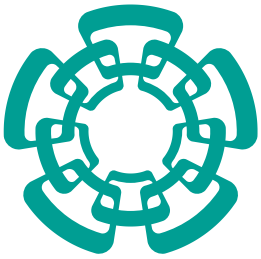

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE MATEMÁTICAS

“Duplicando conjuntos en posición general”

T E S I S

Que presenta

ÓSCAR EDUARDO GARCÍA QUINTERO

Para obtener el grado de

MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE MATEMÁTICAS

Director de la Tesis:

Dr. Ruy Fabila Monroy

Ciudad de México ENERO, 2017

Para
Juan Miguel y Yuliana

Agradecimientos

Quiero darle mis más sinceros agradecimientos a todas aquellas entidades y personas que me brindaron su ayuda de manera incondicional para la realización de esta tesis: Gracias a CONACyT por el apoyo económico que me proporcionó para poder estar en México y al Departamento de Matemáticas del Cinvestav por darme la oportunidad de ser un Maestro en Ciencias. A mi asesor el Doctor Ruy Fabila Monroy quien me impulsó con constante entusiasmo, por ofrecerme su amistad y darme a conocer con sus colegas. Tanto a mi asesor como al Doctor Feliú Sagols, les estoy muy agradecido por los cursos que impartieron durante el periodo de mi Maestría, tanto de Matemáticas como de Ciencias de la Computación, los cuales fueron de gran utilidad para el desarrollo de este trabajo. Esta tesis no se pudo lograr sin la ayuda de la familia Hincapié y mi familia. En especial, muchas gracias a mi amada Yuliana María Hincapié por su grata compañía y apoyo incondicional. Por último, pero no menos importante, le agradezco a Dios por darme la fortaleza de estar fuera de mi país durante todo este tiempo.

Índice general

Resumen

Sea $\overline{cr}(K_n)$ el mínimo número de cruce sobre todos los dibujos rectilíneos de la gráfica completa con n vértices en el plano. En esta tesis se presenta una implementación para construir los puntos de un dibujo rectilíneo de K_{2n} a partir de la duplicación de los puntos de un dibujo rectilíneo de K_n, de tal forma que el número de intersecciones de aristas de K_{2n} dependa solo del dibujo de K_n. De esta forma, se ofrece una herramienta útil para el cálculo de cotas superiores de la constante del problema del número de cruce $q_* = \lim_{n\to\infty} \overline{cr}(K_n)/\binom{n}{4}$.

Abstract

Let $\overline{cr}(K_n)$ be the minimum number of crossings over all rectilinear drawings of the complete graph on n vertices in the plane. In this thesis we present an implementation to construct the points of a rectilinear drawing of K_{2n} from the duplication of the points of a rectilinear drawing of K_n, such that the number of intersections of edges in K_{2n} depend only on the drawing of K_n. In this way, a useful tool is provided for calculating upper bounds of de crossing number problem constant $q_* = \lim_{n\to\infty} \overline{cr}(K_n)/\binom{n}{4}$.

Introducción

El problema de minimizar las intersecciones de aristas de una gráfica G dibujada en el plano es un problema clásico de Geometría Combinatoria. Este problema se presenta constantemente en toda clase de redes de comunicación, ya sea el Internet, rutas aéreas u otro tipo de transporte, circuitos eléctricos, redes telefónicas, entre otras. Todos estos ejemplos tienen algo en común, las intersecciones que se presentan implican un costo adicional, ya que se debe asegurar que lo enviado de un punto A a otro B, llegue de forma exitosa y nunca se salga del camino que comunica a A con B. Por tal motivo se desea que la cantidad de tales cruces sea mínima. Cuando cada punto involucrado en la red está conectado con todos los demás directamente (una gráfica completa), este problema de minimización se conoce como el *problema del número de cruce*.

Este tipo de problema aparece en la literatura desde 1977 en [Tur77]. Y durante el paso de los años, el resolver el *problema del número de cruce rectilíneo* para gráficas con órdenes cada vez mas grandes, ha sido de gran interés. La *constante para el problema del número de cruce* $q_* = \lim_{n\to\infty} \overline{cr}(K_n)/\binom{n}{4}$, resulta ser entonces el número que se desea aproximar con la mayor precisión posible. Una de las mejoras más sobresalientes para el cálculo de la cota inferior de q_* que se ha logrado, se encuentra en el artículo [LVWW04], usando *k-aristas* y *secuencias circulares*. Con respecto a la cota superior de q_* se continúa realizando estudios constantemente para mejorarla.

Esta tesis está dividida en cuatro capítulos. En el Capítulo 1 se introduce la Teoría de Gráficas necesaria, se da un breve recorrido histórico sobre el *problema del número de cruce*, y se hace un pequeño resumen de cómo en el artículo

[LVWW04], se calcula un muy buena cota inferior de q_*. En el Capítulo 2 se presenta la construcción matemática desarrollada por Ábrego y sus colaboradores en [ÁCFM$^+$10], el cual consiste básicamente en que a cada vértice de una gráfica completa K_m dibujada en el plano, es sustituido por un *cúmulo* de puntos, de tal forma que el dibujo K_n que se genera sea tal que su *número de cruce rectilíneo* dependa solo del dibujo de K_m y otros elementos previamente definidos. Se garantiza que la duplicación de una gráfica completa de orden impar depende únicamente de tal gráfica completa y se describe cómo se calcula actualmente la cota superior de q_*. En el Capítulo 3 se presenta una implementación de tal construcción, solo en el caso de duplicar una gráfica completa en coordenadas enteras, el cual es el objetivo principal de esta tesis. Finalmente en el Capítulo 4 se muestran algunas conclusiones sobre la utilidad que tiene la implementación, mostrándose una nueva mejora para la cota superior de q_* y se hacen algunas observaciones de cuanto puede tardar la ejecución de los algoritmos implementados, cuanto tamaño de cuadrícula puede ocupar las gráficas que resultan de las duplicaciones, y se presentan todos los conjuntos de puntos a partir de tres puntos que se pueden construir usando un computador de no muy alto rendimiento.

Capítulo 1

Número de Cruce Rectilíneo

El *problema del número de cruce* inicialmente fue planteado por Paul Turán en 1977 en [Tur77], contando como anécdota que en 1944, durante la Segunda Guerra Mundial muy cerca de Budapest, estaba trabajando en una fábrica de ladrillos, transportando los ladrillos salidos del horno a las bodegas por medio de un sistema de rieles en pequeños vagones de carga. Decía que no había ninguna dificultad en ese trabajo, salvo cuando pasaba el vagón de carga por los cruces de los rieles, lo cual era un dolor de cabeza debido a que el vagón saltaba y los ladrillos se caían al suelo haciéndose una total perdida de tiempo, llegando a la conclusión de que este problema se podría minimizar si se minimiza la cantidad de cruces de rieles.

Antes de comenzar con el planteamiento del *problema del número de cruce*, es necesario hacer una pequeña introducción de Teoría de Gráficas.

1.1. Un poco de Teoría de Gráficas

Una gráfica está constituida por dos cosas, un conjunto de vértices y un conjunto de aristas, el cual es una relación binaria entre los vértices. Gráficamente en el plano se puede considerar a los vértices como puntos y las aristas pueden representarse como segmentos que une a los puntos. Formalmente corresponde

a la siguiente definición.

Definición 1.1. *Una **gráfica** es una dupla $G = (V, E)$ tal que V es un conjunto llamado conjunto de **vértices** y E es una relación*

$$E \subseteq \{\{u, v\} \mid u, v \in V \wedge u \neq v\}$$

*llamado conjunto de **aristas**. El cardinal de V es llamado **orden** de G, y el cardinal de E es llamado **tamaño** de G.*

En ocasiones se considera una gráfica G sin necesidad de indicar al conjunto de vértices y de aristas, para hacer referencia a ellos, el conjunto de vértices es denotado como $V(G)$ y el conjunto de aristas es denotado como $E(G)$. De la definición anterior se tiene que una arista en G es algún par no ordenado de vértices $\{u, v\}$, lo cual será denotado por uv. Se dice que un par de vértices u y v en G son **adyacentes** si uv es una arista de G. Además, si todos los vértices de una gráfica de orden n son adyacentes, entonces es llamada **gráfica completa** y se denota como K_n. Por otro lado se dice que G es una **gráfica bipartita**, si existen un par de conjuntos disjuntos X y Y, tales que $V(G) = X \cup Y$ y no hay aristas entre los vértices de X ni entre los vértices de Y. Una **gráfica dirigida** (o **digráfica**) es una dupla $G = (V, E)$, solo que a diferencia de una gráfica, el conjunto de aristas E está conformado por pares ordenados de la forma (u, v) llamados **arcos**, en vez de dobletes[1] $\{u, v\}$.

El dibujar una gráfica es una actividad frecuentemente realizada en la teoría de gráficas, consiste básicamente en trazar puntos y curvas que unen a los puntos, sin embargo hay ciertas leyes que se deben respetar para que tal dibujo preserve siempre la estructura de la gráfica de tal manera de que no sea mal interpretada.

Definición 1.2. *Sea $G = (V, E)$ una gráfica. Un **dibujo** de la gráfica G es un mapeo φ de G en el plano tal que φ es inyectivo con respecto a los vértices de G enviándolos a puntos en R^2 y para toda arista $uv \in E$ se tiene que $\varphi(uv)$ es una curva (suave) que une a $\varphi(u)$ y $\varphi(v)$, de tal modo que no contenga a $\varphi(w)$ para todo otro $w \in V$, y $\varphi(uv) \cap \varphi(wx)$ sea finito para toda otra arista $wx \in E$. Se dice además que un dibujo*

[1] Nótese que $(u, v) \neq (v, u)$ mientras que $\{u, v\} = \{v, u\}$.

de G es un ***dibujo rectilíneo*** *(o* ***geométrico****) si todas las curvas que se mapean en φ son segmentos de líneas rectas.*

Definición 1.3. *Dado un conjunto de puntos P en el plano, se dice que los puntos del conjunto P están en* ***posición general****, si ninguna terna de puntos son colineales.*

Un atributo interesante que tienen todos los dibujos rectilíneos de gráficas completas es que no hay tres puntos colineales en tales dibujos, de lo contrario existirían aristas que contendrían vértices diferentes de sus extremos.

Lema 1.1. *Para todo entero positivo n, todo dibujo rectilíneo de K_n tiene sus puntos en posición general.*

A continuación se definirá formalmente lo que es el *número de cruce rectilíneo*, lo cual permitirá mas adelante hacer el planteamiento del problema principal de esta tesis.

Definición 1.4. *Para todo conjunto de puntos en posición general P, el* ***número de cruce rectilíneo*** *de P es el numero de intersecciones de pares de segmentos de líneas rectas que son definidas por todos los pares de puntos en P y es denotado por $\overline{cr}(P)$. Para toda gráfica G, el* ***número de cruce rectilíneo*** *de G es el mínimo numero de intersecciones de aristas que hay entre todos los dibujos rectilíneos de G y es denotado por $\overline{cr}(G)$.*

Nótese que por la definición anterior y considerando a $\mathscr{P}(G)$ como el conjunto de todos los conjuntos de puntos en posición general con el mismo orden que G, se tiene que

$$\overline{cr}(G) = \min\{\overline{cr}(P) \mid P \in \mathscr{P}(G)\}.$$

1.2. El Problema del Número de Cruce Rectilíneo

Turán planteó el *problema del número de cruce* en general para gráficas bipartitas con n vértices representando a los hornos y m vértices representando a las bodegas. Sin embargo el problema se puede formular para cualquier gráfica en

general. No obstante, el objeto de interés para esta tesis es el número de cruce rectilíneo para las gráficas completas.

Problema 1.1 (Problema del número de cruce rectilíneo)**.** *Para todo entero positivo n, encontrar el valor de* $\overline{cr}(K_n)$.

En el estudio que abarca la Geometría Combinatoria, existe un parámetro que coincide con el parámetro de número de cruce rectilíneo. Dado un conjunto de puntos P en posición general, el número que hay de cuádruplas de puntos en P cuya envoltura convexa[2] forma un cuadrilátero es llamado **número de cuadriláteros convexos**.

Nótese que la cantidad de intersecciones de pares de segmentos de líneas rectas entre los elementos de un conjunto de puntos, coincide con la cantidad de cuadriláteros convexos que se encuentran del mismo conjunto de puntos. De este modo, una versión equivalente al *problema del número de cruce rectilíneo* sería calcular para todo n el mínimo número de cuadriláteros convexos entre todas las configuraciones posibles de n puntos en posición general.

A lo largo del tiempo, el número de cruce rectilíneo es un parámetro que se ha podido relacionar con varios problemas clásicos de la Geometría Combinatoria, en donde uno muy destacado ha sido el Problema de los Cuatro Puntos de Sylvester.

Problema 1.2 (Problema de los cuatro puntos de Sylvester)**.** *Sea R un conjunto abierto y convexo en el plano y de área finita,*[3] *y sea* $q(R)$ *la probabilidad de que cuatro puntos escogidos de forma aleatoria en R forman un cuadrilátero convexo. Encontrar entre todos los conjuntos abiertos y convexos R el valor máximo y mínimo de* $q(R)$.

Originalmente Sylvester en [Cro85] planteó este problema. Luego él conjetura que en tal problema se logra alcanzar un valor máximo cuando la región es una circunferencia o una elipse, y alcanza un valor mínimo cuando la región es un triángulo. Lo cual logra demostrar Blashke en 1917 en [Bla17]. Mas tarde

[2]La envoltura convexa, en este caso, es la intersección de todos los conjuntos convexos en el plano que contienen a los cuatro puntos.

[3]Existe un disco en el origen de radio finito que lo contiene a R.

Scheinerman y Wilf en [SW94] lograron hacer una demostración de este problema en un caso mas general, sin considerar la hipótesis de que la regiones en el plano sean convexas. Nótese que sin esta última restricción la probabilidad $q(R)$ tiende a 1, y por lo tanto se maximiza, cuando la región R tiende a ser un anillo en el plano. Sin embargo la cota inferior

$$q_* \stackrel{def}{=} \text{ínf}\{q(R) \mid R \text{ abierto y acotado}\}$$

aún no ha sido hallada, pero lograron hacer una conexión directa muy interesante con el parámetro de número de cruce rectilíneo. En los siguientes dos resultados se mostrará una caracterización de q_* con el número de cruce rectilíneo.

