuevo polígono $DV(p_q)$.

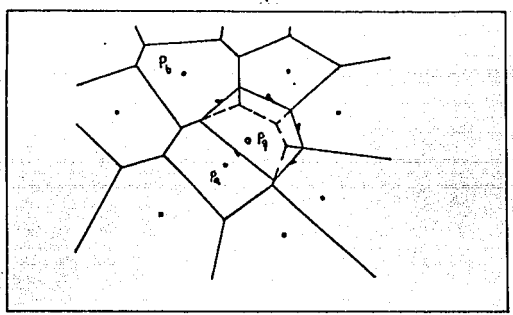


Figura 2.3.2 Forma de añadir un punto en el método iterativo.

Obviamente, para que este algoritmo sea práctico, la estructura de datos que se vaya a usar debe contener la suficiente información en forma de apuntadores, listas de vértices, etc, que permita que los polígonos vecinos se puedan accesar de un modo eficiente al igual que se puedan encontrar los lados de los polígonos. Sin embargo, aún con estas disposiciones, el algoritmo (en el peor de los casos) "corre" en un tiempo del orden de $O(n^2)$.

Finalmente, presentaremos otro algoritmo para la construcción del diagrama de Voronoi pero que tiene la ventaja, sobre el anterior, de ser óptimo. Dicho algoritmo es recursivo y fué propuesto por Shamos y Hoey [1], [2].

- ALGORITMO RECURSIVO PARA EL DIAGRAMA DE VORONOI
- paso 1: Dividir el conjunto de puntos P en dos subconjuntos P_I y P_D, aproximadamente del mismo tamaño, de acuerdo a la mediana de las coordenadas en x (esto es para asegurarnos que P_I y P_D estén linealmente separados).
- paso 2: Construir $DV(P_I)$ y $DV(P_D)$ recursivamente.
- paso 3: Construir la "cadena poligonal σ", que separa P_I y P_D.
- paso 4: Descartar todos los lados de $DV(P_D)$ que caigan a la izquierda de σ y todos los lados de $DV(P_I)$ que caigan a la derecha de σ. El resultado es DV(P), el diagrama de Voronoi del conjunto entero.

Los pasos 1 y 2 no necesitan mayor explicación, sobre todo, por que algo análogo se realizó en el algoritmo iterativo de Lee y Schachter discutido anteriormente. La tarea más complicada se presenta en el paso 3 al tratar de obtener la llamada cadena poligonal o cadena divisora. Sin definirla formalmente diremos que es una secuencia "ininterrumpida" de lados, que son compartidos por pares de polígonos $DV(p_i)$ y $DV(p_j)$ de DV(P), para $p_i \in P_I$ y $p_j \in P_D$.

Los extremos de dicha secuencia son rayos semi-infinitos que se pueden obtener sin mayor dificultad. Recordemos el concepto de tangente común inferior y superior que se utilizó para unir dos envolturas convexas (ver la sección (b) de este mismo capítulo). Los extremos de σ no son más que una porción del bisector perpendicular de dichas tangentes (ver Figura 2.3.3).

Una vez que encontramos un rayo de σ, la construcción continúa, lado por lado, hasta alcanzar el otro rayo. Pensemos que los diagramas de Voronoi que desean unirse se encuentran como en la Figura 2.3.4 (a) y (b).

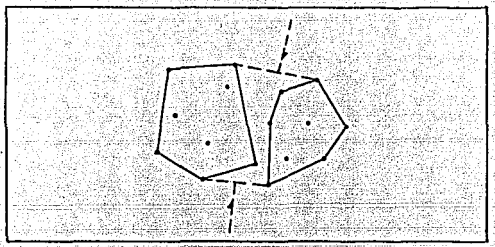
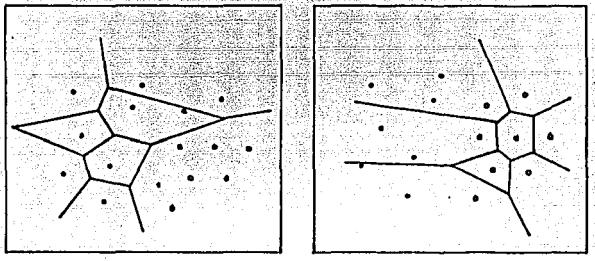


Figura 2.3.3 Encontrando los rayos de σ



(a) (b)

Figura 2.3.4 (a) El diagrama de Voronoi del conjunto izquierdo y (b) el del conjunto derecho.

Refiriéndonos a la Figura 2.3.5, donde por simplicidad el punto p_j se muestra sólo como el índice j, el rayo "superior" de σ es el bisector de los puntos 7 y 14. Imaginemos un punto z en el rayo, moviéndose hacia abajo desde el infinito. Inicialmente z se ubica en un polígono de $DV(P_I)$ y en otro de $DV(P_D)$. Esta situación se mantendrá

hasta que cruce algún lado de uno de esos polígonos, tras lo cual se empezará a mover en una dirección diferente. En nuestro ejemplo, z encuentra un lado de $DV(P_D)$ antes de "chocar" con alguno de $DV(P_I)$. Esto significa que z se encuentra ahora más cerca del punto p_{11} que del p_{14}, por lo que se empieza a mover a lo largo del bisector p_7-p_{11}. Así continúa hasta que topa con el bisector p_6-p_7 de $DV(P_I)$ lo que origina que ahora empieze a recorrer el bisector p_6-p_{11}. Eventualmente tocará el bisector p_{10}-p_{11} de $DV(p_{11})$ y procederá vía el bisector p_6-p_{10}. Este recorrido zigzagueante continuará hasta que se alcance el rayo inferior de σ (para ver una implementación de este proceso y una idea de como realizar el paso 4 ver [2]).

