Universidad Autónoma de San Luis Potosí

Facultad de Ciencias

Aumentando secuencias circulares en busca de dibujos con pocos cruces

Tesis que para obtener el título de

Maestro en Ciencias Aplicadas

presenta

Antonio de Jesús Torres Hernández

bajo la dirección de

Dr. Gelasio Salazar Anaya

San Luis Potosí, San Luis Potosí febrero de 2018.

Agradecimientos

Me gustaría que estas líneas sirvieran para expresar mi más profundo y sincero agradecimiento a todas aquellas personas que con su ayuda han colaborado en la realización del presente trabajo.

A la Universidad Autónoma de San Luis Potosí por darme la oportunidad de estudiar en tan importante centro académico.

Al Dr. Gelasio Salazar Anaya por brindarme su conocimiento, apoyo y paciencia.

A los Doctores Edgardo Ugalde Saldaña y Cesar Hernández Vélez por su amistad y asesoria en la carrera así como también por el tiempo invertido en este trabajo.

A mis padres y mis hermanas por su apoyo y guia en esta etapa y a lo largo de mi vida.

A mis sobrinos para los que espero mis logros sean solo una cota inferior.

Y por ultimo a tí Denae por estár siempre a mi lado.

Índice general

Introducción

Al dibujar la gráfica completa simple K_n con n vértices en el plano, no es difícil notar que la cantidad de cruces que se generan entre los pares de aristas depende de la ubicación de los vértices. Pero dado un entero n, ¿cuál será el mínimo número de cruces que podemos encontrar en un dibujo de K_n, considerando el conjunto de todos los dibujos posibles en el plano?. Esta es una variante del problema de la fabrica de ladrillos de Turán, un problema clásico tanto en la geometría combinatoria como en la teoría de gráficas, y que es conocido también como el problema del número de cruce.
Este problema fue gestado durante la segunda guerra mundial y desde entonces a dado a los matemáticos mucho en que pensar, si bien, este problema permanece abierto hasta hoy día, a lo largo de los años se han generado una gran variedad de trabajos y publicaciones inspiradas en el, así como también se han planteado nuevos e interesantes problemas. Uno de estos problemas es el problema del número de cruce rectilíneo. En esta versión, las aristas de K_n se restrigen a segmentos de recta. También en esta versión se han realizado importantes avances, algunos de los cuales utilizan dibujos de gráficas completas para de forma asintotica acotar el número de cruce en general. Pero encontrar estos dibujos es un problema no trivial.
En este trabajo enfocaremos nuestros esfuerzos en la busqueda de nuevos dibujos cuya gráfica geométrica tenga pocos cruces, mediante el método de dibujos aumentantes ayudándonos algoritmos computacionales.

Dividiremos esta tesis en cuatro capítulos. En el primer capítulo expondremos el problema, haremos un breve recorrido por su interesante historia y desarrollo, mencionaremos varios de los avances más importantes que se han logrado, así como algunas publicaciones que fueron de vital importancia para nosotros durante el proceso de esta tesis. Veremos también como este problema guarda relación con otros problemas importantes, como el problema de Sylvester o con el conteo de cuadriláteros convexos en un conjunto de puntos en el plano. Hablaremos de

la importancia y aportaciones de las construcciones aumentantes y las secuencias circulares, estructuras que utilizamos ampliamente en este trabajo.

En el segundo capítulo mostramos una forma de aumentar las secuencias circulares provenientes de dibujos conocidos que tengan pocos cruces, en busca de nuevos dibujos, de tal forma que en estos la cantidad de cruces nuevos sea la menor posible. Una vez encontradas estas secuencias hemos de proceder a su geometrización, es decir a encontrar los conjuntos de puntos que las generan. En este capitulo también se expone una técnica para geometrizar las secuencias provenientes de dibujos mediante la duplicación de algunos vértices y se muestran algunas tablas de conjuntos que se generaron a través de estas técnicas.

En el tercer capítulo daremos de forma explícita dos nuevos conjuntos cuyo número de cruce rectilíneo es mejor que el de los conjuntos de las mismas cardinalidades conocidos hasta ahora, los cuales fueron generados utilizando las ideas del capítulo 2. Hablaremos también de los posibles caminos a seguir en trabajos a futuro, y plantearemos preguntas que surgieron durante la investigación.

Para finalizar, el cuarto capítulo contiene los códigos de programación implementados en los lenguajes Python y C que utilizamos para calcular el número de cruce, aumentar secuencias de dibujos conocidos y para geometrizar aquellas que provenían de la duplicación de elementos de una secuencia.

CAPÍTULO 1

El Número de Cruce Rectilíneo

1.1. El Problema del Número de Cruce.

En las Matemáticas, algunos problemas famosos, además de su importancia y belleza, vienen acompañados con una historia o anécdota que interesa y motiva a quien los estudia. Este es el caso del problema que trataremos en este trabajo, el problema del número de cruce. Problema expuesto por el matemático hungaro Paul Turán en los $70's$, pero que fue gestado a finales de la Segunda Guerra Mundial. Como el mismo Turán describe en una carta previa enviada al también matemático Richard Guy.