Teorema 1.1. *El siguiente límite existe.*

$$\lim_{n\to\infty} \frac{\overline{cr}(K_n)}{\binom{n}{4}}.$$

Demostración. Sea $m < n$, considérese un dibujo rectilíneo de K_n con el mínimo número de cruce rectilíneo $\overline{cr}(K_n)$, y los conjuntos de puntos A de cardinal m del dibujo de K_n. Si se suma cada valor $\overline{cr}(A)$, entonces se ha contado por cada cruce $\binom{n-4}{m-4}$ veces, ya que cada cruce está comprendido por 4 puntos. Así

$$\overline{cr}(K_n) = \sum_{|A|=m} \overline{cr}(A) / \binom{n-4}{m-4}.$$

Como $\overline{cr}(A) \geq \overline{cr}(K_m)$, entonces

$$\begin{aligned}
\overline{cr}(K_n) &\geq \sum_{|A|=m} \overline{cr}(K_m) / \binom{n-4}{m-4} \\
&= \frac{\overline{cr}(K_m)}{\binom{n-4}{m-4}} \sum_{|A|=m} 1 \\
&= \frac{\binom{n}{m}}{\binom{n-4}{m-4}} \overline{cr}(K_m) \\
&= \frac{\binom{n}{4}}{\binom{m}{4}} \overline{cr}(K_m),
\end{aligned}$$

de este modo

$$\frac{\overline{cr}(K_n)}{\binom{n}{4}} \geq \frac{\overline{cr}(K_m)}{\binom{m}{4}}.$$

Así, se tiene que la sucesión $\{\overline{cr}(K_n)/\binom{n}{4}\}_{n\geq 1}$ es creciente. Por otro lado se tiene que $\overline{cr}(K_n)/\binom{n}{4}$ es acotado por $\overline{cr}(K_5)/\binom{5}{4} = 1/5$ y por 1, esto último se debe a que no pueden haber mas intersecciones de aristas que cuádruplas de los n puntos. Por lo tanto el límite pedido existe. ■

Teorema 1.2. *Se cumple la siguiente identidad.*

$$q_* = \lim_{n\to\infty} \frac{\overline{cr}(K_n)}{\binom{n}{4}}.$$

Demostración. Por ahora, se denota

$$\overline{cr}^* = \lim_{n\to\infty} \frac{\overline{cr}(K_n)}{\binom{n}{4}}$$

Sea R un conjunto abierto y acotado en el plano. Para todo n considérese un dibujo rectilíneo de K_n en la región R y sea los puntos $p_1, \ldots, p_n$ los dibujos resultantes de los vértices de K_n. Se define la función indicadora Y sobre todas las cuádruplas de puntos A de orden 4 en $\{p_1, \ldots, p_n\}$, tal que $Y(A) = 1$ si los cuatro puntos de A forman un cuadrilátero convexo y $Y(A) = 0$ en caso contrario. Por lo tanto, si X es la variable aleatoria que da el total de número de cruce rectilíneo del dibujo, entonces

$$X = \sum_{|A|=4} Y(A).$$

Como $\overline{cr}(K_n)$ es una cota inferior para todo número de cruce rectilíneo, entonces la media debe estar por arriba, esto es,

$$\begin{aligned} \overline{cr}(K_n) &\leq E(X) \\ &= E\left(\sum_{|A|=4} Y(A)\right) \\ &= \sum_{|A|=4} E\left(Y(A)\right) \end{aligned}$$

$$\begin{aligned} &= \sum_{|A|=4} q(R) \\ &= \binom{n}{4} q(R). \end{aligned}$$

Así se tiene que $\overline{cr}(K_n)/\binom{n}{4} \leq q(R)$ para todo n, dando como resultado que $\overline{cr}^* \leq q(R)$ para todo R. Luego por propiedad del ínfimo se tiene que $\overline{cr}^* \leq q_*$.

Para demostrarse la otra desigualdad, para cualquier región abierta y acotada R y todo n, considérese un dibujo rectilíneo óptimo de K_n, es decir, un dibujo tal que el número de cruce rectilíneo es $\overline{cr}(K_n)$. Del Lema 1.1 se tiene que los puntos del dibujo están en posición general. Así todo punto del dibujo está confinado en una región delimitada por algunos de los segmentos que une a los demás puntos del dibujo. Sea R_ε la región conformada por discos abiertos (disjuntos) de radio ε para todos los puntos del dibujo, donde ε es el mínimo tal que todos los discos están dentro de las regiones que confinan a sus centros. Sea q la probabilidad de que al escoger 4 puntos de forma aleatoria en la región R_ε forman un cuadrilátero convexo y además estos puntos están en discos distintos (esta última condición hace que $q \neq q(R_\varepsilon)$). Como los puntos escogidos están muy cerca de los puntos originales del dibujo, se tiene que

$$q = \frac{\overline{cr}(K_n)}{\binom{n}{4}}.$$

Por otro lado, nótese que la probabilidad de escoger cuatro puntos de forma aleatoria en distintos discos de R_ε es de $\left(\frac{n}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{n-3}{n}\right)$, esto es, $1 - O(1/n)$. Por lo tanto la probabilidad de que ocurra lo contrario es de $O(1/n)$. De lo anterior, la probabilidad de que cuatro puntos escogidos aleatoriamente sean tales que definan un cuadrilátero convexo y no estén en 4 discos distintos es a lo sumo $O(1/n)$, es decir, $1 - q \leq O(1/n)$. Por lo tanto se tiene que

$$\begin{aligned} q_* &\leq q(R_\varepsilon) \\ &= q + (1-q) \\ &\leq q + O\left(\frac{1}{n}\right) \end{aligned}$$

$$= \frac{\overline{cr}(K_n)}{\binom{n}{4}} + O\left(\frac{1}{n}\right),$$

para todo n. Luego haciéndose tender n a infinito, se sigue que $q_* \leq \overline{cr}^*$. ■

Del teorema anterior se puede concluir que, mientras mas preciso sea el el cálculo en el problema del número de cruce rectilíneo, mas preciso será el valor de la constante del problema de los cuatro puntos de Sylvester.

1.3. Cotas Inferiores de $\overline{cr}(K_n)$ y k-Aristas

Para el estudio de cotas inferiores para el problema del número de cruce rectilíneo, nótese que para $n = 5$ se obtiene al menos un subconjunto, de las 5 combinaciones posibles que hay, de orden 4 cuya envoltura convexa es un cuadrilátero.[4] De donde para todo $n \geq 5$, se sigue que al menos $1/5$ de los subconjuntos de orden 4 tienen como envoltura convexa un cuadrilátero. Similarmente si se tiene una cota inferior para algún $\overline{cr}(K_{n_0})$ con n_0 fijo, entonces este da una cota inferior para todo $\overline{cr}(K_n)$ con $n \geq n_0$. La mejor cota inferior probada por este método es debido a Aichholzer, Aurenhammer, y Krasser [AAK06]: ellos obtuvieron la cota inferior de $0.3115\binom{n}{4}$ por inspección de todas las configuraciones de 11 puntos.

También se ha abordado este estudio basándose en métodos diferentes, dándose como primer resultado $\frac{53-5\sqrt{13}}{216}\binom{n}{4} + O(n^3) \approx 0.3288\binom{n}{4}$, el cual fue establecido en [Wag03].

En el 2004 se halló una cota inferior del mínimo número de cruce rectilíneo que mejora por mucho a las cotas anteriores, la cual se presenta en [LVWW04]. Para presentar este resultado es necesario introducir la siguiente definición:

Definición 1.5. *Para todo conjunto de puntos, P una k-**arista** de P es un par ordenado (u, v) con $u, v \in P$ y $u \neq v$, tal que hay exactamente k puntos de P en el lado derecho de la línea dirigida que genera (u, v). Se denota como $e_k \stackrel{def}{=} e_k(P)$ al*

[4] Todo dibujo rectilíneo de K_5 en el plano siempre cumple que hay un par de aristas que se cruzan.

*número de k-aristas de P. La colección de todas las i-aristas con $i \leq k$ es llamada una $(\leq k)$-**arista**. Se denota el número de $(\leq k)$-aristas como $E_k \stackrel{def}{=} e_0 + \ldots + e_k$.*

Se denota como $\square$ al número de cuádruplas de puntos en P que definan un cuadrilátero convexo, se denota como $\triangle$ las demás cuádruplas de puntos de P, es decir, aquellas cuádruplas que no definan un cuadrilátero convexo. A continuación se presentará una identidad entre el número de cuadriláteros convexos, y por lo tanto para el número de cruce rectilíneo, con el número de k-aristas, el cual expresa a $\square$ como una combinación lineal positiva de números e_k.

Lema 1.2. *Para todo conjunto de n puntos en el plano en posición general,*

$$\square = \sum_{k<\frac{n-2}{2}} e_k \left(\frac{n-2}{2} - k\right)^2 - \frac{3}{4}\binom{n}{3}.$$

Demostración. Sea P un conjunto de n puntos en posición general. Considérese el conjunto de 4-tuplas (u,v,w,z) de puntos distintos de P tal que la recta dirigida que pasa primero por u y luego por v es tal que deja a w de lado izquierdo y a z de lado derecho. De esta forma para toda cuádrupla de puntos en P que define un cuadrilátero convexo, hay 4 formas de contarlo como 4-tuplas, y en caso contrario (es decir, que no definen un cuadrilátero convexo), hay 6 formas de contarlo como 4-tupla (ver Figura 1.1). De esta forma, la cantidad total de 4-tuplas que se pueden contar en el conjunto P es $4\square + 6\triangle$. Por otro lado, para cualquier k-arista uv hay $k(n-2-k)$ formas para completar una 4-tupla. Por lo tanto, se tiene que

$$4\square + 6\triangle = \sum_{k=0}^{n-2} e_k k(n-k-2). \tag{1.1}$$

Pero se tiene también que $\square + \triangle = \binom{n}{4}$, esto es, $\triangle = \binom{n}{4} - \square$. Luego se tiene de la Ecuación 1.1

$$4\square + 6\left(\binom{n}{4} - \square\right) = \sum_{k=0}^{n-2} e_k k(n-k-2),$$
$$6\binom{n}{4} - 2\square = \sum_{k=0}^{n-2} e_k k(n-k-2).$$

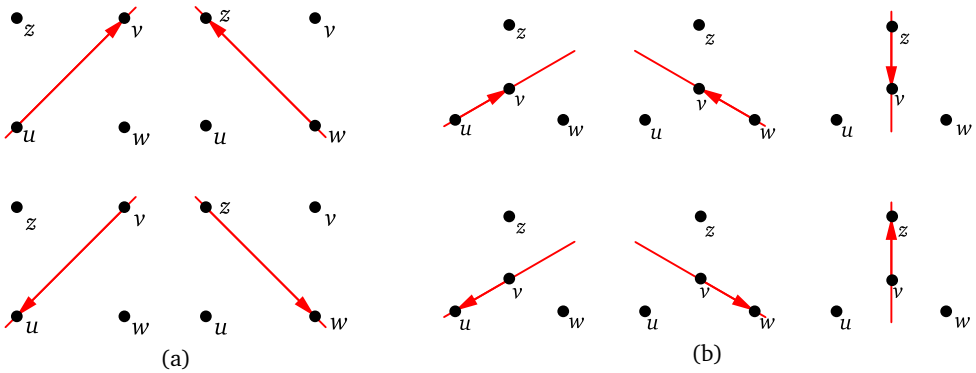

Figura 1.1: (a) Las 4-tuplas de una cuádrupla convexa.
(b) Las 4-tuplas de una cuádrupla no convexa.

Así se tiene que

$$\square = \frac{1}{2}\left(6\binom{n}{4} - \sum_{k=0}^{n-2} e_k k(n-k-2)\right). \tag{1.2}$$

Nótese que el número total de k-aristas con $0 \leq k \leq n-2$ cuenta exactamente dos veces el número de aristas que se pueden definir con el conjunto de puntos P, esto es

$$\sum_{k=0}^{n-2} e_k = n(n-1). \tag{1.3}$$

Luego se tiene fácilmente la igualdad

$$6\binom{n}{4} = \sum_{k=0}^{n-2} e_k \frac{(n-2)(n-3)}{4}.$$

Remplazando esta última ecuación en la Ecuación 1.2, se tiene

$$\square = \frac{1}{2}\sum_{k=0}^{n-2} e_k \left(\frac{(n-2)(n-3)}{4} - k(n-k-2)\right). \tag{1.4}$$

Usándose de nuevo la Ecuación 1.3 se tiene sin dificultad

$$\frac{3}{4}\binom{n}{3} = \frac{1}{2}\sum_{k=0}^{n-2} e_k \frac{n-2}{4}.$$

Ahora, con la ecuación anterior y notándose que $e_k = e_{n-2-k}$ para todo $0 \leq k \leq n-2$, se tiene

$$\begin{aligned}
\sum_{k<\frac{n-2}{2}} e_k\left(\frac{n-2}{2}-k\right)^2 - \frac{3}{4}\binom{n}{3} &= \frac{1}{2}\sum_{k=0}^{n-2} e_k\left(\frac{n-2}{2}-k\right)^2 - \frac{1}{2}\sum_{k=0}^{n-2} e_k\frac{n-2}{4} \\
&= \frac{1}{2}\sum_{k=0}^{n-2} e_k\left(\left(\frac{n-2}{2}-k\right)^2 - \frac{n-2}{4}\right) \\
&= \frac{1}{2}\sum_{k=0}^{n-2} e_k\left(\frac{n^2-4n+4}{4} - nk + 2k + k^2 - \frac{n-2}{2}\right) \\
&= \frac{1}{2}\sum_{k=0}^{n-2} e_k\left(\frac{n^2-5n+6}{4} - k(n-2-k)\right) \\
&= \frac{1}{2}\sum_{k=0}^{n-2} e_k\left(\frac{(n-2)(n-3)}{4} - k(n-k-2)\right).
\end{aligned} \tag{1.5}$$

Luego de las Ecuaciones 1.4 y 1.5, se tiene finalmente que

$$\square = \sum_{k<\frac{n-2}{2}} e_k\left(\frac{n-2}{2}-k\right)^2 - \frac{3}{4}\binom{n}{3}.$$

■

Se ha logrado expresar a $\square$ (con un término de error) como una combinación lineal positiva de números de j-aristas e_j. De este modo, si se consiguen cotas inferiores para cada e_j, entonces se puede obtener una cota inferior para $\square$.