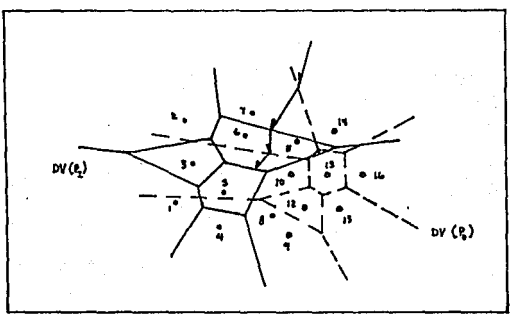


Figura 2.3.5 Un recorrido zigzagueante para obtener σ

III. Implementación del Sistema

El propósito fundamental de ésta sección es dar un panorama general de todos esos "pequeños detalles" que no se señalan cuando se presenta el algoritmo y que se supone que el implementador en el transcurso de su trabajo deberá ir descubriendo y resolviendo.

El proporcionar una descripción más precisa de la estructura que se necesita puede resultar un buen punto de partida. Se va a guardar al conjunto de puntos P en una lista (ordenada según lo establecido en el capítulo anterior). Cada nodo va a estar conformado de la siguiente forma:

- coordenadas de este punto,
- área de su correspondiente polígono de Voronoi,
- bandera que indica si pertenece a la envoltura convexa de P,
- apuntador al siguiente punto y
- apuntador a la lista de adyacencia.

Con respecto a la lista de adyacencia, ésta debe ser doblemente ligada y circular y estará conformada por nodos con los siguientes "campos":

- coordenadas de este punto,
- coordenadas del vértice de Voronoi asociado,
- ángulo que la arista correspondiente forma con la horizontal,
- apuntador al siguiente elemento de la lista de adyacencia y
- apuntador al elemento anterior de la lista de adyacencia.

Así, la estructura resultante podría verse más o menos como en la Figura 3.1.1 (la utilidad de algunos de los campos mencionados, se verá en su momento).

Una de las tareas más importantes en el algoritmo, es el poder insertar correctamente puntos en las listas de adyacencia. Para lograr esto debía pensarse en un método que se encargara de que los puntos

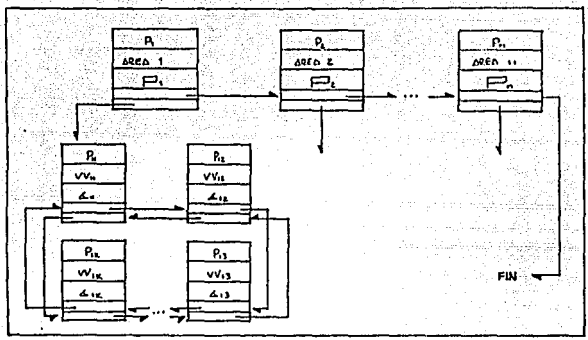


Figura 3.1.1 Estructura de datos que se utiliza en el algoritmo de Lee y Schachter

mantuvieran el mismo orden que los lados correspondientes en la triangulación, por lo que se optó por crear un orden para cada una de las listas. Supongamos que la situación alrededor de un punto p_1 es como en la Figura 3.1.2:

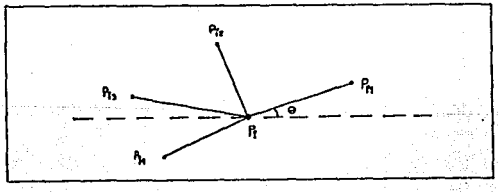


Figura 3.1.2 Forma de ordenar los puntos que pertenecen a la lista de adyacencia de p_1.

Sabemos que si los puntos p_{ij} forman lados de Delaunay con p_i entonces deben pertenecer a su lista de adyacencia. Esta debe tener un primer elemento, llamémosle p_{i1}. Nos fijamos en el ángulo que forma la recta que lo une con p_i, con respecto a la horizontal. Si denotamos a dicho ángulo como θ, entonces al ir a insertar un nuevo punto, éste deberá quedar después de todos los que tengan un menor ángulo según lo siguiente: $\theta < \theta + 1° < \ldots < 360° < 1° < 2° < \ldots < \theta - 1°$. Va a haber ocasiones en las que ese primer elemento debe modificarse, es decir que entra una nueva "cabeza" de lista. En estos casos el nuevo punto se inserta de la misma forma, nada más que a partir de ese momento el orden que se regirá para esa lista será con respecto a él.

De todo esto se concluyó que existía la necesidad de guardar el ángulo que cada lado de Delaunay formaba con la horizontal, puesto que cada vez que hubiera que insertar un nuevo punto habría que estar comparando con los ángulos de los demás lados para así ver en que posición le correspondería estar.

Otro proceso que también desempeña un papel muy importante es la función PruebaC, que es la encargada de checar si un punto está o no dentro del circuncírculo de un determinado triángulo. Los parámetros que recibe son cuatro puntos, llamémoslos H, I, J y K. Los tres primeros son los que dan origen al mencionado triángulo y el cuarto es el que se desea probar.

Una forma de construirla se basa en la siguiente idea: supongamos que los cuatro puntos son los vértices de un cuadrilátero, ordenados en el sentido contrario a las manecillas del reloj. Definimos α como el ángulo que forman JKH y β el de HIJ. Si $\alpha + \beta = 180°$ quiere decir que el punto K está sobre el circuncírculo formado por los otros tres puntos, si $\alpha + \beta < 180°$ significa que cae fuera y si es mayor que 180°

implica que está adentro (todo esto es lo que motiva precisamente el nombre de la función "la prueba del cuadrilátero").