In 1944 *our labor combattation had the extreme luck to work thanks to some very rich comrades in a brick factory near Budapest. Our work was to bring out bricks from the ovens where they were made and carry them on small vehicles which run on rails in some of several open stores which happened to be empty. Since one could never be sure which store will be available, each oven was connected by rail with each store. Since we had to settle a fixed amount of loaded cars daily it was our interest to finish it as soon as possible. After being loaded in the (rather warm) ovens the vehicles run smoothly with not much effort; the only trouble arose at the crossing of two rails. Here the cars jumped out, the bricks fell down; a lot of extra work and loss of time arose. Having this experience a number of times it occurred to me why on earth did they build the rail system so uneconomically; minimizing the number of crossings the production could be made much more economical.*

Paul Turán's letter to Richard Guy.
February 1968.

En 1944 *gracias a algunos camaradas muy ricos, nuestra fuerza de trabajo tuvo la suerte de trabajar en una fábrica de ladrillos cerca de Budapest. Nuestro trabajo consistía en sacar ladrillos de los hornos donde se hacían y llevarlos en pequeños vehículos que corrían sobre rieles a patios de almacenaje vacíos. Puesto que uno nunca podía estar seguro que almacén estaba disponible, cada horno fue conectado mediante rieles a cada almacén. Dado que teníamos que cumplir con una cantidad fija de cargas diariamente, era nuestro interés terminar lo antes posible. Después de ser cargados en los hornos (bastante calientes) los vehículos se deslizaban suavemente sin mucho esfuerzo; el único problema surgía en el cruce entre dos carriles. Ahí los coches saltaban y los ladrillos se caían; lo cual generaba un montón de trabajo extra y pérdida de tiempo. Al sucederme esto varias veces, pensé ¿por qué rayos construyeron el sistema ferroviario tan ineficientemente? Minimizando el número de cruces, la producción podría hacerse mucho más económica.*

Carta de Paúl Turán a Richard Guy.
Febrero de 1968.

Podemos describir este problema en términos matemáticos, utilizando algunos conceptos de teoría de gráficas.

Definición 1.1. Una *gráfica* es un par ordenado $G = (V(G), E(G))$ donde $V(G)$ es un conjunto no vacío de objetos, llamados *vértices*, y $E(G)$ es un conjunto

(posiblemente vacío) de parejas no ordenadas de elementos de $V(G)$ llamados *aristas*.

Si $e = \{u, v\}$ es una arista en la gráfica G, entonces decimos que u y v son *vértices adyacentes*, y la arista e es incidente a cada uno de ellos. Diremos que una gráfica es *completa* si todos sus vértices son adyacentes entre sí. *El orden de una gráfica* es igual a la cardinalidad del conjunto de vértices. Denotaremos por K_n a la gráfica completa de orden n.
Una gráfica G es *k-partita*, $k \geq 1$, si es posible particionar $V(G)$ en k subconjuntos $V_1, V_2, ..., V_k$ tal que toda arista $\{x, y\} \in E(G)$ satisface que $x \in V_i$ y $y \in V_j$ para $1 \leq i \neq j \leq k$. Cuando $k = 2$ diremos que G es *bipartita*, y denotaremos por $K_{m.n}$ a la gráfica bipartita completa dónde $\mid V_1 \mid = m$ y $\mid V_2 \mid = n$.
Así el problema de la fábrica de ladrillos de Turán, se traduce a encontrar el mínimo número de cruces necesarios para dibujar a $K_{m,n}$ en el plano. Donde el subconjunto de los m vértices corresponderán a los hornos, mientras que el conjunto de n vértices, serán los patios de almacenamiento.

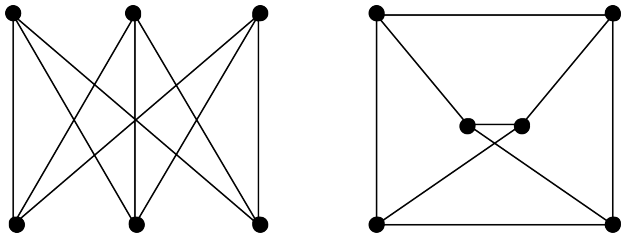


Figura 1.1: Dos dibujos de $K_{3,3}$

En los $50's$, el topólogo Kazimierz Zarankiewicz publicó una "solución" al problema, pero su demostración tenía una falla, misma que fue notada, independientemente, por los matemáticos Paul Kainen y Gerhard Ringel once años después. Sin embargo, este trabajo estableció la siguiente cota superior.

$$cr(K_{m,n}) \leq \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{m}{2} \right\rfloor \left\lfloor \frac{m-1}{2} \right\rfloor.$$

El problema del número de cruce fue conocido ampliamente por la comunidad matemática de la época, pero también interesó a personajes carentes de formación formal en matemáticas. Este es el caso del artista plástico Anthony Hill, cuya obra se centró principalmente en las propiedades estéticas y matemáticas de objetos geométricos y combinatorios. Él estudio por su cuenta varios problemas matemáticos, en particular, el problema de la fabrica de ladrillos. Hill dibujó conjuntos de puntos, unidos mediante curvas, e investigó cuantas veces estas curvas se cruzaban entre sí. En términos matemáticos, el problema concerniente al número de cruce de la gráfica completa K_n.

Dado un conjunto finito P de puntos en el plano en *posición general*, es decir, no tres de ellos colineales, denotamos por $cr(P)$ al número de cruces en la gráfica completa cuyo conjunto de vértices es P, y cuyas aristas son curvas suaves.