Para derivarse una cota inferior estrecha para cada uno de los valores e_j, László Lovász y sus colaboradores en [LVWW04] presentaron el siguiente resultado.

Proposición 1.1. *Para todo conjunto de n puntos en el plano en posición general y para todo $j < \frac{n-2}{2}$,*

$$e_j < 2j + 3.$$

Y para todo $j \geq 0$ y $n \geq 2j+3$, esta cota es estrecha.

La siguiente construcción muestra que esta cota es estrecha.

Lema 1.3. *Para todo conjunto de n puntos en el plano en posición general,*

$$\square = \sum_{j<\frac{n-2}{2}} E_j\,(n-2j-3) - \frac{3}{4}\binom{n}{3} + c_n, \tag{1.6}$$

donde

$$c_n = \begin{cases} \frac{1}{4}E_{\frac{n-3}{2}}, & \text{si } n \text{ es impar,} \\ 0, & \text{si } n \text{ es par.} \end{cases}$$

Nótese que los dos últimos términos en la Ecuación 1.6 son $O\left(n^3\right)$.

1.3.1. Secuencias Circulares

Definición 1.6. *Sea una sucesión de permutaciones cualquiera*

$$\mathbf{\Pi} = \left(\Pi_0, \ldots, \Pi_{\binom{n}{2}}\right).$$

A $\mathbf{\Pi}$ *se le es llamado* ***secuencia circular*** *de* $\{1, \ldots, n\}$ *(o en n elementos), si las permutaciones cumplen que*

$$\Pi_0 = (1, 2, \ldots, n),$$
$$\Pi_{\binom{n}{2}} = (n, n-1, \ldots, 1),$$

y para todo par de permutaciones consecutivas, solo se difieren por exactamente una transposición de dos elementos en posiciones adyacentes.

Las secuencias circulares fueron introducidas por Goodman y Pollack [GP80] y estas permiten codificar cualquier conjunto de puntos en posición general en el plano. Considérese un conjunto de puntos en posición general P, que además tiene la propiedad de que todo par de segmentos distintos entre los puntos de P no son paralelos, es decir, las líneas que pasan por tales segmentos se interceptan. Esto último se puede asumir siempre ya que en caso de que existan pares de rectas paralelas distintas que pasen por cuatro puntos de P, es posible hacer ligeras perturbaciones en tales puntos, de tal forma que todas las rectas se intercepten preservándose la cantidad de cuadriláteros convexos.

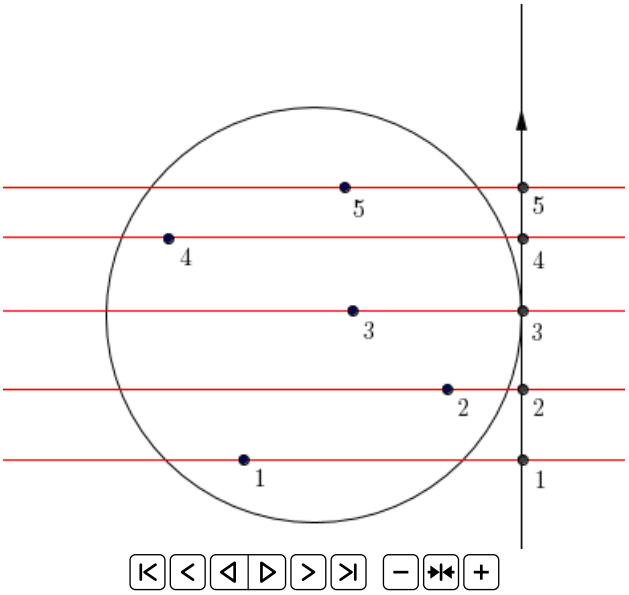

Figura 1.2: Obteniendo la secuencia circular $\mathbf{\Pi}$.

Sea l una línea dirigida que no sea perpendicular a ninguna de las líneas que pasan por los puntos de P, y supóngase que $P = \{p_1, \ldots, p_n\}$, donde los puntos son etiquetados de forma ordenada a lo largo de la línea l. Supóngase ahora que se rota l en dirección contraria a las manecillas del reloj. De este modo, el orden de las proyecciones de los puntos de P en l comienzan a cambiar cuando l empieza a ser ortogonal con alguna recta formada por dos puntos $p_i, p_j \in P$. Esto es, el orden formado por los puntos proyectados en l tiene a p_i y p_j adyacentes mientras no son ortogonales, y no solo eso, un instante antes de la ortogonalidad se tiene, sin pérdida de generalidad, la proyección de p_i antes que la de p_j en l, y después de la ortogonalidad se tiene un instante en que la proyección de p_j está antes que la de p_i en l. Con un periodo de este tipo de eventos lo suficientemente pequeño, se logran obtener una sucesión de permutaciones como las que se describen en la Definición 1.6, y con tan solo rotar la recta l un ángulo de 180° se puede construir una secuencia circular $\mathbf{\Pi} \stackrel{def}{=} \mathbf{\Pi}(P)$. Se asumirá que la recta dirigida l inicialmente es vertical y apunta hacia arriba (Ver Figura 1.2).

Nótese que si se tiene una secuencia circular $\mathbf{\Pi} = \mathbf{\Pi}(P)$ como la construida anteriormente, entonces toda $(i-1)$-arista de P corresponde a la transposición

entre elementos en las posiciones i y $i+1$, o en las posiciones $n-i$ y $n-i+1$.

Definición 1.7. *Sea P un conjunto de n puntos en posición general en el plano y sea $\mathbf{\Pi} = \mathbf{\Pi}(P)$ una secuencia circular. Las transposiciones entre elementos en las posiciones i y $i+1$, o en las posiciones $n-i$ y $n-i+1$ son llamadas **transposiciones i-críticas** de la secuencia circular $\mathbf{\Pi}$, para todo $1 \leq i < n$. Y para todo $k \leq n/2$, el número de transposiciones i-críticas con $i \leq k$ es llamado **número de transposiciones** $(\leq k)$-**críticas**.*

El segundo ingrediente para encontrar la cota inferior para el problema del número de cruce, es la siguiente cota para el número de transposiciones $(\leq k)$-críticas.

Teorema 1.3. *Para toda secuencia circular $\mathbf{\Pi}$ en n elementos y cualquier $k \leq n/2$, el número de transposiciones $(\leq k)$-críticas es al menos $3\binom{k+1}{2}$.*

Para obtener la cota inferior para la constante q_*, se usa una consecuencia del resultado en [Wel86]:

Teorema 1.4. *Sea S un conjunto de n puntos en el plano en posición general, y $k < n/2$. Entonces el número de $(< k)$-aristas de S es al menos*

$$\binom{n}{2} - n\sqrt{n^2 - 2n - 4k^2 + 4k}.$$

Esta cota es mejor que $3\binom{k+1}{2}$ si $k > 0.4956n$.

Por último en [LVWW04] se presentó el resultado principal para la cota inferior del problema del número de cruce rectilíneo.

Teorema 1.5. *Sea P un conjunto de n puntos en el plano en posición general. Entonces el número de cuadriláteros convexos determinado por P es al menos*

$$(3/8 + \varepsilon)\binom{n}{4} + O\left(n^3\right) > 0.37501\binom{n}{4},$$

donde $\varepsilon \approx 1.0872 \cdot 10^{-5}$.

Capítulo 2

Construcción de Dibujos Rectilíneos Aumentados

En búsqueda de mejorar la cota superior para la constante del problema de número de cruce (la misma constante del Problema de los Cuatro Puntos de Sylvester q_*), se emplea el siguiente método desarrollado por Ábrego y sus colaboradores en [ÁCFM$^+$10], el cual consiste en realizar dibujos rectilíneos de grafos completos cada vez más grandes a partir de un dibujo rectilíneo de un grafo completo, lo cual se conoce como **método de dibujo aumentante**. Con el fin de hacer una construcción de nuevos dibujos minimizando la cantidad de cruces, se considera el dibujo rectilíneo de una gráfica completa K_m como punto de partida. Sea $P = \{p_1, \ldots, p_m\}$ el conjunto de puntos que corresponden a los vértices de K_m en tal dibujo rectilíneo. El método de dibujo aumentante consiste básicamente en remplazar cada punto p_i en el conjunto de puntos P por un cúmulo C_i, el cual es una copia afín de un conjunto de puntos S_i llamado **modelo de cúmulo** fijado previamente (de esta forma el tipo de orden de C_i y S_i son el mismo. Ver Definición 3.3). Cada conjunto C_i se reemplaza por p_i de tal manera que la pendiente que hay entre todo par de puntos en C_i es (salvo por un error) casi igual a la pendiente de una recta l_i que pasa por p_i. Sea $C = \bigcup_{i=1}^{m} C_i$, y la línea l_i se construye de tal modo que trate de dividir al conjunto $C \setminus C_i$ en dos partes tan iguales como sea posible, es decir que logre bisecar al conjunto y además permita que para todo

par de puntos en C_i, la línea que pasa a través de estos también logre la misma partición del conjunto como lo hace l_i. De esta manera, con n igual al orden de C, se espera obtener un dibujo de K_n aumentando el dibujo de K_m.

2.1. Entrada y Salida

Tanto en la entrada como en la salida para el método de dibujo aumentante son expuestos como en el artículo [ÁCFM+10]. Para la entrada se requieren básicamente tres cosas: Un conjunto de puntos P en posición general subyacente del dibujo rectilíneo de K_m, los modelos de cúmulos $S_1, \ldots, S_m$, y unas líneas $l_1, \ldots, l_m$ las cuales son llamadas **líneas pre-bisectoras**.

2.1.1. Entrada

1. **Conjunto Base**: Un conjunto de puntos $P = \{p_1, \ldots, p_m\}$ en posición general.

2. **Modelos de Cúmulos**: Son conjuntos de puntos $S_1, \ldots, S_m$ en posición general. Cada conjunto S_i debe cumplir que para todo par de puntos u y v en S_i, las abscisas (es decir, las coordenadas de u y v en el eje x) son distintas. Sea s_i la cantidad de puntos de cada S_i y sea $I = \{i \mid s_i > 1\}$

3. **Líneas Prebisectoras**: Son líneas dirigidas $\beta_1, \ldots, \beta_m$ con la propiedad de que cada β_i contiene a p_i. Para cada β_i sea $\mathscr{L}_i$ el conjunto de los índices k tales que p_k está en el semiplano izquierdo correspondiente a β_i. De forma similar sea $\mathscr{R}_i$ el conjunto de índices k tales que p_k está en el semiplano derecho correspondiente a β_i. Si β_i pasa por un punto p_j distinto a p_i, se dice que p_i y β_i son **división**. En tal caso se dice que β_i **divide** a p_j y se define $\sigma(i) \stackrel{def}{=} j$. En otro caso se dice que p_i y β_i son **simples**. Adicionalmente, tal colección de líneas debe cumplir lo siguiente:

   a) Si $s_i \neq s_j$, entonces β_i no es igual a β_j en ninguna de las direcciones.

b) Si β_i es simple, entonces

$$0 \leq \sum_{k \in \mathscr{L}_i} s_k - \sum_{k \in \mathscr{R}_i} s_k \leq 1.$$

c) Si β_i es división, entonces β_i es dirigido de p_i a $p_{\sigma(i)}$ y

$$\left| \sum_{k \in \mathscr{L}_i} s_k - \sum_{k \in \mathscr{R}_i} s_k \right| \leq s_{\sigma(i)} - 1.$$

La condición en la Entrada 2 con respecto al eje de coordenada x, es con el objetivo de permitir hacer proyecciones en una línea paralela al eje x posteriormente. En la Entrada 3, nótese que $\sigma(i)$ no está definido cuando p_i y β_i son simples. La propiedad 3*b* lo que indica es que para cada β_i simple, debe haber un balance con respecto a la cantidad de puntos (que reemplazarán a cada p_j distinto de p_i) entre los dos semiplanos que define β_i, dando como resultado a lo sumo un punto más en el semiplano izquierdo con respecto a β_i. La propiedad 3*c* es un poco similar a la anterior, esto se debe a que en este caso para cada β_i división se busca que al reemplazar (de forma adecuada) el punto $p_{\sigma(i)}$ por los $s_{\sigma(i)}$ puntos correspondientes, se logre obtener una desigualdad similar a la desigualdad en 3*b*. (Ver Figura 2.1).

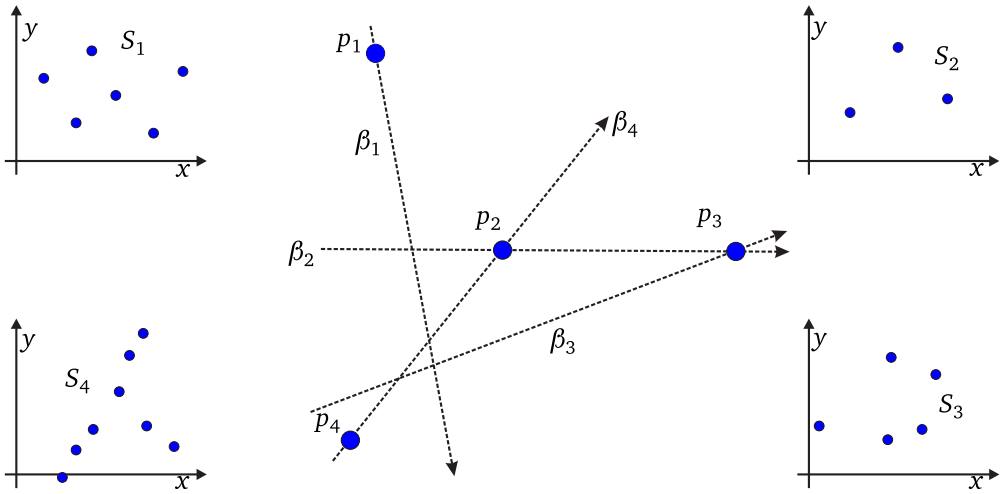

Figura 2.1: La entrada.