En cuanto a como se calcula β (α sería análogo) tenemos lo siguiente. Interpretamos a J, I y H más que como simples puntos como vectores (ver Figura 3.1.3). Calculamos I-H y J-I, y sustituimos en la siguiente fórmula

$$\beta = \text{angcos}\left[\frac{(J-I)\ (I-H)}{J-I \quad I-H}\right]$$

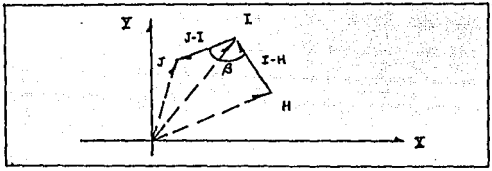


Figura 3.1.3 Cálculo del ángulo β

Sin embargo, aquí hay un problema. Debemos de estar checando que los puntos entren en el orden apropiado y de no ser así, colocarlos correctamente. Para evitar ésta tarea que puede resultar difícil, podemos usar otro método que de hecho es más intuitivo. Lo que deseamos es saber cuando un determinado punto cae dentro de un círculo, por lo que resulta razonable tratar de obtener la ecuación de éste último. Obteniendo primero su centro y después su radio podemos conseguir los resultados deseados.

En forma analítica tenemos:
sea $H = (x_1,y_1)$, $I = (x_2,y_2)$, $J = (x_3,y_3)$ y $K = (x_4,y_4)$. Por definición, el centro del circuncírculo es el punto de intersección de

las mediatrices correspondientes a los lados del triángulo HIJ. La verdad es que sólo se necesita intersectar a dos de ellas, para lo cual haremos uso de lo siguiente

punto medio de H a I = $(x_1 + x_2 / 2, y_1 + y_2 / 2) = (x_{m1}, y_{m1})$

punto medio de I a J = $(x_2 + x_3 / 2, y_2 + y_3 / 2) = (x_{m2}, y_{m2})$

pendiente de la recta ortogonal al segmento de H a I = $x_1 - x_2 / y_2 - y_1 = m_{\perp HI}$

pendiente de la recta ortogonal al segmento de I a J = $x_2 - x_3 / y_3 - y_2 = m_{\perp IJ}$

ecuación de la mediatriz del segmento HI $\quad y = y_{m1} + m_{\perp HI} (x - x_{m1})$

ecuación de la mediatriz del segmento IJ $\quad y = y_{m2} + m_{\perp IJ} (x - x_{m2})$

con lo cual procedemos a intersectar a las dos mediatrices, o lo que es lo mismo resolvemos un sistema de ecuaciones simultáneas

$$y_{m1} + m_{\perp HI}\, x - m_{\perp HI}\, x_{m1} = y_{m2} + m_{\perp IJ}\, x - m_{\perp IJ}\, x_{m2}$$

$$\Longrightarrow \quad \begin{cases} x = \dfrac{(y_{m2} - y_{m1}) + m_{\perp HI}\, x_{m1} - m_{\perp IJ}\, x_{m2}}{(m_{\perp HI} - m_{\perp IJ})} \\ y = y_{m1} + m_{\perp HI} (x - x_{m1}) \end{cases}$$

Este punto (x,y) no es más que el centro del círculo que estábamos buscando. Luego para sacar su radio hacemos

$$\text{radio} = \text{distancia (centro,H)} = \sqrt{(x - x_1)^2 + (y - y_1)^2}$$

con lo cual si K está dentro del circuncírculo, entonces se deberá cumplir que

$$(x_4 - x)^2 + (y_4 - y)^2 < \text{radio}^2.$$

Resuelto esto, podemos ocuparnos ahora de la recursión. Como el conjunto de datos se va partiendo en subconjuntos mutuamente excluyentes, hay que ver como se va a trabajar cuando se llega al nivel más bajo de recurrencia, es decir cuando únicamente quedan dos o tres puntos.

Supongamos que nos quedan tres puntos a los cuales se les tiene que empezar a construir su lista de adyacencia. Para cualquiera de los puntos sabemos que su lista va a contener a los otros dos, ya que sólo hay una posible triangulación, por lo que únicamente resta saber cual de esos puntos debe ser el primero en cada lista.

Empezamos con el punto que tenga la menor coordenada en Y y le llamamos p_0 (ver Figura 3.1.4). De los dos puntos sobrantes, el que será el primero en su lista será el que tenga el menor ángulo formado por la recta, que lo una con p_0, y la horizontal. Sea este punto p_1. Después seguirá el punto restante, digamos p_2. A su vez la lista de adyacencia del punto p_1 empezará con p_2 y después irá p_0.

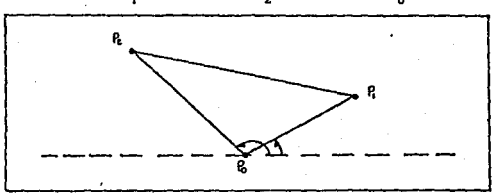


Figura 3.1.4 Construcción de una triangulación sobre tres puntos.

Para p_2 sería análogo, es decir primero p_0 seguido de p_1. Si únicamente hubieran quedado dos puntos sería mucho más sencillo, ya que cada uno sería el inicio de la lista del otro.

Una vez que se tienen dos triangulaciones, viene el proceso de juntarlas. Ya sabemos que lo primero es obtener las dos tangentes comunes que sirven como datos de entrada para la subrutina "Combinar". Hay, sin embargo, un par de detalles que si no se toman en cuenta harán fracasar todo el proceso. Al colocar la tangente común inferior nos fijamos en su extremo izquierdo. Como este punto está sobre la envoltura convexa del conjunto P_I sabemos que trae como primer elemento de su lista al punto que le sigue sobre dicha envoltura en el sentido contrario a las manecillas del reloj. Pero ahora también forma parte de la envoltura correspondiente a P, por lo que su lista deberá modificarse de tal modo que empiece ahora con el extremo derecho de la tangente. Por el contrario, para el otro extremo no hay necesidad de cambiar su "inicio" de lista, solamente se agrega el punto correcto en donde le corresponde. Algo similar ocurre al insertar la tangente común superior, salvo que aquí se invierten los papeles en cuanto a que extremo es el que va a cambiar de "inicio".