Definición 1.2. El *número de cruce de* n, es el mínimo número de cruces determinados por cualquier conjunto de n puntos en el plano, es decir, $cr(n) = min_{|P|=n} cr(P)$.

Utilizando una variante de una construcción propuesta por Hill, Blazêk and M.Koman [6] probaron la siguiente cota superior para el número de cruce

$$cr(n) \leq \frac{1}{4} \left\lfloor \frac{n}{2} \right\rfloor \left[\frac{n-1}{2} \right] \left[\frac{n-2}{2} \right] \left[\frac{n-3}{2} \right].$$

1.2. El Número de Cruce Rectilíneo.

Anthony Hill también propuso el caso particular en el que las líneas que unen los puntos en los dibujos son segmentos de recta, generando así el problema del número de cruce rectilíneo.
Dado un conjunto finito P de puntos en el plano en posición general, denotamos por $\overline{cr}(P)$ al número de cruces en la *gráfica geométrica* cuyo conjunto de vértices es P, es decir, la gráfica completa cuyas aristas son segmentos de rectas. Esta gráfica es también llamada *dibujo rectilíneo* de K_n.

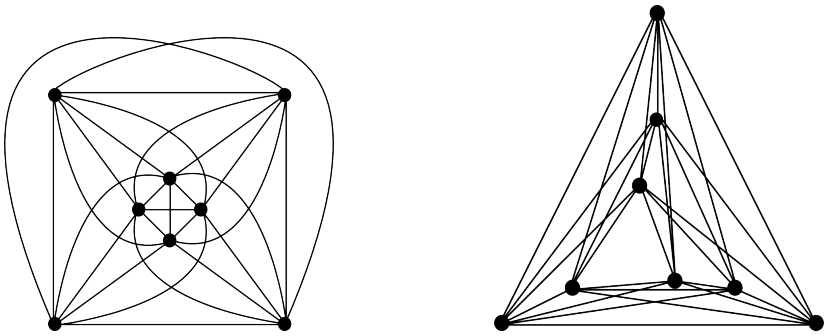
Definición 1.3. El *número de cruce rectilíneo de* n, es el mínimo número de cruces determinado por cualquier conjunto de n puntos en el plano, es decir, $\overline{cr}(n) = min_{|P|=n} \overline{cr}(P)$.

Notemos que el número de cruce rectílineo acota superiormente al número de cruce, es decir $cr(n) \leq \overline{cr}(n)$, esto a razón de la menor libertad que se tiene al restringir las curvas a segmentos de recta. Así el enunciado del problema propuesto por Hill es el siguiente.

Problema 1.4. (Problema del número de cruce rectilíneo) *Dado un número natural* n*, hayar el valor de* $\overline{cr}(n)$.

1.3. Cuadriláteros Convexos y j-Aristas.

Definición 1.5. Dado un conjunto P de puntos en posición general en el plano. El mínimo número de subconjuntos convexos de cuatro elementos cuya envolvente convexa forma un cuadrilátero se conoce como el *número de cuadriláteros*



(a) H_1 (b) H_2

Figura 1.2: Dibujos de K_8 con $cr(H_1) = 18$ y $\overline{cr}(H_2) = 19$.

convexos, y se denota como $\square(P)$.

Definición 1.6. Dado un conjunto P de n puntos en posición general en el plano. Una *j-arista* de P es un par ordenado vu con $v, u \in P$ y $v \neq u$, tal que existen exactamente j puntos de P en el semiplano derecho definido por la línea vu. Si n es par, la línea recta que definen los puntos de una $(n-2)/2-$arista es llamada *recta mediana*. Denotaremos por e_j al número de *j-aristas* de P y una *$i-$arista* con $i \leq j$ será llamada una *$(\leq j)$-arista*.

El número de cuadriláteros convexos $\square(P)$ será de suma importancia a lo largo de esta tesis, pues resulta ser equivalente al número de cruce rectilíneo $\overline{cr}(P)$, ya que dos aristas en un dibujo rectilíneo forman un cruce si, y sólo si, sus cuatro vértices forman un cuadrilátero convexo. Luego, gracias a esta equivalencia, podemos reescribir el número de cruce rectilíneo en función de las *j-aristas* utilizando el siguiente resultado obtenido por Lovász et al [4], e independientemente obtenido por Ábrego y Fernández-Merchant.

Teorema 1.7. *Para todo conjunto P de n puntos en el plano en posición general,*

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

Demostración. Denotemos por $\triangle(P)$ al número de subconjuntos de cuatro elementos en P tales que su envolvente convexa forma un triángulo. Claramente

$$\square(P) + \triangle(P) = \binom{n}{4}. \tag{1}$$

En búsqueda de otra ecuación que relacione también a $\square(P)$ y $\triangle(P)$, contaremos, de dos formas diferentes, las cuátruplas (u, v, w, z) tales que w está en el

semiplano derecho de la línea uv y z está en el semiplano izquierdo. Primero, si el conjunto $\{u, v, w, z\}$ forma un cuadrilátero convexo, entonces podemos ordenar las cuádruplas de cuatro maneras, mientras que si $\{u, v, w, z\}$ forma un triángulo, entonces hay seis posibles formas de ordenar las cuádruplas. Así, el número de cuádruplas ordenadas es $4\square(P)+6\triangle(P)$. Por otro lado, toda $j-arista\ uv$ puede ser completada a una cuádrupla de $j(n-j-2)$ maneras. Entonces tenemos

$$4\square(P) + 6\triangle(P) = \sum_{j=0}^{n-2} e_j j(n-j-2), \tag{2}$$

De (1) y (2) tenemos que

$$\square(P) = \frac{1}{2}\left(6\binom{n}{4} - \sum_{j=0}^{n-2} e_j(n-j-2)j\right),$$