2.1.2. Salida

Antes de continuar con la salida del método de dibujo aumentante, es preciso dar una definición previa.

Definición 2.1. *Sean Q y C dos conjuntos de puntos y l una línea dirigida. Se dice que l **biseca** a Q si el semiplano izquierdo con respecto a l contiene a $\lceil |Q|/2 \rceil$ puntos de Q y el otro semiplano contiene a los demás $\lfloor |Q|/2 \rfloor$ puntos de Q. Adicionalmente se dice que C **biseca** a Q como l, si toda línea definida por cualquier par de puntos en C, definen la misma partición de los puntos de Q que define la línea l.*

La salida consiste en un conjunto de puntos $C = \bigcup_{i=1}^{m} C_i$, donde cada punto p_i en P es reemplazado por el conjunto de puntos C_i, el cual es una copia afín del modelo de cúmulo S_i. A la vez que se construye cada C_i, también se construye una línea dirigida l_i de tal forma que al final de la construcción, C_i junto con l_i deben satisfacer las siguientes propiedades.

1. **Propiedad de tipo de orden heredada**. Para todo $q_i \in C_i$, $q_j \in C_j$ y $q_k \in C_k$, donde $i, j, k \in \{1, \ldots, m\}$ son distintos, el tipo de orden[1] de la terna de puntos $q_i q_j q_k$ es el mismo que el tipo de orden de $p_i p_j p_k$.
2. **Propiedad de bisección**. Para cada $i \in I$, la línea l_i biseca a $C \setminus C_i$ y el conjunto de puntos C_i biseca a $C \setminus C_i$ como l_i.

2.2. La Construcción: ¿Qué pasa en medio de la entrada y la salida?

Paso 1. Para cada punto p_i en P, tomándose a este como centro, se considera un disco D^i de radio pequeño.

Para cada $i = 1, \ldots, m$ sea D^i un disco de radio r_i centrado en p_i, tal que $D^1, \ldots, D^m$ cumplen que para cualquier terna de índices distintos i, j y k y para todo $q_i \in D^i$, $q_j \in D^j$ y $q_k \in D^k$, el tipo de orden de la terna de puntos $q_i q_j q_k$

[1] Ver Definición 3.3.

es el mismo que el tipo de orden de $p_i p_j p_k$. Esto se logra haciendo $r_1, \ldots, r_m$ lo suficientemente pequeños.

Paso 2. Reemplazar cada punto p_i con un conjunto U_i contenido en $D^i \cap \beta_i$.

En una primera instancia, se reemplaza a cada punto p_i en P por un conjunto de puntos colineales U_i en β_i y contenido en el disco D^i. Para cada $i \in I$, tomando como el origen a p_i y de eje x a β_i, sea U_i la proyección de una copia afín C_i del modelo de cúmulo S_i contenida en el disco D^i que dista de p_i a lo sumo $r_i/2$. Por tal motivo es que se necesita la condición de que S_i no tenga un par de puntos con la misma abscisa. Si $i \notin I$, esto es, si $s_i = 1$, entonces $U_i = \{p_i\}$. De esta forma se tiene que U_i está contenido en D^i y no se aleja más de $r_i/2$ del centro. (Ver Figura 2.2).

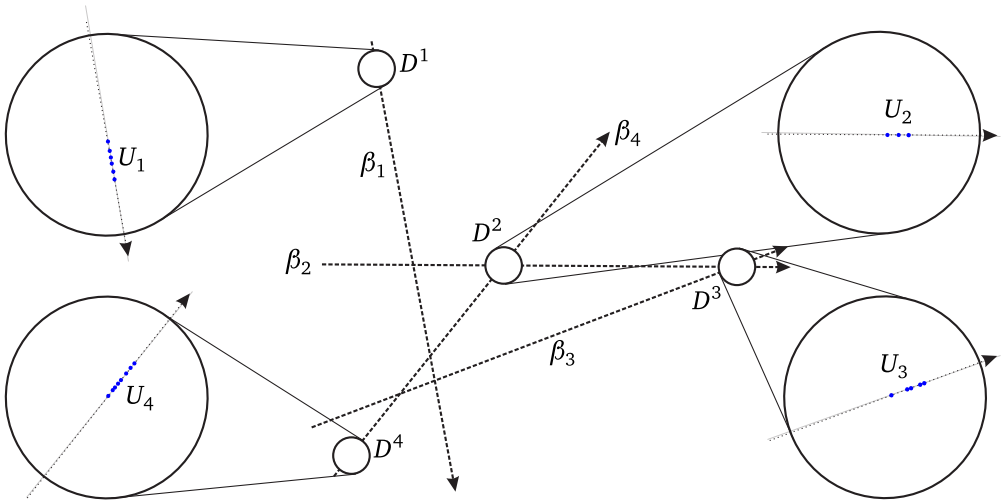

Figura 2.2: Reemplazando cada punto p_i por el conjunto U_i.

Antes de continuar con el próximo paso, se nota que cada línea β_i es un buen candidato para ser una linea bisectora para el conjunto de puntos de la salida. En efecto, si β_i es simple, esta logra bisecar al conjunto de puntos $U \setminus U_i$. Y si β_i es división, entonces la diferencia de puntos en $U \setminus U_i$ en cada lado de β_i es a lo sumo $S_{\sigma(i)} - 1$. En este caso, β_i no necesariamente biseca a $U \setminus U_i$, pero intercepta a $D^{\sigma(i)}$, el cual contiene $S_{\sigma(i)}$ puntos de $U \setminus U_i$. Por lo tanto, con una pequeña

rotación de U_i (y a su vez de β_i) se puede balancear esta diferencia (ver Figura 2.3). Sin embargo, no solo basta con hacer de manera iterativa rotaciones de conjuntos de puntos U_i, ya que eso solo sirve para ajustar una línea β_i particular en una cierta iteración. Si posteriormente en otra iteración se intenta ajustar la línea $\beta_{\sigma(i)}$, entonces al rotar el conjunto U_i es posible que se pierda la propiedad de bisección de β_i. Para evitar este tipo de fracasos, se procede adicionalmente también con traslaciones de los conjuntos $U_{\sigma(i)}$, lo cual se describe en el siguiente paso.

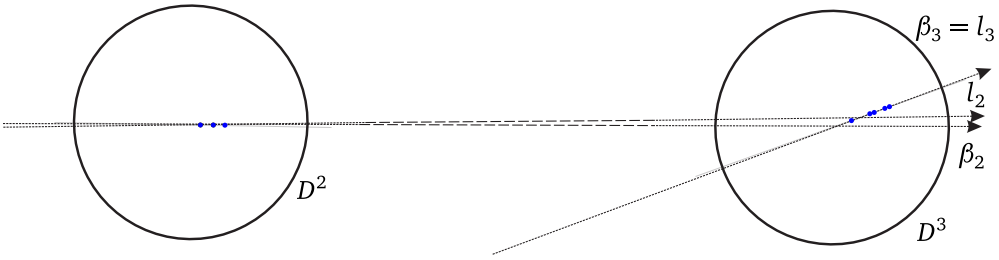

Figura 2.3: Rotando cada conjunto U_i.

Paso 3. Mover los conjuntos $U_1, \ldots, U_m$, de tal modo que cada U_i esté en una línea l_i que biseque a $U \setminus U_i$.

El objetivo en este paso es mover ligeramente (rotando o trasladando) cada U_i con $i \in I$, tal que cada línea que contiene a U_i pase por p_i y biseque a $U \setminus U_i$. Se denota como la línea dirigida l_i a la línea que contiene a los puntos U_i (la cual inicialmente es β_i). Si $s_i = 1$, entonces $U_i = \{p_i\}$ no cambia durante todo el proceso. La función central de todo el proceso se define a continuación.

Propiedad clave. Si $s_i > 1$, entonces U_i está contenido en el interior de D^i y está en l_i en todo el proceso. La posición final de U_i es tal que l_i pasa por p_i y biseca a $U \setminus U_i$.

Para describir el proceso, se considera una gráfica dirigida G con vértices $P' = \{p_i \in P \mid i \in I\}$ y para todo par de puntos $p_i, p_j \in P'$ se considera como arco al par ordenado (p_i, p_j), si $\sigma(i) = j$ (ver Figura 2.4). Así, si p_i es simple, entonces su grado de salida es 0, y si p_i es división, entonces su grado de salida es 1. Estas dos

propiedades garantizan que cada una de las componentes fuertemente conexas de G sean acíclicas o contengan a lo sumo un ciclo dirigido. En cualquiera de ambos casos, cada componente fuertemente conexa de G puede tener un vértice, llámese **raíz**, que puede ser alcanzado desde cualquier otro vértice de la componente, es decir, para todo vértice distinto de la raíz, existe un camino dirigido de tal vértice a la raíz.

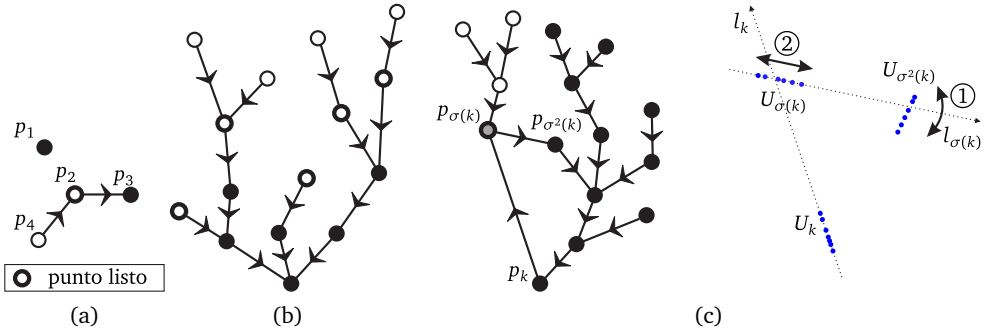

Figura 2.4: (a) La digráfica G correspondiente al ejemplo de la Figura 2.1. (b) Un componente acíclica. (c) Un componente con un ciclo y una aplicación de la etapa (2) del Paso 3.

Para cada una de las componentes fuertemente conexas se hace el siguiente trabajo. Sea $P_c \subseteq P$ una componente fuertemente conexa de G con raíz p_k. Se comienza con todos los vértices de G de color blanco. Pintar un punto p_i de negro significa que l_i y U_i alcanzaron su posición final. El color de p_k es negro y si p_k es división, entonces $p_{\sigma(k)}$ es gris. Un punto blanco o gris se dice que está **listo** si $p_{\sigma(k)}$ es negro. Mientras hayan puntos listos, se aplica la etapa 1 o 2:

1. Si se puede, se escoge un punto listo blanco cualquiera p_i. Se rota ligeramente U_i en torno a p_i hasta que l_i biseque a $U \setminus U_i$. Esto se puede lograr siempre si se pide que l_i siempre intercepte a $D^{\sigma(i)}$, porque $\beta_i \neq \pm\beta_j$, l_i intercepta a $D^{\sigma(i)}$, $D^{\sigma(i)}$ tiene $s_{\sigma(i)}$ puntos, y antes de rotar a U_i se tiene un desbalance de a lo sumo $s_{\sigma(i)} - 1$. Después se pinta p_i de negro.

2. Si la etapa 1 no pudo ser aplicada, entonces se trabaja con el punto gris $p_{\sigma(k)}$. Primero se hace lo mismo que en la etapa 1, esto es, se rota $U_{\sigma(k)}$ en torno a $p_{\sigma(k)}$ hasta que $l_{\sigma(k)}$ biseque a $U \setminus U_{\sigma(k)}$. Después se traslada a

$U_{\sigma(k)}$ a través de $l_{\sigma(k)}$ (el cual permanece aún bisecando a $U \setminus U_{\sigma(k)}$) hasta que la línea l_k logre bisecar al conjunto $U \setminus U_k$. Como inicialmente $U_{\sigma(k)}$ fue contenido en un disco de radio $r_{\sigma(i)}/2$ centrado en $p_{\sigma(k)}$, entonces $U_{\sigma(k)}$ aún está contenido en el disco $D^{\sigma(k)}$ durante la traslación. Luego se pone $p_{\sigma(k)}$ de color negro. (Ver Figura 2.4(c)).

Obsérvese que la etapa 2 es aplicada solo una vez, y si no se pudo aplicar ninguna de las etapas 1 y 2, entonces todos los puntos ya están pintados de negro. Ya que la propiedad clave es mantenida en todo el proceso para cada punto escogido en las etapas anteriores, entonces al final se logra que la propiedad clave se cumpla para todos los puntos, es decir, cada U_i está en l_i y está contenido en el interior de D^i, también l_i pasa por p_i y biseca al conjunto $U \setminus U_i$. Los movimientos de los conjuntos U_i son a su vez realizados para las copias afines C_i, para conservar la propiedad de que U_i sea la proyección de C_i, con punto de origen p_i y como eje de coordenada x la línea l_i.

Paso 4. Aplanando cada copia afín C_i a su proyección U_i.

Para cada $i \in I$, se considera ahora acercar afínmente cada C_i a U_i para obtener la posición final. Si $s_i = 1$, entonces $C_i = \{p_i\}$. Para cada $i \in I$ considerando de nuevo cada punto $p_i \in P$ como el origen y a la línea l_i como el eje de coordenadas x, todas las ordenadas (la componente de coordenada y) de los puntos de C_i son multiplicadas por un número $\varepsilon \in [0,1]$, se llama al conjunto de puntos resultante $C_i(\varepsilon)$. El objetivo es hacer a ε lo suficientemente pequeño para que se cumpla la propiedad de bisección, esto es, l_i biseca a $C \setminus C_i(\varepsilon)$ y $C_i(\varepsilon)$ biseca a $C \setminus C_i(\varepsilon)$ como lo hace l_i, donde $C = \bigcup_{i=1}^{m} C_i(\varepsilon)$. Para lograr este fin, se define la función $f(\varepsilon)$ como el mínimo para todo i de las distancias de todos los puntos en el conjunto $\bigcup_{j \neq i} C_j(\varepsilon)$ a l_i tal que para $j \neq i$, una distancia de un punto en $C_j(\varepsilon)$ se hace negativa si su respectivo punto en U_j está en lado opuesto con respecto a l_i. Obsérvese que la función f es continua y que $f(0) > 0$ ya que $\bigcup_{i=1}^{m} C_i(\varepsilon) = U$. Por lo tanto existe un $\varepsilon' > 0$ tal que $f(\varepsilon') > 0$. La posición final de C_i es $C_i(\varepsilon')$. Luego $C = \bigcup_{i=1}^{m} C_i$ y cada C_i está contenido en D_i. Así C logra satisfacer la propiedad de orden heredada. Y debido a que U cumple la propiedad de bisección, entonces C también satisface tal propiedad.