Ya con esto, el algoritmo de triangulación puede codificarse sin mayores complicaciones, por lo que podemos pasar ahora al diagrama de Voronoi.

Para su construcción iremos calculando de uno en uno cada polígono de los que lo conforman, en el mismo orden en como se encuentran los puntos después de la triangulación. La idea básica será intersectar las mediatrices correspondientes a los lados de Delaunay que tienen al mismo punto como uno de sus extremos, con lo que la zona interior resultante (que encierra al punto en cuestión) será el polígono de Voronoi de ese punto. Sin embargo, hay que hacer la distinción entre un punto que está sobre la frontera de la envoltura convexa y otro que no (por ello se añadió un campo que indica esto precisamente).

i) Punto que no está en la frontera de la envoltura convexa:

Supóngase que deseamos calcular el polígono de Voronoi de un punto p_i, o lo que es lo mismo, deseamos obtener los vértices vv_{ij} (ver la Figura 3.1.5). Vemos que el número de puntos en la lista de adyacencia (proveniente de la triangulación) de p_i es igual al número de vértices de Voronoi que hay que calcular, de donde resulta razonable que estos puntos se guarden en la misma lista. Así, según la Figura, el vértice vv_{11} se guardará en el registro correspondiente a p_{11}, vv_{12} en el de p_{12} y así sucesivamente.

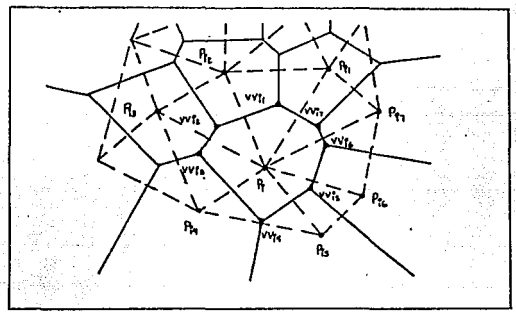


Figura 3.1.5 Cálculo de un polígono de Voronoi cerrado.

Es claro que si esto se hiciera sin mayor consideración, entonces cada vértice se calcularía tres veces (ya que tres polígonos lo comparten). Para evitar esto lo que hay que hacer es lo siguiente: pensemos que ya se calculó el vértice vv_{11} y que fué incluido en el registro de p_{11}. Vamos a la lista de adyacencia de p_{11} y colocamos ese vértice en el registro del punto p_{12}. Análogamente, se incluye también

en la lista de p_{12} en el registro de p_1; con lo cual un vértice sólo se calculará si es que no ha sido calculado previamente.

ii) Punto que está en la frontera de la envoltura convexa:

Si ahora p_1 está sobre la envoltura convexa, entonces el procedimiento será parecido, salvo que ahora como el polígono resultante no es cerrado entonces hay un vértice menos con respecto al número de puntos en la lista de adyacencia correspondiente (ver Figura 3.1.6).

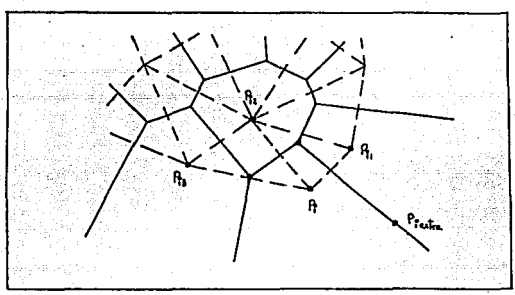


Figura 3.1.6 Cálculo de un polígono de Voronoi abierto.

Sin embargo, para poder dibujar el polígono, se necesita de un punto adicional que dará lugar a un lado de éste (al unirse con el primer punto, o sea el vv_{11}) y que se guardará en el registro al que no se le asignó un vértice. Llamémosle p_{lextra} y veamos como se obtiene: sea

$H = p_1 = (x_1, y_1)$,
$I = p_{11} = (x_2, y_2)$,
$J = vv_{11} = (x_3, y_3)$.

$p_{1\ extra} = (x,y)$ y

$m_{\perp HI}$ = pendiente de la recta ortogonal al segmento de H a I

$= x_1 - x_2 \ / \ y_2 - y_1$.

Hay que hallar las coordenadas del punto que dista "d" unidades del punto J y tal que la pendiente de la recta que lo une con él es $m_{\perp HI}$, es decir que

$$(x - x_3)^2 + (y - y_3)^2 = d^2 \qquad \text{"y"} \qquad y - y_3 \ / \ x - x_3 = m_{\perp HI} \ldots\ldots (*)$$

Haciendo las sustituciones adecuadas, obtenemos

$$(x - x_3)^2 + (m_{\perp HI}(x - x_3) + y_3) - y_3\ ^2 = d^2$$

$$\Rightarrow \quad x^2 - 2x_3 x + x_3^2 - d^2 \ / \ (1 + m_{\perp HI}^2) = 0.$$

Si $b = -2x_3$ y $c = x_3^2 - d^2 \ / \ (1 + m_{\perp HI}^2)$ entonces

$$x = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

de donde se escoge la "x" correcta y de ahí se obtiene el valor de "y" sustituyendo en (*).

Finalmente, pensando ya en las aplicaciones, se necesita calcular el área y el centro de masa de cada uno de los polígonos.

Para lo primero nos basamos en que un polígono convexo con l lados puede ser visto como la unión de l-2 triángulos, a los cuales se les puede calcular su área mediante la resolución de un sencillo determinante. Así el área de cada polígono no es más que la suma de las áreas de los triángulos que lo forman, y es una cantidad que se irá guardando en la lista principal de puntos.