Usando que

$$\sum_{j=0}^{n-2} e_j = n(n-1), \tag{3}$$

podemos escribir

$$6\binom{n}{4} = \sum_{j=0}^{n-2} e_j \frac{(n-2)(n-3)}{4},$$

y obtenemos

$$\begin{aligned}
\square(P) &= \frac{1}{2}\sum_{j=0}^{n-2} e_j \left(\frac{(n-2)(n-3)}{4} - j(n-j-2)\right), \\
&= \frac{1}{2}\sum_{j=0}^{n-2} e_j \left(\frac{(n-2)(n-3)}{4} - \left(\frac{n-2}{2}\right)^2 + j^2 + 2j - nj + \left(\frac{n-2}{2}\right)^2\right), \\
&= \frac{1}{2}\sum_{j=0}^{n-2} e_j \left(\frac{2-n}{4} + \left(\frac{n-2}{2} - j\right)^2\right), \\
&= \frac{1}{2}\sum_{j=0}^{n-2} e_j \left(\frac{2-n}{4}\right) + \frac{1}{2}\sum_{j=0}^{n-2} e_j \left(\frac{n-2}{2} - j\right)^2,
\end{aligned}$$

Usando nuevamente (3) y el hecho de que $e_j = e_{n-2-j}$ tenemos

$$\begin{aligned}
\square(P) &= \frac{n(n-1)(2-n)}{8} + \sum_{j<\frac{n-2}{2}} e_j \left(\frac{n-2}{2} - j\right)^2, \\
&= \sum_{j<\frac{n-2}{2}} e_j \left(\frac{n-2}{2} - j\right)^2 - \frac{3}{4}\binom{n}{3}.
\end{aligned}$$

□

1.4. El Problema de los Cuatro Puntos de Sylvester.

El número de cruce rectilíneo está estrechamente relacionado con un problema muy conocido en la geométria combinatoria, propuesto por el matemático inglés James Joseph Sylvester [7]. En 1864, en la edición de abril del *"Educational Times"*, Sylvester publicó un problema aparentemente sencillo: Probar que la probabilidad de que cuatro puntos formen un cuadrilátero convexo es 3/4 si estos son tomados en un plano indefinido. Rápidamente la comunidad matemática así como el mismo Sylvester, se dieron cuenta que este enunciado fue planteado erróneamente. El problema reside en que en el plano indefinido no existe ninguna medida de probabilidad natural. Como se pensaría el problema fue replanteado en el siguiente enunciado.

Problema 1.8. (Problema de los cuatro puntos de Sylvester) *Sea R un conjunto convexo y acotado en el plano, y sean cuatro puntos escogidos aleatoria, uniforme e independientemente, ¿cuál es la probabilidad, $q(R)$, de que los cuatro puntos formen un cuadrilátero convexo? más aún, ¿para qué R se obtiene la mayor y la menor probabilidad $q(R)$?*

Pasaron cincuenta años hasta que se encontrara una solución. En 1917 Wilhelm Blaschke [8] demostró que para todos los cuerpos convexos y compactos $R \in \mathbb{R}$ el valor máximo de $q(R)$ se alcanza cuando R es un disco, mientras que el valor mínimo se logra cuando R es un triángulo. Este resultado, sin embargo, no aborda completamente la cuestión de los valores extremos de $q(R)$, porque considera sólo las regiones convexas. Es fácil ver que si relajamos la hipótesis de convexidad, entonces el supremo de $q(R)$ es 1. Pues al considerar a R como un arco de circunferencia, cualesquiera cuatro puntos en R forman un cuadrilátero convexo.
Por otro lado, encontrar el ínfimo de los $q(R)$ es un problema que aún hoy permanece abierto. Sin embargo, en 1994 Scheinerman y Wilf [12] probaron una interesante relación entre el ínfimo de $q(R)$ y el número de cruce rectilíneo.

Teorema 1.9. *El siguiente límite existe*

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

.

Demostración. Claramente la sucesión $S_n = \{\overline{cr}(n)/\binom{n}{4}\}$ está acotada superiormente por 1, pues $\overline{cr}(n) \leq \binom{n}{4}$. Basta entonces probar que S_n es monótona no decreciente.