Se obtuvo en [ÁCFM$^+$10] junto con [ÁFM07] el cálculo del número de cruces rectilíneo que posee un conjunto de puntos que resulta al aumentar otro conjunto de puntos en posición general. Un caso particular de este resultado fue el número de cruces en el caso de la duplicación de puntos, para ello se usó emparejamientos de líneas bisectoras (ver Definición 3.7).

Teorema 2.1. *Sea P un conjunto de puntos en posición general de tamaño m para el cual existe un emparejamiento de líneas bisectoras. Entonces existe un conjunto de puntos Q en posición general de tamaño $2m$, que tiene un emparejamiento por líneas bisectoras, y satisface que*

$$\overline{cr}(Q) = 16\overline{cr}(P) + \frac{m}{2}\left(2m^2 - 7m + 5\right).$$

En base a la duplicación de un conjunto impar de puntos en posición general, se tiene el siguiente resultado.

Teorema 2.2. *Sea P un conjunto de puntos impar en posición general de m elementos. Entonces*

$$\overline{cr}(K_n) \leq \frac{24\overline{cr}(P) + 3m^3 - 7m^2 + (30/7)m}{m^4}\binom{n}{4} + \Theta(n^3).$$

De esta manera se presenta una forma de acotar superiormente la constante del número de cruce rectilíneo q_* a partir de un conjunto de puntos impar en posición general.

El problema que hay hasta la fecha es que no existe una implementación del método de dibujo aumentante, ni si quiera en el caso particular de la duplicación de conjuntos de puntos en posición general. Por lo tanto, no hay una manera sistemática de construir los conjuntos de puntos usando este método, y la utilidad que tiene la duplicación de puntos, sería solo el determinar la cota superior de q_* usando el Teorema 2.2.

Capítulo 3

Implementación de la Construcción

Basándose en la construcción de dibujos rectilíneos aumentados del Capítulo 2, se presentará en este capítulo una implementación en coordenadas enteras para duplicar conjuntos de puntos en el plano. Para lograr esto, en la Sección 3.1 se introducirá el concepto de tipos de orden. Posteriormente en la Sección 3.2 se definirá formalmente una estructura de datos llamada Treap. Luego en la Sección 3.3 se partirá el problema de duplicación en dos casos: Cuando el conjunto de puntos es impar y cuando el conjunto de puntos es par.

3.1. Tipos de Orden

Lo que se pretende es trabajar con conjuntos de puntos grandes eficientemente (no necesariamente en posición general), de tal forma que se requiere establecer un orden entre ellos. Una manera de hacerlo es definiendo un orden en el plano con respecto a un punto fuera del conjunto de puntos. Para poder hacerlo, se puede considerar la siguiente definición.

Definición 3.1. *Sean $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ y $p_3 = (x_3, y_3)$ puntos en el plano. Se dice que p_1, p_2 y p_3 tiene* ***orientación positiva*** *(o* ***gira a la izquierda***

desde p_1), y es denotado como $p_1p_2p_3 > 0$, si

$$\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} > 0.$$

Se dice que p_1, p_2 y p_3 tiene ***orientación negativa*** *(o* ***gira a la derecha*** *desde p_1), y se de nota como $p_1p_2p_3 < 0$, si el determinante anterior es negativo. Y se dice que p_1, p_2 y p_3 tiene* ***orientación nula*** *(o* ***no gira****), denotado como $p_1p_2p_3 = 0$, si el determinante anterior es cero.*

Nótese que el orden en que se dice que tres puntos tienen orientación positiva o negativa es importante, ya que si la orientación de tres puntos p_1, p_2 y p_3 en el plano es no nula, entonces la orientación de p_3, p_2 y p_1 es contraria a la anterior ya que

$$\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} = - \begin{vmatrix} 1 & x_3 & y_3 \\ 1 & x_2 & y_2 \\ 1 & x_1 & y_1 \end{vmatrix}.$$

Con la Definición 3.1 se tiene lo necesario para definir un orden total en un conjunto de puntos en el plano.

Definición 3.2. *Sea $P = \{p_1, \ldots, p_n\}$ un conjunto de puntos en el plano, y sea p un punto que no está en P. Se define el* ***orden en sentido antihorario*** *de P con respecto a p como la relación $<_p$ tal que para todo par de puntos distintos $p_i, p_j \in P$ se tiene que $p_i <_p p_j$ si se cumple alguno de los siguientes casos:*

1. *Si p, p_i y p_j son colineales y p no está entre p_i y p_j, y la distancia de p_i a p es menor que la distancia de p_j a p.*

2. *En caso contrario, es decir, si p, p_i y p_j no son colineales ó, son colineales y p está entre p_i y p_j. Tomando como inicio la semirrecta en dirección hacia abajo en el plano (en dirección al sur) con origen en el punto p, y en sentido contra las manecillas del reloj, se cumple que p_i está antes que p_j.*

Para lograr implementar la definición anterior, se puede considerar una transformación lineal que transforme el plano en el semiplano que está arriba de la línea horizontal que pasa por p, que preserve la propiedad de que todo punto de coordenadas enteras es mapeado a un punto de coordenadas enteras y además que preserve el orden antihorario tomando como origen al punto p. De esta manera el caso 2 puede ser examinado tan solo usando la orientación que hay entre los puntos p, y los puntos resultantes de mapear a p_i y p_j. Considérese entonces la siguiente función

$$f(x,y) = \begin{cases} (x, |x| + y), & \text{si } y \geq 0, \\ (x - sig(x)y, |x|), & \text{si } y < 0, \end{cases}$$

donde $sig(x)$ toma el valor 1 si $x \geq 0$ y el valor -1 en caso contrario. Luego para todo punto p se define la función $f_p(q) = f(q-p)+p$, donde q es cualquier punto en el plano distinto de p.

Nótese que con la función f_p se cumple con todos los requisitos de la transformación lineal que se describe arriba. Claramente f_p cumple que todo punto con coordenadas enteras es mapeado a un punto con coordenadas enteras. Y se puede apreciar a través de la Figura 3.1 que f_p preserva el orden antihorario con respecto al punto p. Además de la propiedad de que $f_p(p) = p$, se tiene una propiedad muy importante: Cuando $pf_p(p_i)f_p(p_j) = 0$, es decir, cuando la orientación de los puntos p, $f_p(p_i)$ y $f_p(p_j)$ es nula, es equivalente a que p, p_i y p_j son colineales y p no está entre p_i y p_j, lo cual es una parte del caso 1 de la Definición 3.2.

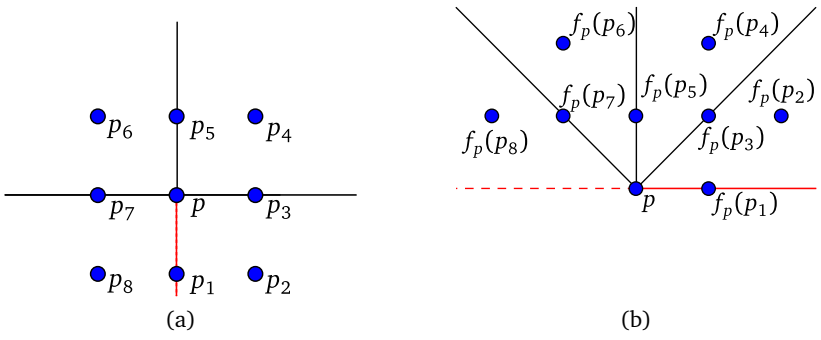

Figura 3.1: (a) Los puntos $P = \{p_1, \ldots, p_8\}$ y $p \notin P$. (b) Todos los puntos de P mapeados con la transformación lineal f_p.

Con base a todo lo anterior, se tiene el siguiente teorema.

Teorema 3.1. *Sea P un conjunto de puntos en el plano (no necesariamente en posición general) y sea p un punto que no está en P. Entonces para todo par de puntos distintos $p_i, p_j \in P$ se tiene que $p_i <_p p_j$ si y solo si se cumplen algunos de los siguientes dos casos:*

1. $pf_p(p_i)f_p(p_j) > 0$.
2. $pf_p(p_i)f_p(p_j) = 0$ *y la distancia de p_i a p es menor que la distancia de p_j a p.*

Obsérvese que el orden antihorario descrito en la Definición 3.2 y en el Teorema 3.1 es un orden total sobre los conjuntos de puntos en el plano.

Por otro lado, regresando al problema de número de cruce, nótese que existe infinidades de conjuntos distintos de n puntos en posición general que pueden ser representados por un solo conjunto de n puntos. Esto se debe a que puede existir un reetiquetamiento para cada uno de tal infinidad de conjuntos con su representante, de tal manera que se preserva la propiedad de cruce entre las aristas. Goodman y Pollack se enfrentaron al problema de clasificación de conjuntos de puntos (no solo en el plano, también en dimensiones superiores) en [GP83], y junto con la Definición 3.1, presentaron la siguiente forma de clasificar los conjuntos de puntos en el plano.

Definición 3.3. *Sea $P = \{p_1, \dots p_n\}$ un conjunto de puntos en el plano en posición general. El **tipo de orden** de P es la función que asigna a cada terna de puntos en P su orientación.*

Definición 3.4. *Sea P y Q dos conjuntos de puntos en el plano en posición general. Se dice que P y Q tienen el **mismo tipo de orden** si existe una función biyectiva de P a Q que preserva las orientaciones entre ternas de puntos.*

3.2. Treaps

Una ves se ha establecido un orden total para los conjuntos de puntos en el plano, también se hace necesario tener una manera eficiente para tener acceso a

los puntos, y poder hacer inserción y borrado de puntos. Para ello se debe elegir una estructura de datos adecuada.

Una **estructura de datos** es una manera de almacenar y organizar datos para facilitar el acceso y las modificaciones. Dependiendo del propósito que se tenga, se debe elegir el tipo de estructura de datos para implementar, ya que no existe una estructura de datos que sirva para todos los fines.

Por lo anterior en la implementación se usará para los conjuntos de puntos una estructura de datos llamada *Treap*. Sea S un conjunto de pares ordenados (los cuales están en una función inyectiva), llamados **nodos** (o **vértices**). Para un nodo $v \in S$, se denota como $v.key$ al valor de la abscisa de v y es llamado **llave** de v, y se denota como $v.priority$ al valor de la ordenada de v y es llamado **prioridad** de v.

Definición 3.5. *Sea S un conjunto de nodos. Un **Treap** para S, denotado como $T(S)$, es una estructura de datos, donde el nodo de S con menor prioridad es llamado **raíz** y se denota como $T(S).root$, y $T(S)$ está definido por la siguiente recursión:*

1. *Si $|S| = 1$, entonces el único nodo $T(S).root$ es llamado **hoja** de $T(S)$.*
2. *Si $|S| > 1$, entonces $T(S_l)$ es llamado **subtreap izquierdo** y $T(S_r)$ es llamado **subtreap derecho** de $T(S)$, donde*

$$S_l \stackrel{def}{=} \{v \in S \mid v.key < T(S).root.key\},$$

$$S_r \stackrel{def}{=} \{v \in S \mid v.key > T(S).root.key\}.$$

Usualmente, en la literatura un treap sobre un conjunto de nodos S se define como una combinación de dos estructuras de datos, un *árbol de búsqueda binaria* con respecto a las llaves de los nodos, y un *heap* con respecto a las prioridades de los nodos. Nótese que para todo par de nodos distintos $u, v \in S$ se cumple lo siguiente en el treap definido por S:

- Si v está en el subtreap izquierdo de u, entonces $v.key < u.key$,
- Si v está en el subtreap derecho de u, entonces $v.key > u.key$,

- Si v es **descendiente** de u (o u es **ancestro** de v), es decir, v está en uno de los subtreaps de u, entonces $v.priority > u.priority$.

Las primeras dos propiedades corresponden a la definición de lo que es un árbol de búsqueda binaria, mientras que la tercera propiedad es la propiedad de un heap.

Se puede considerar que un treap es un árbol de búsqueda binaria que fue construido con las llaves de los nodos en orden a cada una de las prioridades. Esto último es muy importante tenerlo en cuenta, ya que una propiedad muy importante en los árboles de búsqueda binaria es que, si se construye el árbol en un orden aleatorio con respecto a sus llaves, entonces la altura esperada del árbol es $O(\log n)$. Por lo tanto si las prioridades en los nodos del conjunto S se consideran aleatorios para sus respectivas llaves, entonces el treap $T(S)$ tendrá una altura esperada $O(\log n)$.

Para insertar nuevos nodos en un treap, inicialmente se inserta como si se estuviera insertando el nodo en un árbol de búsqueda binaria con respecto a las llaves, esto es, si v es un nuevo nodo que se va a insertar en un treap $T(S)$, recursivamente se hace lo siguiente:

- Si $|S| = 0$, entonces $T(S) = T(\{v\})$.
- Si $|S| = 1$:
  - Si $v.key < T(S).root.key$, entonces v es el **hijo izquierdo** de $T(S).root$
  - Si $v.key > T(S).root.key$, entonces v es el **hijo derecho** de $T(S).root$
- Si $|S| > 1$:
  - Si $v.key < T(S).root.key$, entonces se inserta v en $T(S_l)$.
  - Si $v.key > T(S).root.key$, entonces se inserta v en $T(S_r)$.

Este proceso tiene un tiempo de ejecución de $O(\log n)$, ya que en el peor de los casos, la inserción recorre toda una trayectoria desde la raíz hasta la hoja el cual es de longitud estimada $O(\log n)$, teniendo en cuenta que la prioridad de los nodos se escogió de forma aleatoria.