En cuanto al centro de masa, éste se obtiene como un simple promedio de las coordenadas de los vértices que definen al polígono en cuestión.

IV. Conclusiones

a) Conclusiones

El propósito del presente trabajo fué el hacer una revisión tanto de los métodos de construcción del diagrama de Voronoi como de su dual; la triangulación de Delaunay y con base en esto desarrollar un sistema para computadora encargado de la elaboración de dichas estructuras.

La motivación para realizar el mencionado sistema fué que en el medio universitario en el cual nos encontramos, existen personas que utilizan, en particular, el diagrama de Voronoi, pero que no disponen de una herramienta que les permita manejarlo de forma práctica y automatizada. Ejemplo de esto son los ecólogos que necesitan de un programa que no sólo les construya ésta teselación sino que además les proporcione el área de cada uno de los polígonos que lo conforman.

Con esto en mente, es que durante la etapa de diseño del sistema, hubo la necesidad de analizar las características de los distintos algoritmos que podían emplearse. Después de sopesar sus ventajas y sus desventajas, se eligió el método recursivo de Lee y Schachter para construir la triangulación de Delaunay, a partir de la cual no es difícil obtener el diagrama de Voronoi correspondiente.

Sin embargo, no obstante el que no se hayan escogido para este trabajo en particular, se presentan brevemente aquellos métodos alternativos que se hallan en la literatura científica correspondiente. Es así que primero encontramos la versión iterativa según Lawson para obtener la mencionada triangulación y después vienen dos de los algoritmos disponibles para obtener de forma directa la teselación de Voronoi, uno que es iterativo (por Green y Sibson) y otro que es recursivo (por Shamos y Hoey).

Comparando los algoritmos iterativos con los recursivos, no fué difícil el concluir lo siguiente: los primeros representan una propuesta idónea para cuando se desea añadir o quitar un punto de alguna de las estructuras, pero desempeñan un papel poco eficiente en términos de los tiempos de ejecución de la construcción total. Los segundos son exactamente lo opuesto, es decir, debido a que se basan en el concepto de "divide y vencerás", son pésimos a la hora de querer hacer modificaciones, pero óptimos para construir en su totalidad la estructura correspondiente.

El programa resultante, tratando de ser un sistema "amigable", incluye un medio ambiente gráfico con el cual se puede realizar la comunicación hombre-máquina de manera directa. Esta interfaz para el usuario se basa en un sistema de menús, cuyas opciones se accesan usando las teclas de control (como las flechas). Toda la codificación y desarrollo del programa se efectuó en Turbo Pascal (versión 5.5) y los listados así como el programa ejecutable se encuentran a disposición del público en general y en especial de las personas interesadas en el rubro de las aplicaciones mencionadas.

b) Descripción del Sistema para el Usuario

El sistema realizado, con los comentarios pertinentes, se encuentra en el archivo llamado "VORONOI.PAS". Si se desea ejecutar ese programa habría que compilarlo primero y después "correrlo" siguiendo los pasos pertinentes según la versión del compilador de Turbo Pascal con que se cuente (de preferencia la 5.5). Si se cuenta con el programa ejecutable "VORONOI.EXE", basta llamarlo con dicho nombre (sin el punto y la extensión).

Una vez cargado el programa, aparece un sistema de menús en el cual uno se puede mover usando las flechas del teclado (←,→). La forma de seleccionar una cierta opción es posicionando el cursor en la elección deseada y presionando ya sea <ENTER> o bien la flecha hacia abajo (↓). Hecho esto, aparece inmediatamente otro menú con las distintas tareas que se pueden efectuar (ver la Figura 4.2.1). Para escoger una de tales tareas uno se vuelve a mover con las flechas (↑,↓) y presiona <ENTER> en la opción correcta. Si después de "abrir" uno de estos menús verticales uno desea volver al menú principal sin escoger ninguna opción sólo hay que oprimir <ESC>.

Existen, adicionalmente, cuatro de éstas opciones que a su vez dan origen a un segundo nivel de menú vertical (como en la Figura 4.2.2). Estos funcionan de la misma manera que los otros menús verticales.

El sistema opera, básicamente, de la siguiente forma: se guarda una serie de conjuntos de puntos o listas en memoria. A cada una de estas listas se le identificará por medio de un nombre de hasta ocho caracteres. Generalmente estas listas se "leerán" de algún archivo de tipo texto que el mismo usuario proporciona, sin embargo, se verá más adelante que no es ésta la única forma de "crear" una lista.

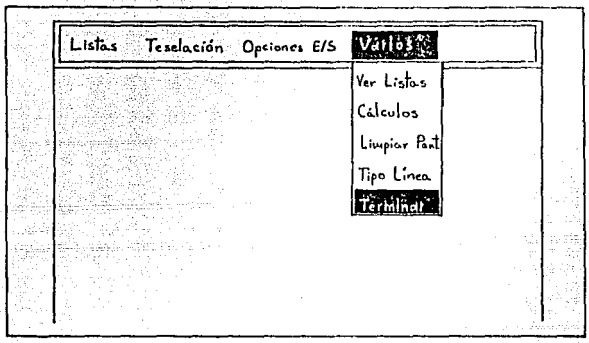


Figura 4.2.1 Un menú vertical

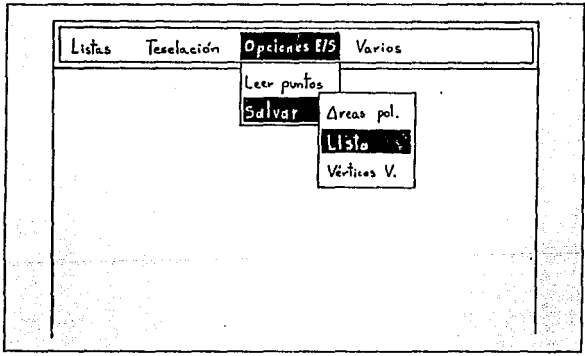


Figura 4.2.2 Un segundo nivel de menú vertical

A continuación se proporciona una descripción de las opciones disponibles en el programa.