Sea m un entero positivo menor que n, y H un dibujo rectilíneo con n vértices tal que $\overline{cr}(H) = \overline{cr}(n)$. Consideremos todos los subconjuntos I de m puntos de H. Como cada cruce en estos subconjuntos está determinado por cuatro puntos, y los subconjuntos son tomados a su vez de n puntos, tenemos

$$\overline{cr}(H) = \frac{\sum_{|I|=m} \overline{cr}(I)}{\binom{n-4}{m-4}}$$

y como $\overline{cr}(H) = \overline{cr}(n)$ y $\overline{cr}(I) \geq \overline{cr}(m)$ para todo I, entonces

$$\overline{cr}(n) \geq \frac{\binom{n}{m}\overline{cr}(m)}{\binom{n-4}{m-4}}$$

luego, usando la igualdad $\binom{n}{m}/\binom{n-4}{m-4} = \binom{n}{4}/\binom{m}{4}$ y despejando, obtenemos

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

por lo tanto la sucesión S_n es una sucesión monótona no decreciente. Concluimos entonces que v existe. □

Denotemos q_* como el ínfimo de los $q(R)$ sobre todos los conjuntos abiertos R de área finita. Scheinerman y Wilf demostraron la siguiente identidad.

Teorema 1.10.

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

Demostración. Sea R un conjunto abierto y acotado en el plano. Elegimos p_1, p_2, ..., p_n puntos de R tomados independiente, uniforme y aleatoriamente, y sean estos puntos los vértices de un dibujo rectilíneo H de K_n. Consideremos la variable aleatoria

$$X = \sum_{a,b,c,d} I\{p_a, p_b, p_c, p_d\},$$

donde I es una función indicadora que actúa sobre los subconjuntos de puntos de cardinalidad 4, tal que $I\{, ..., \} = 1$ si la envolvente convexa de p_a, p_b, p_c, p_d forma un cuadrilátero y 0 en otro caso. Como el dibujo óptimo de K_n no puede tener más cruces que la media, sacando la esperanza tenemos

$$\overline{cr}(n) \leq E(X) = \binom{n}{4} q(R)$$

para todo n. Luego, dividiendo entre $\binom{n}{4}$ y haciendo tender n al infinito obtenemos que $v \leq q(R)$ para toda n, así $v \leq q_*$.
Para la otra desigualdad, consideremos un dibujo rectilíneo H tal que $\overline{cr}(H) =$

$\overline{cr}(n)$. Sea R_ε el conjunto abierto y disconexo formado al colocar un disco de radio ε centrado en cada vértice del dibujo H. Escogemos ε lo suficientemente pequeño tal que para cualquier elección de n puntos, uno en cada disco, al conectar a pares los vértices por segmentos de recta, el número de cruce es igual a $\overline{cr}(n)$, i.e, todos los dibujos son óptimos. Claramente este ε existe.
Ahora consideremos la siguiente pregunta: Al elegir aleatoria e independientemente cuatro puntos distintos en R_ε. ¿cuál es la probabilidad q de que estos formen un cuadrilátero convexo?.
Por un lado, q es el número de cuadriláteros convexos en el dibujo original dividido por $\binom{n}{4}$. Pero los primeros están en correspondencia uno-uno con los cruces, así hay exactamente $\overline{cr}(n)$ de ellos y tenemos que $q = \overline{cr}(n)/\binom{n}{4}$. Por otro lado, $q(R_\varepsilon)$ es la probabilidad de que cuatro puntos escogidos aleatoria e independientemente en R_ε formen un cuadrilátero convexo. Pero cuatro puntos caerán en distintos discos de R_ε con probabilidad $1 - O(1/n)$. Así

$$q = q(R_\varepsilon) + O(1/n) \geq q_* + O(1/n).$$

Combinando estos hechos acerca de q obtenemos

$$q = \overline{cr}(n)/\binom{n}{4} \geq q_* + O(1/n).$$

Luego haciendo tender n al infinito tenemos que $v \geq q_*$, lo cual concluye la demostración. □

1.5. Secuencias Circulares

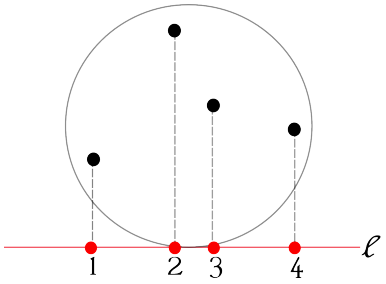
Definición 1.11. Sea $\Pi = \pi_0, \pi_1, ..., \pi_{\binom{n}{2}}$ una sucesión de permutaciones del conjunto $[n] = \{1, 2, 3, ..., n\}$. Diremos que Π es una *secuencia circular* si se cumple;

1) Cualesquiera dos permutaciones consecutivas difieren en exactamente una pareja de elementos adyacentes.

2) $\pi_{\binom{n}{2}}$ es la permutación inversa de π_0, es decir, $\pi_0 = (1, 2, ..., n)$ y $\pi_{\binom{n}{2}} = (n, n-1, ..., 1)$.