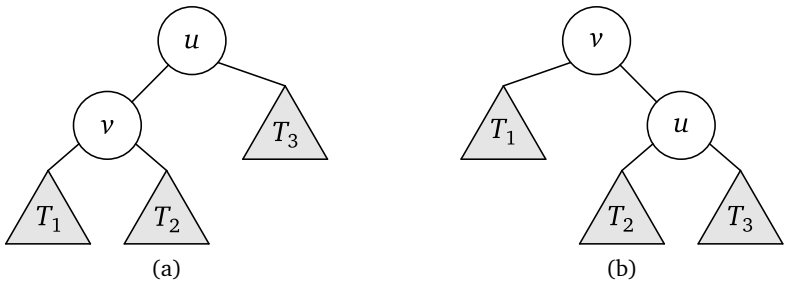

Figura 3.2: En (a) se rota a la derecha el árbol a (b) para que v sea la nueva raíz del árbol. Y en (b) se rota a la izquierda el árbol a (a) para que u sea la nueva raíz del árbol

Nótese que, como está en este momento la inserción de v en $T(S)$, es posible que aún no se haya logrado obtener el treap $T(S \cup \{v\})$ ya que tal vez no está cumpliendo la propiedad de heap. Esto se puede solucionar haciendo rotaciones de árbol. Los árboles de búsqueda binaria también tienen una operación llamada *rotación*, la cual le permite cambiar la raíz que tiene actualmente por cualquiera de sus hijos (ver Figura 3.2). Para lograr construir el treap $T(S \cup \{v\})$, se realiza de forma iterativa unas rotaciones sobre el subtreap que tiene como raíz a v. La operación de rotación aplicada una sola vez tiene tiempo de ejecución constante $O(1)$. Sin embargo, es posible que en el peor de los casos, el nodo que se inserta baja hasta una hoja en la inserción como árbol de búsqueda binaria con respecto a su llave, pero el valor de su prioridad es el más pequeño. Por tal motivo ese nodo tendrá que hacer un estimado de $O(\log n)$ rotaciones para poder estar en el lugar que le corresponde.[1] Por lo tanto el tiempo total de ejecución que tarda el insertar un nodo en un treap sigue siendo de $O(\log n)$.

En la práctica, las rotaciones esperadas que se requieren para tal inserción es a lo sumo dos rotaciones. Esto último es muy importante tenerlo en cuenta para la implementación, ya que cuando se hace una rotación en un treap, se hacen cambios de apuntadores. En vista de que en el acceso a los nodos en un treap solo se hace operaciones de lectura y eso no es relativamente muy costoso, es bueno contar con que también la operación de inserción no sea muy costosa comparada con esto.

[1] La altura estimada del treap $T(S)$ es $O(\log n)$.

Por último, la extracción (o borrado) de un nodo v en un treap $T(S)$ se puede hacer de la siguiente manera recursiva. Se le actualiza el valor de la prioridad $v.priority = \infty$. Luego se hacen las rotaciones necesarias para que $T(S)$ vuelva a ser un treap, ya que con el paso anterior $T(S)$ perdío la propiedad de $heap$. Finalmente se borra v del treap $T(S)$, dando como resultado el treap $T(S \setminus \{v\})$. Estas operaciones son nuevamente de rotaciones. Por lo tanto el tiempo de ejecución sigue siendo de $O(\log n)$.

A continuación se presentarán los treaps que se utilizarán para la implementación de este capítulo.

Definición 3.6. *Sea $P = \{p_1, \ldots, p_n\}$ un conjunto de puntos en el plano, y sea p un punto que no está en P. Entonces se define el **treap generado** por p, denotado por T_p, como un treap cuyas llaves son los puntos de P bajo el orden en sentido antihorario con respecto a p, y cuyas prioridades son números enteros asociados a cada punto de P de forma aleatoria e inyectiva. Se dice además que p es el **origen** de T_p.*

En lo posterior de esta tesis, siempre que se mencione acerca de un treap T, se tomará de antemano que existe un punto p tal que $T = T_p$, y el conjunto de puntos P con el que se define el treap T puede ser el conjunto vacío o no necesariamente ser un conjunto de puntos en posición general.

3.3. Duplicación de un Conjunto de Puntos

Sin pérdida de generalidad se asumirá a partir de aquí que todo conjunto de puntos que se defina en el plano, tiene todos sus puntos definidos en coordenadas enteras. Para poder realizar una duplicación de un conjunto de puntos en posición general dado, es necesario que tal conjunto cumpla con una única característica, la cual se presenta en la siguiente definición.

Definición 3.7. *Sea $P = \{p_1, \ldots, p_n\}$ un conjunto de n puntos en posición general. Se dice que P tiene un **emparejamiento de líneas bisectoras**, si existe un conjunto de n líneas $H = \{l_1, \ldots, l_n\}$, tal que cada línea l_i es una línea que pasa por p_i y cada*

uno de los semiplanos generados por l_i al dividir el plano, contiene $\lfloor n/2 \rfloor$ puntos de P.

A continuación se dividirá el problema de duplicación de un conjunto de puntos en dos casos: El primer caso es cuando el conjunto de puntos es impar, y el segundo caso es cuando el conjunto de puntos es par. El motivo por el que se hace de esta manera es, y se mostrará posteriormente, que por un lado todo conjunto de puntos en posición general de orden impar tiene un emparejamiento de líneas bisectoras, y que por otro lado no siempre existe un emparejamiento de líneas bisectoras si el conjunto de puntos es par.

3.3.1. Duplicación de un Conjunto Impar de Puntos

Teorema 3.2. *Sea $P = \{p_1, \ldots, p_{2n+1}\}$ un conjunto impar de puntos en posición general. Entonces P tiene un emparejamiento de líneas bisectoras.*

Demostración. Sea $l'_1, l'_2, \ldots, l'_{2n}$ las líneas dirigidas ordenadas de forma creciente con respecto a la pendiente y orientadas en orden contrarreloj, de tal forma que pasan por p_{2n+1} y por un punto de $P \setminus \{p_{2n+1}\}$, se supone sin pérdida de generalidad que pasan por $p_1, p_2, \ldots, p_{2n}$ respectivamente. Sea además R_i la cantidad de puntos de P a la derecha de l'_i y sea $L_i = 2n - R_i$, esto es, el número de puntos de P que está a la izquierda de l'_i contando al punto p_i. Con respecto a las líneas l'_i y l'_{i+1} se tiene que p_i puede estar en el lado derecho de l'_{i+1} o que p_{i+1} puede estar en el lado derecho de l'_i, por lo tanto se tiene que $|R_i - R_{i+1}| \leq 1$. De forma similar, con respecto a las líneas l'_1 y l'_{2n} se tiene que p_1 puede estar en el lado izquierdo de l'_{2n} o que p_{2n} puede estar al lado derecho de l'_1, y p_{2n} se cuenta en L_{2n}, por lo tanto $0 \leq L_{2n} - R_1 \leq 2$. Si $R_1 = L_1$, entonces toda línea que pase por la región vacía con respecto a $P \setminus \{p_{2n+1}\}$, delimitada por las rectas l'_1 y l'_{2n}, es una línea bisectora que pasa por p_{2n+1}. Sin pérdida de generalidad supóngase que $R_1 > L_1$, entonces sustituyendo a L_1 se tiene que $R_1 > n$ y así por lo anterior $L_{2n} > n$, esto es, $L_{2n} > R_{2n}$. Como $R_1 > L_1$, $R_{2n} < L_{2n}$ y $|R_i - R_{i+1}| \leq 1$ para $1 \leq i \leq 2n$, por lo tanto existe $i > 1$ tal que $R_i = L_i$. Luego toda línea que pase por la región vacía con respecto a $P \setminus \{p_{2n+1}\}$, delimitada por las rectas l'_i y l'_{i-1}, es una línea bisecto-

ra que pasa por p_{2n+1}. Similarmente se logra demostrar el mismo resultado en el caso de que $R_1 < L_1$. De forma análoga se puede encontrar líneas bisectoras para los demás puntos de P. ■

En la demostración del Teorema 3.2 se presenta una manera de como encontrar una región que contenga líneas bisectoras. Se muestra a continuación una implementación que solo manipule coordenadas enteras para encontrar una de tales regiones.

Algoritmo 3.1 Algoritmo para hallar una región de líneas bisectoras.

```
1: procedure REGION(p, P)
2:     n := orden de P
3:     Q := P reetiquetado tal que q_1 <_p ... <_p q_n
4:     i := 0
5:     while not ( q_i p q_(i+n/2) > 0 and q_i p q_(i+n/2+1) < 0 ) do
6:         i := i + 1
7:     end while
8:     return [q_i, q_(i-1), q_(i+n/2), q_(i+n/2+1)]
9: end procedure
```

En el Algoritmo 3.1 se ingresa como entrada un punto p y un conjunto de puntos de orden par P visto como una lista tal que $p \notin P$ y que $P \cup \{p\}$ esté en posición general. Lo que retorna este algoritmo es una lista de cuatro puntos $R = \left[q_1, q_2, q_3, q_4\right]$ tales que todo punto r que cumpla $q_2 <_p r <_p q_1$ y $q_3 <_p -r + 2p <_p q_4$, donde $-r + 2p$ es el reflejo de r en p, entonces la línea que pasa por p y r es una línea que biseca a $P \cup \{p\}$. Ver Figura 3.3.

El siguiente paso es definir un punto en coordenadas enteras lo suficientemente cercano al punto p que esté a su vez dentro de la región que se generó.

Definición 3.8. *Sea p un punto en el plano. Se dice que un punto r es **visible** para p, si no existe un punto que tenga coordenadas enteras que esté dentro del segmento rectilíneo definido por p y r.*

Si $p = (0,0)$ y $r = (h,k)$, salvo en los casos $(h,k) = (0,\pm 1)$ o $(h,k) = (\pm 1, 0)$, se cumple que el número k/h es la pendiente de la recta que pasa por los puntos

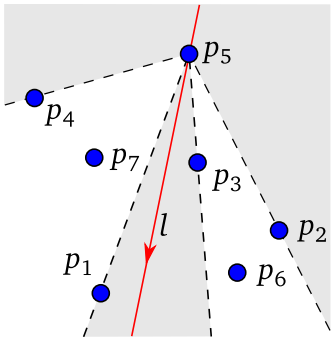

Figura 3.3: Toda línea l dentro de la región sombreada comprendida entre los puntos p_5, p_1, p_3, p_2 y p_4 es una línea bisectora que pasa por p_5 y biseca a $P = \{p_1, \ldots, p_7\}$, donde $p_3 <_{p_5} p_6 <_{p_5} p_2 <_{p_5} p_4 <_{p_5} p_7 <_{p_5} p_1$.

p y r. Por tanto existen enteros i y j tales que el máximo común divisor entre ellos es 1, es decir $\gcd(i, j) = 1$, cumplen que $k/h = j/i$, y se satisface que si (i, j) es un punto visible para $(0, 0)$, entonces $i = h$ y $j = k$. Por lo tanto el conjunto de todos los puntos visibles para un punto p cualquiera son todos aquellos puntos de la forma $(i, j) + p$ donde $\gcd(i, j) = 1$, junto con los puntos $(0, \pm 1) + p$ y $(\pm 1, 0) + p$.

Dada una región determinada por un punto p y una lista de puntos $[q_1, q_2, q_3, q_4]$, se usa la Definición 3.8 para buscar un punto r que sea visible para p tal que la línea que pasa por tales puntos está dentro de la región.

Algoritmo 3.2 Algoritmo para hallar un punto visible dentro de una región.

```
1:  procedure CandidateVector(p, q1, q2, q3, q4)
2:      for i = 1 to 4 do
3:          q_i := Punto visible de p en dirección q_i
4:      end for
5:      k := 2
6:      while True do
7:          for i = 1 to k do
8:              j := k − i
9:              if gcd(i, j) = 1 then
10:                 V := [(i, j), (−i, j), (i, −j), (−i, −j)]
11:                 for all r ∈ V do
12:                     r1 := r + p
13:                     r2 := −r + p
```

14: **if** $\left(\begin{array}{l} q_2pr_1 > 0 \textbf{ and } r_1pq_1 > 0 \textbf{ and} \\ q_3pr_2 > 0 \textbf{ and } r_2pq_4 > 0 \end{array}\right)$ **then**
15: **return** r_1
16: **end if**
17: **end for**
18: **end if**
19: **end for**
20: **end while**
21: **end procedure**

Con el Algoritmo 3.2 se describe un procedimiento para hallar tal punto r, siempre que p, q_1, q_2, q_3 y q_4 sean puntos distintos en posición general. Una vez hecho esto es necesario saber si para cualquier punto $q \in P$ se cumple que p y r son consecutivos en el orden contrarreloj generado por q sobre el conjunto $(P \cup \{p, r\}) \setminus \{q\}$.

Definición 3.9. *Sea P un conjunto de puntos, $r \notin P$ un punto y $p, q \in P$ dos puntos distintos. Se dice que p y r son **gemelos** con respecto a q, si se cumple que no existe $q' \in P \setminus \{p, q, r\}$ tal que $p <_q q' <_q r$ o que $r <_q q' <_q p$.*

Para implementar un algoritmo que permita verificar rápidamente si dos puntos p y r son gemelos con respecto al tercer punto q, se hace uso de los treaps, ya que permite localizar, insertar y extraer nodos de forma eficiente. Al considerarse el treap T_q, donde el conjunto de nodos es $P \cup \{r\}$, si se cumple que p y r son las llaves mayor y menor en T_q, o que p es el sucesor o predecesor de r, entonces se satisface que p y q son gemelos con respecto a q.

El siguiente algoritmo es una prueba que permite determinar si dado un treap T_q, y dos puntos $p \in P$ y $r \notin P$ son gemelos con respecto a q, donde el conjunto de puntos $P \setminus \{q\}$ es el conjunto de nodos de T_q.