MENU DE LISTAS

Pintar.- Aquí se dibujan, en la pantalla, los puntos que conforman una de las listas que se tengan en memoria. Para ello, se le pregunta al usuario con que nombre fué guardada (ver la opción de "leer puntos"), (ver la Figura 4.2.3).

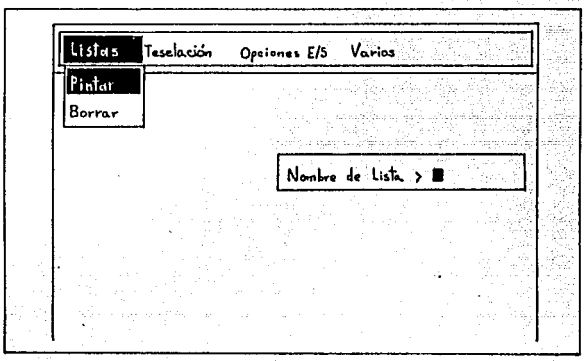


Figura 4.2.3 Petición de un nombre de lista

Borrar.- Esta opción sirve, como su nombre lo indica, para borrar toda una lista de las que se encuentran cargadas en memoria. Una vez borrada no hay forma alguna de recuperarla (a menos que se haya "salvado" previamente).

MENU DE TESELACIONES

Delaunay.- Construye la triangulación de Delaunay de una lista y la

muestra en la pantalla. Si la triangulación ya había sido calculada antes ésta opción sólo la muestra.
Voronoi.- Análoga a la anterior salvo que aquí se construye el diagrama de Voronoi. Para la construcción de este último se necesita la triangulación de Delaunay por lo que si ésta no se ha calculado, ésta opción se encarga de efectuarla.

MENU DE OPCIONES E/S
Leer Puntos.- Se usa para cuando se desea leer alguna lista. Los puntos que la conforman deben estar guardados en un archivo de tipo texto. El programa pregunta cual es el nombre de ese archivo y le asigna a la lista correspondiente el mismo nombre (siempre que no haya existido algún error en el proceso de lectura). A partir de ese momento la lista queda "cargada" en memoria. Cabe hacer notar que hay un número máximo de listas que se pueden guardar al mismo tiempo.
Salvar.- Esta opción da origen a otro menú vertical que permite seleccionar que elementos se desean salvar (en disco) en un archivo que el usuario designará. Las tres posibilidades que existen son: las áreas de los polígonos de Voronoi de una lista, los puntos que la conforman o bien los vértices de Voronoi correspondientes. En el caso de la primera y tercera posibilidad es obvio que se necesita haber ya calculado el diagrama correspondiente.

MENU DE VARIOS
Ver Listas.- Esto origina que aparezca un recuadro en la esquina inferior derecha de la pantalla con los nombres de las listas que en ese momento estén cargadas en memoria. En caso de no haber ninguna, se le notifica al usuario de ello.
Cálculos.- Aquí se muestra otro menú vertical en el cual aparecen dos posibles elecciones: cálculo de las áreas de los polígonos de Voronoi correspondientes o bien de los centros de masa de los mismos. En ambos

casos se necesita haber construido el diagrama de Voronoi de la lista elegida. Si se desea se pueden salvar los centros de masa en una lista para su uso posterior.

Limpiar Pantalla.- Se usa para decidir si se desea que se borre la pantalla después de cada tarea realizada.

Tipo de Linea.- Aquí se escoge el tipo de linea que se va a utilizar a la hora de graficar las estructuras. Se puede escoger entre linea punteada o continua.

Terminar.- Se elige para poder finalizar el programa.

Apéndice 1

Este apéndice presenta las demostraciones de las propiedades más importantes tratadas durante el capitulo dos.

Sea $P = \{p_1,\ldots,p_n\}$ un conjunto de puntos en el plano tal que no existen cuatro de ellos que sean cocirculares.

Teorema 1. Cada vértice del diagrama de Voronoi es la intersección común de exactamente tres lados del diagrama.

Demostración:

De hecho, un vértice es la intersección común de un conjunto de lados. Sea l_1, l_2,..., l_k (para $k \geq 2$) la secuencia de lados, ordenados en el sentido de las manecillas del reloj, que coinciden en el vértice v (ver Figura A.1). El lado l_1 es común a los polígonos $DV(p_{i-1})$ y $DV(p_i)$ para $i = 2,\ldots,k$ y l_i es común a $DV(p_k)$ y $DV(p_1)$. Nótese que v es equidistante de p_{i-1} y p_i dado que pertenece a l_i; por otra parte, por el mismo argumento, es equidistante de p_i y p_{i+1} y así sucesivamente. Así v es equidistante de p_1, p_2,...; p_k. Pero esto significa que $p_1,\ldots,$ p_k son cocirculares, violando la suposición inicial si $k \geq 4$. Por lo tanto $k \leq 3$. Supongamos ahora que $k = 2$. Entonces l_1 es común a $DV(p_2)$ y $DV(p_1)$ e igualmente l_2: de hecho, ambos pertenecen al bisector perpendicular del segmento p_1p_2, por lo que no se intersectan en v lo que es una contradicción. ■

Esto nos indica que los vértices de Voronoi son los centros de círculos definidos por tres puntos de P. Para un vértice v, denotamos C(v) a dicho círculo.

Teorema 2. Para cada vértice v del diagrama de Voronoi de P, el círculo C(v) no contiene ningún otro punto de P.