Nuestro interés en las secuencias circulares radica en el hecho de que dado un conjunto finito P de n puntos en el plano en posición general , se le puede asociar una secuencia circular Π_P que contendrá toda la información de P.
Esta secuencia circular Π_P se obtiene de la siguiente manera: Como P es finito, comenzamos con un círculo C que contiene al conjunto P en su interior, y una línea dirigida ℓ tangente a C. Rotamos la línea ℓ manteniéndola siempre tangente al círculo C, hasta garantizar que todos los puntos de P tienen diferente

proyección sobre ℓ (siempre podemos ajustar los puntos mediante perturbaciones suficientemente pequeñas de forma que el número de cruces no cambie). Proyectamos P ortogonalmente sobre ℓ y registramos el orden de los puntos de P en ℓ. Esta será la permutación inicial π_0 de Π. Ahora rotamos ℓ sobre C en sentido antihorario y seguimos proyectando P sobre ℓ. Justo después de que dos puntos se traslapen en la proyección, digamos p y q, el orden de P sobre ℓ cambia. Este nuevo orden de P sobre ℓ será π_1. Notemos que π_1 se obtuvo de π_0 mediante la transposición de dos elementos adyacentes. Continuamos con este procedimiento, rotando ℓ sobre C y registrando las permutaciones correspondientes de P, hasta completar media vuelta a C. En este momento el orden de P sobre ℓ será el inverso al original. Más aún, habrán transcurrido exactamente $\binom{n}{2}$ transposiciones, una por cada par de puntos. A lo largo de este trabajo, para toda secuencia circular, la línea ℓ será paralela al eje X durante la primer proyección π_0, recorriendo para las subsecuentes proyecciones al disco que contiene a P en sentido antihorario.
Si bien, dado un conjunto de puntos P le podemos asignar una secuencia circular,



$$\begin{aligned}
\pi_0 &= (1,\ 2,\ 3,\ 4)\\
\pi_1 &= (1,\ 3,\ 2,\ 4)\\
\pi_2 &= (1,\ 3,\ 4,\ 2)\\
\pi_3 &= (1,\ 4,\ 3,\ 2)\\
\pi_4 &= (4,\ 1,\ 3,\ 2)\\
\pi_5 &= (4,\ 3,\ 1,\ 2)\\
\pi_6 &= (4,\ 3,\ 2,\ 1)
\end{aligned}$$

Cuadro 1.1: Proyeción un conjunto de 4 puntos, y la secuencia circular correspondiente.

no toda secuencia circular viene de un conjunto de puntos. Esto fue demostrado por Pollack y Goodman [10], quienes dieron una secuencia circular de cinco elementos, y probaron que no existe conjunto de puntos que la genere.

Definición 1.12. Una transposición que ocurre entre dos elementos en los lugares i y $i+1$ con $1 \leq i < n$, será llamada una *$i-$transposición*. Para $i \leq n/2$, una *$i-$transposición crítica* es una $i-$transposición y una una $(n-i)-$transposición, y que una *$(\leq k)-$transposición crítica* es una $i-$transposición que es critica para algún $i \leq k$.

Observemos que existe una relación entre las $i-$transposiciones y las $j-$aristas, pues la recta definida los dos elementos que se intercambian en una $i-$transposición, es una $(i-1)-$arista en P. Así, contando el número de $i-$transposiciones en Π_P, y utilizando el teorema 1.7, podemos obtener el número de cruce rectilíneo. En esta fórmula es fácil notar que las $i-$transposiciones críticas contribuyen en me-

nor cantidad al número de cruce cuando i es más cercana a $\lfloor (n-2)/2 \rfloor$. Por lo que podriamos pensar que entre más $\lfloor (n-2)/2 \rfloor -$aristas o rectas medianas tengamos, el número de cruce será menor. Si bien aún no se ha demostrado que exista una relación entre el número de cruce rectilíneo y las rectas medianas, es posible que así sea.

1.6. Construcciones Aumentantes.

En la búsqueda de mejorar las cotas del número de cruce rectilíneo y la cota superior del problema de Sylvester, se han utilizado varias técnicas, una de ellas es la *construcción de dibujos aumentantes*, que consiste, esencialmente, en generar conjuntos de puntos a partir de conjuntos conocidos de menor cardinalidad mediante la duplicación de puntos o remplazo de los mismos por nuevos conjuntos de puntos llamados cúmulos.
Un método de duplicación, fue utilizado por Abrego y Fernandez-Merchant en [2] logrando el siguiente resultado.

Teorema 1.13. *Si P es un conjunto de N puntos en posición general, con N par, y P tiene un emparejamiento por rectas medianas; entonces*

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

Un par de años después Ábrego et al [1] siguiendo esta línea de ideas, y perfeccionando construcciones previas, desarrollaron una nueva construcción en la que se remplazan algunos puntos por cúmulos, los cuales son copias afines de conjuntos de puntos óptimos. Ellos obtuvieron la desigualdad anterior para conjuntos P de N puntos en posición general con N impar, prescindiendo de la existencia de un emparejamiento por rectas medianas. Además de obtener los siguientes resultados

Teorema 1.14. *Sea P un conjunto $3-$descomponible de n puntos en el plano en posición general. Entonces*

$$\overline{cr}(P) \geq \frac{2}{27}(15 - \pi^2)\binom{n}{4} + \Theta(n^3) > 0{,}380029\binom{n}{4} + \Theta(n^3).$$

Además de lograr la mejor cota superior para el problema de Sylvester, obtenida hasta ese momento, utilizando un dibujo de K_{90} con 951526 cruces.
Obteniendo:

$$q_* \leq \frac{83247328}{218791125} < 0{,}380488.$$

CAPÍTULO 2

Aumentando Puntos

2.1. Aumentando Secuencias Circulares

Como mencionamos anteriormente, el uso de construcciones aumentantes en busca de mejores conjuntos de puntos es una técnica recurrente y que ha logrado buenos resultados. Si bien en [1] se da una construcción en la que se remplazan puntos de un conjunto inicial por cúmulos, la forma en la que se eligen estos puntos y cuál será la cardinalidad de los cúmulos que los remplazarán, sigue siendo un problema complicado.