Algoritmo 3.3 Prueba para determinar si dos puntos p y r son gemelos en un treap T_q

1: **procedure** TRYTWIN(p, r, T_q)
2: Insertar r en T_q
3: p_{min} := Mínimo nodo en T_q
4: p_{man} := Máximo nodo en T_q

5: **if** $\left(\begin{array}{l}(p = p_{min} \text{ and } r = p_{max}) \text{ or} \\ (p = p_{max} \text{ and } r = p_{min}) \text{ or} \\ (p \neq p_{min} \text{ and } r \text{ es predecesor de } p \text{ en } T_q) \text{ or} \\ (p \neq p_{max} \text{ and } r \text{ es sucesor de } p \text{ en } T_q)\end{array}\right)$ **then**
6: Extraer r de T_q
7: **return** True
8: **else**
9: Extraer r de T_q
10: **return** False
11: **end if**
12: **end procedure**

En el Algoritmo 3.3, se extrae el punto r del treap T_q, debido a que no necesariamente aún se puede determinar si r es un punto adecuado para ser candidato a ser la duplicación de p. Esto último se debe a que puede existir otro punto $q' \in P$ distintos de los ya mencionados tal que p y r no sean gemelos con respecto a q'.

El siguiente paso es utilizar los Algoritmos 3.2 y 3.3, de tal manera que se pueda determinar si el punto visible r que se logra encontrar dentro de una región determinada por p y $R = (q_1, q_2, q_3, q_4)$, cumple ser gemelo de p para todos los puntos $P \setminus \{p\}$. Para ello se tiene en cuenta el conjunto $\mathscr{T}$ conformado por todos los treaps T_q para todo $q \in P$, salvo el treap T_p.

Algoritmo 3.4 Algoritmo para hallar un punto visible r y gemelo de p dentro de una región $R = (q_1, q_2, q_3, q_4)$ con respecto a $\mathscr{T}$.

1: **procedure** FINDHALVINGTWIN($p, R, \mathscr{T}$)
2: $r :=$ CANDIDATEVECTOR(p, q_1, q_2, q_3, q_4)
3: **for** $T \in \mathscr{T}$ **do**
4: **if** p no es el origen de T **then**
5: **if not** TRYTWIN(p, r, T) **then**
6: **return** None
7: **end if**
8: **end if**
9: **end for**
10: **return** r
11: **end procedure**

En el caso favorable de que se logre encontrar un punto r que sea buen candidato para duplicar a p, es necesario insertar r en todos los treaps $T \in \mathscr{T}$, crear el treap T_r , considerar ahora que $T_r \in \mathscr{T}$, definir además un nuevo conjunto de puntos Q que almacene a r y a los demás puntos que se vayan duplicando, y constantemente ir actualizando todas las regiones por donde pasan líneas bisectoras. De esta manera se logrará hacer la duplicación de conjuntos de puntos de orden impar en posición general, tal como se hace en la construcción matemática de el Capítulo 2.

Algoritmo 3.5 Algoritmo para duplicar un conjunto de puntos impar en posición general P

```
 1: procedure DUPLICATEODDPOINTS(P)
 2:     procedure UPDATEREGION(p,R)
 3:         if (R[1] tiene un gemelo r ∈ Q  and rpR[1] > 0) then
 4:             R[1] := r
 5:         end if
 6:         if (R[2] tiene un gemelo r ∈ Q  and R[2]pr > 0) then
 7:             R[2] := r
 8:         end if
 9:         if (R[3] tiene un gemelo r ∈ Q  and R[3]pr > 0) then
10:             R[3] := r
11:         end if
12:         if (R[4] tiene un gemelo r ∈ Q  and rpR[4] > 0) then
13:             R[4] := r
14:         end if
15:         return R
16:     end procedure
17:     n := Tamaño de P
18:     Q := Lista vacía
19:     𝒯 := Lista tal que 𝒯[i] := T_{P[i]}
20:     ℛ := Lista tal que ℛ[i] := REGION(P[i],P)
21:     for i = 1  to n do
22:         p := P[i]
23:         R := UPDATEREGION(p,ℛ[i])
24:         r := FINDHALVINGTWIN(p,R,𝒯)
25:         while r = None do
26:             for q ∈ P do
27:                 q := 2q
```

```
28:             end for
29:             r := FindHalvingTwin(p, R, 𝒯)
30:         end while
31:         Q[i] := r
32:         for T ∈ 𝒯 do
33:             insertar r en T
34:         end for
35:         𝒯[n + i] := T_r
36:         for q ∈ P do
37:             insertar q en 𝒯[n + i]
38:         end for
39:         P[n + i] := r
40:     end for
41:     return P
42: end procedure
```

En el Algoritmo 3.5 se considera un procedimiento interno llamado UpdateRegion el cual corresponde a la actualización de las regiones mencionada anteriormente. Se debe realizar tal actualización ya que al haber nuevos puntos en los treaps, las regiones previamente calculadas pueden no ser las adecuadas para la siguiente iteración. Lo que retorna este procedimiento es un conjunto de puntos P, el cual tiene como su primera mitad al conjunto P original, y en su segunda mitad tiene a todos sus respectivos gemelos.

3.3.2. Duplicación de un Conjunto Par de Puntos

Como ya se mencionó al inicio de este capítulo, para un conjunto de puntos par en posición general no siempre existe un emparejamiento de líneas bisectoras. En la Figura 3.4 se presenta un ejemplo de un conjunto de cuatro puntos en posición general al que no se le puede determinar un emparejamiento de líneas bisectoras, debido a que existe un punto al que no se le puede asociar una línea bisectora única.

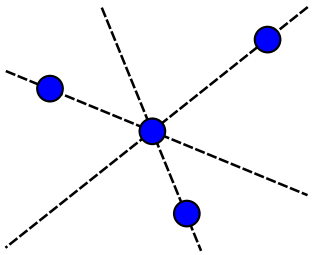

Figura 3.4: Solo se pueden trazar 3 líneas bisectoras para este conjunto de cuatro puntos en posición general.

Por tal motivo, el procedimiento para hacer la duplicación recibirá como entrada un conjunto de puntos par en posición general P como una lista, tal que existe un emparejamiento de líneas bisectoras el cual será asumido como una lista de índices M de la misma longitud que P tal que la línea l_i que pasa por $P[i]$ y $P[M[i]]$ es la línea bisectora asociada a $P[i]$ en el emparejamiento.

La estrategia para la implementación consiste básicamente en duplicar dos veces cada punto de $P[i]$ en los puntos r_1 y r_2, llamados **punto izquierdo** y **punto derecho** de $P[i]$ respectivamente, tal que l_i pasa por $r_1, P[i], r_2$ y $P[M[i]]$ en ese orden. De tal modo que para todos los puntos considerados en esa iteración se cumpla que tanto $P[i]$ y r_1 como $P[i]$ y r_2 sean gemelos.

El conjunto de puntos que se considera por iteración es todo punto $P[j]$ con $j > i$, ya que todos los puntos $P[k]$ con $k \leq i$ pasan a la marginalidad por haber sido o estar siendo duplicados dos veces por un nuevo par de puntos, y adicionalmente se debe cumplir que $j \neq M[i]$, debido a la colinealidad que se menciona en el párrafo anterior. Adicionalmente se considera un conjunto de puntos Q como lista, el cual va almacenando todos los puntos izquierdos y derechos de las iteraciones.

Para los puntos que se considera en la descripción anterior, se tienen en cuenta sus treaps para determinar si las parejas de puntos $P[i], r_1$ y $P[i], r_2$ son gemelos con el Algoritmo 3.3, solo que hay que tener ciertas precauciones con los puntos que se les inserten a los treaps debido a que hay cuatro puntos colineales durante cada iteración. Con respecto a los puntos considerados en P y todos los puntos que están en Q, salvo el punto izquierdo de $P[j]$ donde $i = M[j]$, se considera

que los nodos de todos los treaps sean todos esos puntos (salvo sus respectivos orígenes) incluyendo a $P[i]$. En caso de existir en Q el punto izquierdo de $P[j]$, donde $i = M[j]$, al treap considerado para tal punto, se le insertan todos los puntos ya mencionados, salvo el punto $P[i]$ ya que el que hará el rol de este punto será el punto derecho de $P[j]$.

Al tenerse en cuenta que no siempre se logra que los puntos $P[i], r_1$ y $P[i], r_2$ sean gemelos para todos los treaps, entonces lo que se hace es duplicar las coordenadas de todos los puntos, para así poder escoger unos nuevos puntos r_1 y r_2 que estén aún más cerca de $P[i]$ en proporción a como lo estaban antes.

Así al final de todas las iteraciones, se tendrá que el conjunto de puntos Q será entonces aquel que duplicó a P. A continuación se presenta formalmente este mismo proceso para obtener la duplicación de tal conjunto de puntos P.

Algoritmo 3.6 Algoritmo para duplicar un conjunto de puntos par en posición general P.

```
1:  procedure DUPLICATEPAIRPOINTS(P, M)
2:      n := Tamaño de P
3:      Q := Lista vacía
4:      T_1 := Lista tal que T_1[i] := T_{P[i]}
5:      T_2 := Lista vacía
6:      T_3 := Lista vacía
7:      for i = 1 to n do
8:          d :=  Es el valor absoluto del máximo común divisor entre la abscisa y
                  la ordenada del punto P[M[i]] − P[i]
9:          if d = 1 then
10:             p_1 := (−P[M[i]] + P[i])/d + P[i]
11:             p_2 := (P[M[i]] − P[i])/d + P[i]
12:         else
13:             for all q ∈ P ∪ Q do
14:                 q := 2q
15:             end for
16:             p_1 := (−P[M[i]] + P[i])/2 + P[i]
17:             p_2 := (P[M[i]] − P[i])/2 + P[i]
18:         end if
19:         procedure DUPLICATEGRID(p, T)
20:             while not ( TRYTWIN(p, p_1, T) and
                            TRYTWIN(p, p_2, T) ) do
```

```
21:                 for all q ∈ P ∪ Q do
22:                     q := 2q
23:                 end for
24:                 p_1 := (2p_1 − P[i])/2 + P[i]
25:                 p_2 := (2p_2 − P[i])/2 + P[i]
26:             end while
27:         end procedure
28:         for all T ∈ 𝒯_1 ∪ 𝒯_2 do
29:             if P[M[i]] no es el origen de T then
30:                 DuplicateGrid(P[i], T)
31:             end if
32:         end for
33:         m := Tamaño de 𝒯_3
34:         for j = 1 to m do
35:             if P[i] = P[M[j]] then
36:                 DuplicateGrid(Q[j + n], 𝒯_3[j])
37:             else
38:                 DuplicateGrid(P[i], 𝒯_3[j])
39:             end if
40:         end for
41:         Insertar p_2 en T_{p_1}
42:         Insertar p_1 en T_{p_2}
43:         for j = (i + 1) to n do
44:             Insertar P[j] en T_{p_1} y en T_{p_2}
45:             Insertar p_1 y p_2 en 𝒯_1[j]
46:             Extraer P[i] de 𝒯_1[j]
47:         end for
48:         for all q ∈ Q do
49:             Insertar q en T_{p_1} y en T_{p_2}
50:         end for
51:         Extraer P[M[i]] de T_{p_1}
52:         for all T ∈ 𝒯_2 ∪ 𝒯_3 do
53:             Insertar p_1 y p_2 en T[j]
54:             Extraer P[i] de T[j]
55:         end for
56:         𝒯_2[i] := T_{p_2}
57:         𝒯_3[i] := T_{p_1}
58:         Q[i] := p_1
59:         Q[i + n] := p_2
60:     end for
61:     return Q
62: end procedure
```

El conjunto de puntos Q que retorna el Algoritmo 3.6 es tal que su primera mitad de los puntos que hay en Q es gemela de la segunda mitad. Esto se debe a que todo punto de P era gemelo tanto del punto de la izquierda como el de la derecha, y una vez pasando tal punto a la marginalidad, los puntos izquierdos y derechos se vuelven gemelos con respecto a todos los otros puntos de Q, salvo algunas pequeñas excepciones.

Nótese que tanto el procedimiento DUPLICATEODDPOINTS como DUPLICATEPAIRPOINTS, retornan un conjunto de puntos par en posición general Q tal que su primera mitad de puntos es gemela con su segunda mitad de puntos respectivamente. Esta propiedad es muy interesante, ya que permite hacer una construcción de un emparejamiento de líneas bisectoras y así poder construir tal lista M, la cual es de gran importancia en caso de que se desee hacer una duplicación del conjunto Q posteriormente.

Lo anterior se debe a que todos los gemelos se construyeron sobre una línea bisectora previamente establecida. Siendo $Q = \{q_1, \ldots, q_{2n}\}$, donde q_i y q_{i+n} son gemelos para todo otro punto en Q, se tiene que la línea dirigida l_i que pasa por q_i primero y luego por q_{i+n} es una línea bisectora de Q. Para encontrar una segunda línea bisectora que pase por alguno de estos dos puntos, solo basta con girar en sentido contrarreloj la línea l_i sobre el punto q_i hasta que la línea l_i pase por un nuevo punto q_j, sea l_i' esta nueva línea. De esta manera si se tiene que la línea l_i' pasa primero por q_i y luego por q_j, entonces se considera l_i como la línea correspondiente a q_i en el emparejamiento, es decir, $M[i] = i+n$ y se considera a la línea dirigida l_i'' que pasa primero por q_{i+n} y luego por q_j como la línea bisectora correspondiente a q_{i+n} en el emparejamiento, es decir, $M[i+n] = j$. En caso de que la línea l_i' pase primero por q_j y luego por q_i, entonces la línea l_i' es la línea que le corresponde a q_i en el emparejamiento, es decir que $M[i] = j$, y la línea l_i es la línea que le corresponde a q_{i+n} en el emparejamiento, esto es $M[i+n] = i$.

El emparejamiento anterior se sigue textualmente de la construcción del emparejamiento presentado en el artículo [ÁCFM$^+$10]. Una implementación en coordenadas enteras de este procedimiento se sigue a continuación.