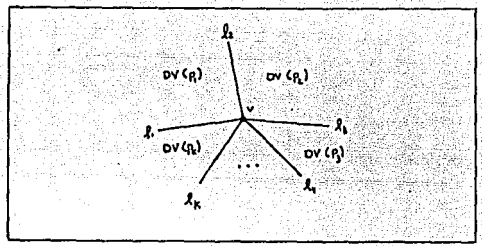


Figura A.1 Lados de Voronoi que coinciden en un vértice de Voronoi

Demostración:

Por contradicción. Refiriéndonos a la Figura A.2, sean p_1, p_2 y p_3 los tres puntos de P que determinan el círculo C(v). Si C(v) contiene algún otro punto, digamos p_4, entonces p_4 está más cerca de v que cualquiera de los otros tres puntos, en cuyo caso (por definición de un polígono de Voronoi) v debería pertenecer a $DV(p_4)$ y no a $DV(p_1)$, $DV(p_2)$ o $DV(p_3)$. Pero esto es una contradicción dado que v es común a $DV(p_1)$, $DV(p_2)$ y $DV(p_3)$. ■

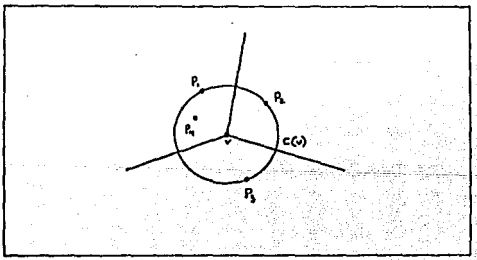


Figura A.2 El círculo C(v) no contiene otro punto de P

Lema 1. Un punto p_i no está en la frontera de la envoltura convexa de P si y sólo si cae dentro de algún triángulo cuyos vértices están en P pero no es un vértice del triángulo.

Teorema 3. El polígono $DV(p_i)$ es no acotado si y sólo si p_i es un punto que está sobre la frontera de la envoltura convexa del conjunto P.

Demostración:

Si p_i no está en la envoltura convexa de P entonces, por el lema 1, es interno a algún triángulo $p_1p_2p_3$. Consideremos los círculos C_{12}, C_{13} y C_{23} determinados por p_i y cada uno de los tres pares de vértices (p_1,p_2), (p_1,p_3) y (p_2,p_3) respectivamente (ver Figura A.3). Cada uno de estos círculos tiene un radio finito. En el círculo C_{12} (y análogamente para C_{13} y C_{23}) el arco externo A_{12} es el arco circular entre p_1 y p_2 que no contiene a p_i; de donde se concluye que cualquier punto de A_{12} está más cerca de p_1 ó p_2, que de p_i. Sea C un círculo que contenga a C_{12}, C_{13} y C_{23}. Afirmamos que cualquier punto x que caiga afuera de C está más cerca de p_1, p_2 o p_3 que de p_i. Para demostrar esto, consideremos el segmento xp_i. Por el teorema de curva de Jordan, xp_i intersecta uno de los lados del triángulo $p_1p_2p_3$, digamos p_1p_2; por lo tanto también intersecta A_{12} en un punto u. Pero como u está más cerca de p_1 ó p_2 que de p_i, se obtiene la afirmación anterior. Dado que x está más cerca de p_1, p_2 ó p_3 que de p_i, el polígono $DV(p_i)$ está contenido completamente dentro de C y por lo tanto es acotado.

A la inversa, supongamos que $DV(p_i)$ es acotado y sea l_1, l_2,..., l_k ($k \geq 3$) la secuencia de los lados que lo forman. Cada l_h ($h = 1,\ldots, k$) pertenece al bisector de un segmento $p_ip'_h$, $p'_h \in P$. Es inmediato el concluir que p_i es interno al polígono p'_1, p'_2,..., p'_k, es decir que p_i no está en la envoltura convexa de P. ■

Teorema 4. Dada una triangulación T, si una aplicación del POL a un lado l produce un intercambio con un lado l' originando una nueva triangulación T' entonces $T < T'$.

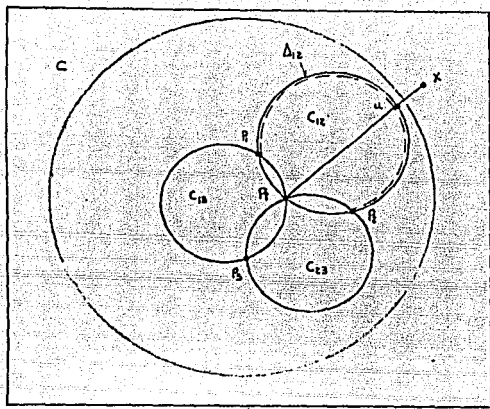


Figura A.3 Para la demostración del Teorema 3

Demostración:

Sea I el vector indicador de T (tal y como se definió en el texto). La medida de los ángulos más pequeños en los dos triángulos de T, que comparten el lado l, son dos de las componentes de I, digamos I_j e I_k con j < k, con lo cual $I_j \leq I_k$. Dado que se realizó un intercambio al aplicar el POL, el ángulo más pequeño de los dos nuevos triángulos de T' que comparten el lado l' debe ser estrictamente menor que el indicador l' de T' y por lo tanto T < T'. ■

Teorema 5. Todos los lados internos de una triangulación T de un conjunto finito P son localmente óptimos si y sólo si ningún punto de P cae dentro del circuncírculo de un triángulo de T.