La aplicación de algoritmos computacionales es sin duda una herramienta que reduce considerablemente los esfuerzos al realizar estas construcciones. Por ello en esta sección estudiaremos la manera de aumentar conjuntos de puntos a través de sus secuencias circulares, buscando aquellas con pocos cruces, para posteriormente crear conjuntos de puntos cuya respectiva secuencia circular sea lo más similar posible. Daremos primero una manera de generar secuencias circulares aumentadas a partir de conjuntos de puntos.

La entrada y la salida

La entrada

Un conjunto $P = (p_1, p_2, ..., p_n)$ de puntos en el plano en posición general. Donde, preferentemente, P sea la base de un dibujo geométrico K_n con pocos cruces.
Una función $w : P \to \mathbb{N} \setminus \{0\}$ que llamaremos función de peso. Esta función nos indica cuantos puntos habrá en el conjunto final P' por cada punto existente en P.

La salida

Una secuencia circular $\Pi' = (\pi'_0, \pi'_1, ..., \pi'_{\binom{m}{2}})$ de permutaciones del conjunto de cardinalidad $m = \sum_{i=0}^{n} w(p_i)$.

La construcción

Dada la secuencia circular Π inducida por el conjunto P, generáremos otra secuencia Π' con un mayor número de elementos y que guarde cierta estructura con la secuencia original.
Recordemos que π_{i+1} es generada por π_i después de una transposición entre dos elementos adyacentes. Sea $C = ((s_1, q_1), (s_2, q_2), ..., (s_n, q_n))$ donde (s_i, q_i) son los elementos que se intercambian en π_{i-1} para generar π_i, la cual llamaremos lista de transposiciones de Π. Claramente C, y π_0 contienen la información necesaria para reconstruir Π. Para construir a Π', debemos obtener una lista de transposiciones C' y una secuencia inicial π'_0.
Por cada elemento $p_i \in P$ habrá $w(p_i)$ elementos en P'. A los conjuntos de puntos que provienen de un mismo elemento los llamaremos bloques y los denotaremos por $B(p_i) = (p_{i_1}, p_{i_2}, ..., p_{i_{w(p_i)}})$, donde $p_{i_1} = p_i$, y $p_{i_2}, p_{i_3}, ..., p_{i_{w(p_i)}}$ son elementos nuevos que serán puntos cercanos a p_{i_1} en la geometrización. Para facilitar la lectura, denotaremos estos bloques simplemente como B_i. Definimos también la secuencia inicial

$$\begin{aligned} \pi'_0 &= (B_1, B_2, ..., B_n) \\ &= (p_{1_1}, ..., p_{1_{w(p_1)}}, p_{2_1}, ..., p_{2_{w(p_2)}}, ..., p_{n_1}, ..., p_{n_{w(p_n)}}). \end{aligned}$$

Y utilizando C, generamos una primer lista de transposiciones $\overline{C}$, donde por cada cambio (q_i, s_i) en C, existirá una secuencia de cambios en $\overline{C}$, que transponen a los bloques $Q_i = B(q_i)$ y $S_i = B(s_i)$. La secuencia de cambios entre los bloques Q_i y S_i será la siguiente.

$$(s_{i_{w(s_i)}}, q_{i_1}), (s_{i_{w(s_i)-1}}, q_{i_1}), ..., (s_{i_1}, q_{i_1}), (s_{i_{w(s_i)}}, q_{i_2}), (s_{i_{w(s_i)-1}}, q_{i_2}), ...,$$

$$(s_{i_1}, q_{i_2}), ..., (s_{i_{w(s_i)}}, q_{i_{w(q_i)}}), (s_{i_{w(s_i)-1}}, q_{i_{w(q_i)}}), ..., (s_i, q_{i_{w(q_i)}}).$$

Esto no es más que recorrer los elementos del segundo bloque de manera ordenada hasta la posición que les corresponderá al final, por ejemplo, al transponer los bloques $S = (s_1, s_2, s_3)$ y $Q = (q_1, q_2, q_3)$ tendremos.

$$\begin{array}{ll} s_1, s_2, s_3, q_1, q_2, q_3 & q_1, s_1, q_2, s_2, s_3, q_3 \\ s_1, s_2, q_1, s_3, q_2, q_3 & q_1, q_2, s_1, s_2, s_3, q_3 \\ s_1, q_1, s_2, s_3, q_2, q_3 & q_1, q_2, s_1, s_2, q_3, s_3 \\ q_1, s_1, s_2, s_3, q_2, q_3 & q_1, q_2, s_1, q_3, s_2, s_3 \\ q_1, s_1, s_2, q_2, s_3, q_3 & q_1, q_2, q_3, s_1, s_2, s_3 \end{array}$$