Algoritmo 3.7 Algoritmo para encontrar un emparejamiento de lineas bisectoras para un conjunto par de puntos en posición general

```
1:  procedure GetHalvingLineMatching(P)
2:      n := El tamaño de P
3:      M := Una lista vacía de tamaño n
4:      for k = 1 to n/2 do
5:          p_1 := P[k+n]
6:          p_2 := −P[k+n] + 2P[k]
7:          q_1 := p_2
8:          q_2 := p_1
9:          for i = 1 to n do
10:             if (P[k]q_1P[i] > 0 and P[k]P[i]p_1 > 0) then
11:                 q_1 := P[i]
12:                 i_1 := i
13:             else if (P[k]q_2P[i] > 0 and P[k]P[i]p_2 > 0) then
14:                 q_2 := P[i]
15:                 i_2 := i
16:             end if
17:         end for
18:         r := −q_2 + 2P[k]
19:         if P[k]rq_1 > 0 then
20:             M[k] := k + n
21:             M[k+n] := i_1
22:         else if P[k]rq_1 < 0 then
23:             M[k] := i_2
24:             M[k+n] := k + n
25:         end if
26:     end for
27:     return M
28: end procedure
```

Finalmente con este último algoritmo es posible hacer cuantas veces se desee una duplicación de puntos en posición general, partiendo de un conjunto impar o de un conjunto par al cual se le tenga un emparejamiento por líneas bisectoras de antemano. Existen muchos otros algoritmos que permiten determinar y encontrar emparejamientos por líneas bisectoras, y en caso de ser posible encontrar tal emparejamiento, se podrá empezar a duplicar cuantas veces se quiera a partir de tal conjunto.

Capítulo 4

Conclusiones

4.1. El Propósito del Algoritmo de Duplicación

Las implementaciones de todos los algoritmos que se presentaron en el Capítulo 3 se hicieron en el lenguaje de programación Python. Estos algoritmos permiten seguir duplicando de forma indefinida cualquier conjunto de puntos en posición general de tamaño impar, o de tamaño par para el cual exista un emparejamiento de líneas bisectoras. Lo cual resulta ser una herramienta muy útil para duplicar el mejor conjunto de puntos P en posición general, esto es, que genere la mejor cota superior para q_* que se tenga a la fecha, y guardarse en Q. Esto último se haría con el fin de aplicarse toda clase de métodos y estrategias sobre Q, para optimizarlo de tal forma que este nuevo conjunto tal vez pueda proporcionar una mejor cota superior. El conjunto Q es un buen punto de partida para hacer esta estrategia, ya que por el Teorema 2.1 se garantiza que el número de cruce $\overline{cr}(Q)$ depende solamente de $\overline{cr}(P)$ y el tamaño de P. Con esta nueva utilidad que tiene la construcción matemática presentada en el Capítulo 2, se ha logrado mejorar la cota superior, la cual es generado por un conjunto de 237 puntos en posición general y cuyo valor es aproximadamente 0.380452502.

Antes de lograrse esta implementación, la mejor cota superior para q_* que se tenía hasta el momento era 0.380473, la cual fue hallada en el año 2014 por

Fabila y sus colaboradores en [FL14]. El método consistía básicamente en mover de forma aleatoria puntos de un conjunto en posición general, de tal forma que el conjunto se actualiza cada vez que el número de cruce sobre ese conjunto sea mejorado. Esta cota superior fue generada por un conjunto de puntos en posición general de tamaño 75.

4.2. El Tiempo que se Tarda en Duplicar

Para el estudio de los tiempos de ejecución, se consideraron los conjuntos de puntos en posición general $P_3, P_5, \ldots, P_{149}$, donde cada P_i es de tamaño i y cada punto es escogido de forma aleatoria. A estos conjuntos se les aplicó el Algoritmo 3.5, dando como resultado los conjuntos de puntos $P_6, P_{10}, \ldots, P_{298}$. En la Figura 4.1 se presenta una gráfica de los tamaños de los conjuntos resultantes y el tiempo que tardó en construirlos usando un computador de rendimiento promedio.

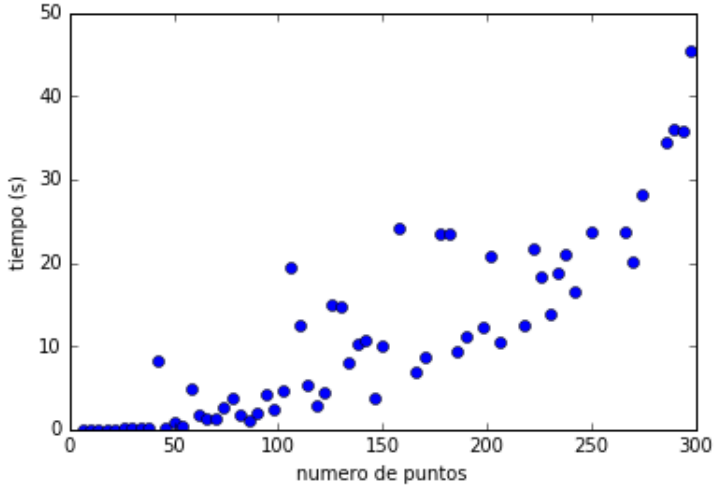

Figura 4.1: Gráfica de duplicación de conjuntos impares de puntos versus su tiempo de ejecución

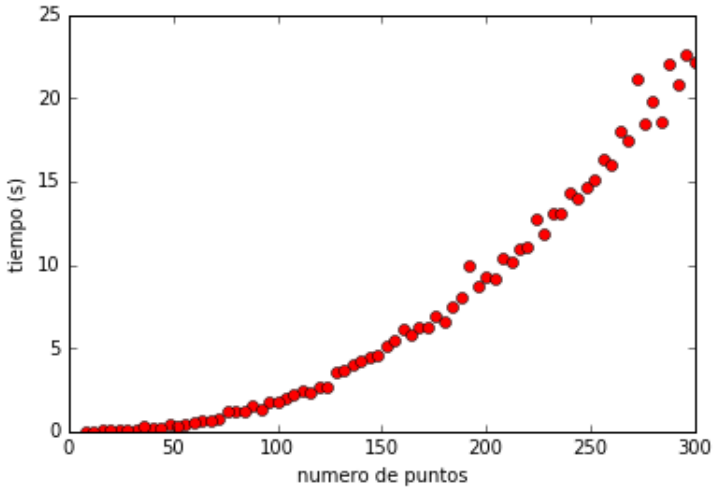

Figura 4.2: Gráfica de duplicación de conjuntos pares de puntos versus su tiempo de ejecución

Luego se consideran los conjuntos $P_6, P_{10}, \ldots, P_{150}$ y empleando el Algoritmo 3.6 se termina de generar todos los conjuntos de puntos de tamaño par hasta P_{300}, teniendo en cuenta que los emparejamientos de líneas bisectoras se pueden construir con el Algoritmo 3.7. En la Figura 4.2 se presenta una gráfica de los tamaños de los conjuntos resultantes y el tiempo que tardó en construirlos usando un computador de rendimiento promedio.

Se puede apreciar de las gráficas de las Figuras 4.1 y 4.2, que el Algoritmo 3.5 tiende a tardar al menos el doble de tiempo que tarda el Algoritmo 3.6 para duplicar conjuntos de casi el mismo tamaño. Esto es, el duplicar conjuntos de tamaño impar tiende a tardar el doble de tiempo que tarda el duplicar un conjunto de tamaño par al que se le puede calcular rápidamente un emparejamiento de líneas bisectoras.

4.3. Crecimiento de la Cuadrícula

Pero ¿cuánto debe de ser el tamaño de una cuadrícula para poder contener los conjuntos de puntos que resultan al usar algoritmos de duplicación?. La cuadrícu-

Conjunto	Tiempo (s)	Tamaño de Cuadrícula
P_6	0.00323	4
P_{12}	0.0167	22
P_{24}	0.0663	1600
P_{48}	0.336	410368
P_{96}	1.4	840439808
P_{192}	7.64	27539531726848
P_{384}	39.5	3609661502501814272
P_{768}	191	242240282857429913934233600
P_{1536}	886	260103523161612727739907004927311872

Tabla 4.1: Tiempo y valor de n para el tamaño de la cuadrícula generada por la duplicación de P_3 hasta P_{1536}.

la se considerará de tamaño n, donde n es la máxima diferencia entre la diferencia mas grande que hay entre las abscisas y la diferencia más grande que hay entre las ordenadas con respecto a los puntos. Para ello a partir del conjunto de puntos $P_3 = \{(0,0),(4,0),(0,4)\}$, el cual está en una cuadrícula de tamaño 4, se usaron los algoritmos de duplicación hasta obtenerse el conjunto P_{1536} y el tamaño de la cuadrícula es 260103523161612727739907004927311872.

En la Tabla 4.1 se muestra el tiempo requerido y como es el comportamiento del tamaño de la cuadrícula con respecto al tamaño de los conjuntos de puntos que resultan al ser duplicados en el ejemplo anterior. El crecimiento de la cuadrícula se debe a que, para cada punto p que hay en un conjunto P resultante de cualquiera de los algoritmos de duplicación, se satisface que exista un punto q en P que sea gemelo de p con respecto a casi todos los demás puntos de P. En la Figura 4.3 se puede apreciar todos los conjuntos de puntos que se generaron a partir de P_3.

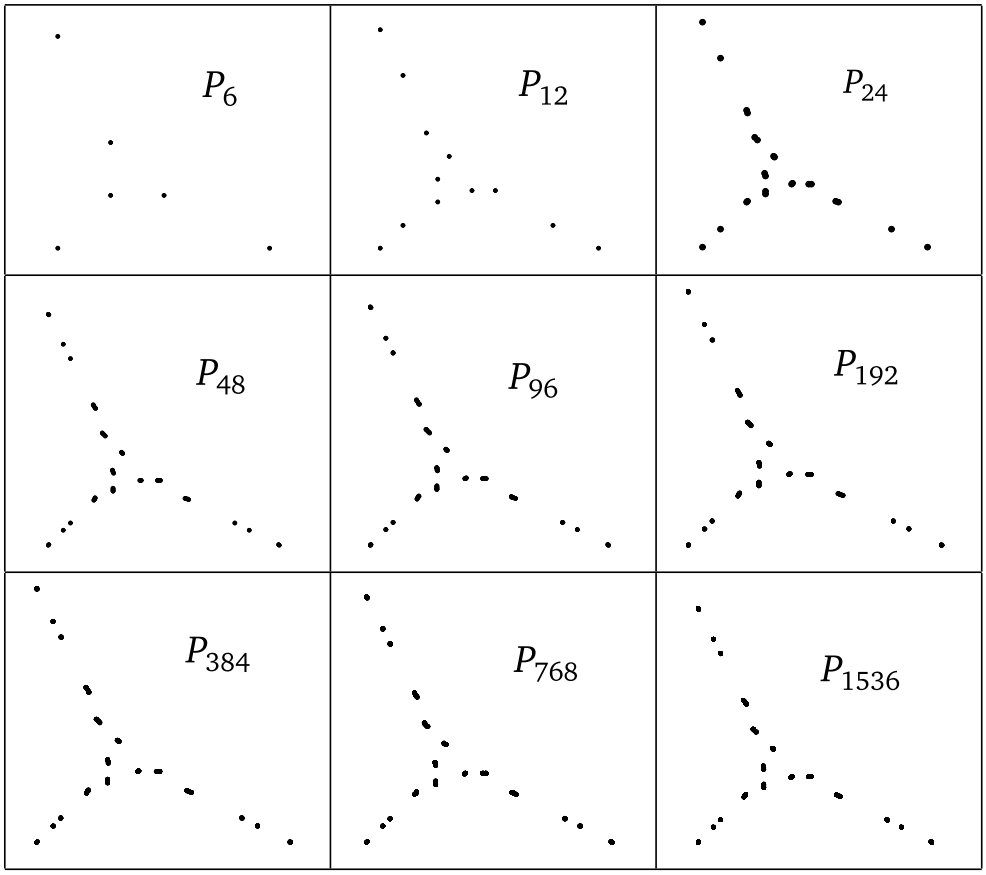

Figura 4.3: Las duplicaciones de P_3.

Bibliografía

[AAK06] O. Aichholzer, F. Aurenhammer, and H. Krasser. On the crossing number of complete graphs. *Computing*, 76(1-2):165–176, 2006.

[ÁCFM+10] B. M. Ábrego, M. Cetina, S. Fernández-Merchant, J. Leaños, and G. Salazar. 3-symmetric and 3-decomposable geometric drawings of K_n. *Discrete Appl. Math.*, 158(12):1240–1458, 2010.

[ÁFM07] Bernardo M. Ábrego and Silvia Fernández-Merchant. Geometric drawings of K_n with few crossings. *J. Combin. Theory Ser. A*, 114(2):373–379, 2007.

[Bla17] Wilhelm Blaschke. Über affine geometrie xi: Lösung des "vierpunktproblems" von sylvester aus der theorie der geometrischen wahrscheinlichkeiten. *Leipziger Berichte*, 69:436–453, 1917.

[Cro85] M. W. Crofton. Probability. In *Encyclopedia Britannica*, volume 19, pages 768–788. 9th edition, 1885.

[FL14] Ruy Fabila-Monroy and Jorge López. Computational search of small point sets with small rectilinear crossing number. *Journal of Graph Algorithms and Applications*, 18(3):393–399, 2014.

[GP80] Jacob E. Goodman and Richard Pollack. On the combinatorial classification of nondegenerate configurations in the plane. *J. Combin. Theory Ser. A*, 29(2):220–235, 1980.

[GP83] Jacob E. Goodman and Richard Pollack. Multidimensional sorting. *SIAM J. Comput.*, 12(3):484–507, 1983.

[LVWW04] László Lovász, Katalin Vesztergombi, Uli Wagner, and Emo Welzl. Convex quadrilaterals and k-sets. In *Towards a theory of geometric graphs*, volume 342 of *Contemp. Math.*, pages 139–148. Amer. Math. Soc., Providence, RI, 2004.

[SW94] Edward R. Scheinerman and Herbert S. Wilf. The rectilinear crossing number of a complete graph and Sylvester's "four point problem" of geometric probability. *Amer. Math. Monthly*, 101(10):939–943, 1994.

[Tur77] Paul Turán. A note of welcome. *Journal of Graph Theory*, 1(1):7–9, 1977.

[Wag03] Uli Wagner. On the rectilinear crossing number of complete graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003)*, pages 583–588. ACM, New York, 2003.

[Wel86] Emo Welzl. More on k-sets of finite sets in the plane. *Discrete Comput. Geom.*, 1(1):95–100, 1986.