Demostración:

Si ningún punto de P es interior al circuncírculo de

cualquier triángulo de T, entonces la aplicación del POL a cualquier lado no producirá un intercambio. Así todos los lados son localmente óptimos. Si todos los lados son localmente óptimos, entonces demostraremos que no hay un punto de P que sea interior al circuncírculo de cualquier triángulo. Supongamos que el circuncírculo K del triángulo Δabc contiene un punto p de P. Sea δ la distancia de p al lado más cercano, digamos (a,c) como se muestra en la Figura A.4. Pensemos que de entre todos los triángulos de T cuyos circuncírculos contienen a p como punto interior, ninguno tiene un lado que este a una distancia menor que δ de p. Dado que p está en el lado opuesto de (a,c) con respecto a "b", el lado (a,c) no está en la frontera de T. Por lo tanto, existe otro triángulo Δacq que comparte un lado con Δabc. El vértice q no puede ser interior al círculo K ya que estaría contradiciendo la hipótesis de que el lado (a,c) es localmente óptimo. El vértice q no puede estar en la región sombreada del diagrama, o Δacq contendría a p en su interior. Supongamos que el lado (c,q) es el lado más cercano de Δacq a "p". Nótese que la distancia de (c,q) a p es menor que δ. Dado que el circuncírculo de Δacq también contiene a p en su interior, tenemos una contradicción con respecto a que Δabc es el triángulo con un lado a la distancia más corta de p. ■

Lema 2. El triángulo $p_i p_j p_k$ es un triángulo de Delaunay de DT(P) si y sólo si su circuncírculo no contiene otro punto de P en su interior.

Teorema 6. Una triangulación T(P) es una triangulación de Delaunay si y sólo si su vector indicador es lexicográficamente máximo, es decir, ninguna triangulación le sigue en el ordenamiento lineal.

Demostración:

Si la triangulación T es lexicográficamente máxima, entonces todos los lados de T deben ser localmente óptimos, lo que implica que no hay un circuncírculo de algún triángulo que contenga un punto de P

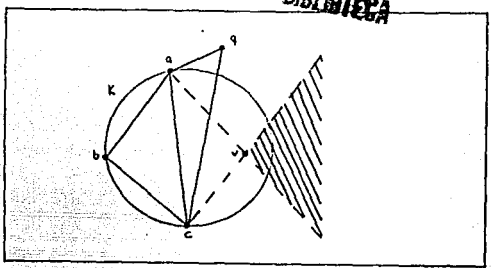


Figura A.4 Ilustración para la prueba del teorema 5

en su interior (Teorema 5). Así T es la triangulación de Delaunay, DT(P). Para probar a la inversa, supongamos que la triangulación de Delaunay no es máxima en el ordenamiento lineal. Esto significaría que existe otra triangulación T(P) tal que DT(P) < T(P). Aplicando repetidamente el POL a T(P) obtendríamos una triangulación T'(P) que tuviera todos sus lados localmente óptimos. Dado que DT(P) < T(P) < T'(P), T'(P) también sería una triangulación de Delaunay (por el Teorema 5 y el Lema 2). Sin embargo, como la triangulación de Delaunay es única, T'(P) = DT(P) lo que es una contradicción. ■

Bibliografia

[1]. M.I. Shamos and D. Hoey. Closest-point problems. Proc. 16th Annual IEEE Sympos. on Foundations of Computer Science (1975), 151-162.

[2]. Computational Geometry. M.I. Shamos and F.P. Preparata. Springer-Verlag. New York (1985).

[3]. Computational Geometry. M.I. Shamos. Ph.D. Thesis, Dept. Comput. Sci., Yale Univ., New Haven, Conn. (1978).

[4]. D.T. Lee and C.K. Wong. Voronoi diagrams in L_1 (L_∞) metrics with two-dimensional storage applications. *SIAM J. Comp.* 9 (1980), 200-211.

[5]. D.T. Lee. Two-dimensional Voronoi diagrams in the L_p metric. *J. ACM* 27 (1980), 604-618.

[6]. Pattern Models. N. Ahuja and B.J. Schachter. John Wiley and Sons. New York (1983).

[7]. R.A. Jarvis and E.A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Trans. Comput.* C-22 (1973), 1025-1034.

[8]. Principles of Forest Hidrology. J.D. Hewlett. The University of Georgia Press. Athens (1982).

[9]. Ecology. M. Begon, L.L. Harper and C.R. Townsend. Blackwell Scientific Publications. Oxford(1986).

[10]. R. Mithen, J.L. Harper and J. Weiner. Growth and mortality of individual plants as a function of "available area". *Oecologia* 62 (1984), 57-60.

[11]. D.T. Lee and B.J. Schachter. Two algorithms for constructing a Delaunay triangulation. *Int. J. Computer and Information Sciences* 9 (1980), 219-242.

[12]. C.L. Lawson. Generation of a triangular grid with applications to contour plotting. Tech. Memo. 299, Jet Propulsion Lab., Pasadena, California (1972).

[13]. C.L. Lawson. Software for C^1 surface interpolation. En: Mathematical Software III, ed. J. Rice. (Academic Press, 1977).

[14]. A.R Forrest. Recent work on geometric algorithms. En: Mathematical Methods in Computer Graphics and Design, ed. K.W. Brodlie (Academic Press, London, 1980).

[15]. P.J. Green and R. Sibson. Computing Dirichlet tessellations in the plane. *Computer J.* 21(2) (1978), 168-173.

[16]. J.F. O'Callaghan. Computing the perceptual boundaries of dot patterns. *Comput. Graphics Image Processing* 3 (1974), 141-162.

[17]. N. Rivier, R. Occelli, J. Pantaloni and A. Lissowski. Discrete simulation of phyllotaxis. En: Dynamical Systems and Cellular Automata, eds. J. Demongeot, E. Goles and M. Tchuente. (Academic Press, London 1985).

[18]. N. Rivier, R. Occelli, J. Pantaloni and A. Lissowski. *J. de Physique* 45 (1984).