Abusando un poco de la notación, denotaremos a estas secuencias de transposiciones por (S_i, Q_i), y las llamaremos transposiciones de bloques, luego, siguiendo el orden de las transposiciones de elementos en C, obtenemos una nueva lista de transposiciones de bloques

$$\begin{aligned}\overline{C} &= ((S_1, Q_1), (S_2, Q_2), ..., (S_n, Q_n)),\\ &= ((s_{1_{w(s_1)}}, q_{1_1}), ..., (s_{1_1}, q_{1_1}), ..., (s_{1_{w(s_1)}}, q_{1_{w(q_1)}})..., (s_{1_1}, q_{1_{w(q_1)}}), ...,\\ &\quad ((s_{n_{w(s_n)}}, q_{n_1}), ..., (s_{n_1}, q_{n_1}), ..., (s_{n_{w(s_n)}}, q_{n_{w(q_n)}})..., (s_{n_1}, q_{n_{w(q_n)}}).\end{aligned}$$

Realizando los cambios de $\overline{C}$ a la secuencia inicial π'_0, se genera la secuencia de permutaciones

$$\overline{\Pi} = (\pi'_0, \pi'_1, ..., \pi'_{|\overline{C}|}),$$

donde

$$\begin{aligned}\pi'_{|\overline{C}|} &= (P_n, P_{n-1}, ..., P_0)\\ &= (p_{n_1}, ..., p_{n_{w(p_n)}}, p_{n-1_1}, ..., p_{n-1_{w(p_{n-1})}}, ..., p_{1_1}, ..., p_{1_{w(p_1)}}).\end{aligned}$$

Esta secuencia es ya cercana a la secuencia circular buscada, excepto por el hecho de que los bloques están invertidos. Para corregir esto, es necesario modificar $\overline{C}$ agregando los cambios necesarios. Como ya hemos mencionado, las $i-$transposiciones críticas que aportan menos al número de cruce, son aquellas en las que i es más cercano a $\lfloor (m-2)/2 \rfloor$. Esto implica que el mejor momento para invertir los bloques, es cuando estos están lo más centrados posible. Como cada transposición entre bloques $(Q, S) \in \overline{C}$ define un grupo de permutaciones en $\overline{\Pi}$. Un buen momento para invertir un bloque, será justo entre dos trasposiciones (Q_j, S_j) y (Q_{j+1}, S_{j+1}) de $\overline{C}$ para algún subíndice j. Esta inversión, deberá a su vez contribuir lo menos posible con el número de cruce. Por lo tanto para invertir un bloque de r elementos utilizaremos la lista de cambios definida por la secuencia circular del mejor dibujo conocido de K_r. Por ejemplo, para un bloque $Q = q_1, q_2, q_3, q_4$ la inversión será dada por la secuencia de cambios $((q_4, q_3), (q_4, q_2), (q_4, q_1), (q_2, q_1), (q_3, q_2))$, la cual de forma más visual es

$$\begin{gathered}q_1, q_2, q_3, q_4\\ q_1, q_2, q_4, q_3\\ q_1, q_4, q_2, q_3\\ q_4, q_1, q_2, q_3\\ q_4, q_2, q_1, q_3\\ q_4, q_2, q_3, q_1\\ q_4, q_3, q_2, q_1\end{gathered}$$

Denotemos a las secuencias de transposiciones que invierten un bloque P_i como $(P_i)^{-1}$, y al bloque invertido como P_i^{-1}. Claramente no basta con insertar estas secuencias entre dos transposiciones de bloques, pues al realizar dicho proceso se deben modificar también las subsecuentes transposiciones de bloques en $\overline{C}$.

Atendiendo esto, si el bloque B_i se invierte entre las transposiciones de bloques (Q_j, S_j) y (Q_{j+1}, S_{j+1}), insertamos la secuencia $(B_i)^{-1}$ entre ellas y cambiamos a B_i por B_i^{-1} en las transposiciones de bloques siguientes.
Al término de esta construcción, habremos agregado todas las secuencias de transposiciones que determinan las inversiones de los bloques a la secuencia de cambios $\overline{C}$ obteniendo una nueva lista de transposiciones C', de tal manera que aplicando estas transposiciones a la permutación inicial π'_0, obtenemos la secuencia circular

$$\Pi' = (\pi'_0, \pi'_1, ..., \pi'_{\binom{m}{2}}),$$

donde

$$\begin{aligned} \pi'_{\binom{m}{2}} &= (B_n^{-1}, B_{n-1}^{-1}, ..., B_1^{-1}), \\ &= (p_{n_{w(p_n)}}, ..., p_{n_1}, p_{n-1_{w(p_{n-1})}}, ..., p_{n-1_1}, ..., p_{1_{w(p_1)}}, ..., p_{1_1}). \end{aligned}$$

Claramente el número de cruce rectilíneo de Π', puede optimizarse mediante algunas modificaciones en la construcción, por ejemplo modificando la forma en la que se llevan a cabo algunas transposiciones de bloques cuando estas se realizan cerca de la posición $\binom{m}{2}$, de forma que durante el proceso se centre completamente uno de los bloques involucrados y se realice en ese momento, la inversión del mismo. Otra manera de optimizar este proceso, es invertir los bloques de tal forma que la mayor cantidad de transposiciones entre sus elementos se realicen lo más centradas posible.
Las secuencias circulares construidas de la manera descrita anteriormente, preservan mucho de la estructura de los conjuntos de los que provienen, y al tomar conjuntos iniciales con pocos cruces, uno esperaría que las secuencias resultantes tuvieran también un número de cruce rectilíneo pequeño. Esto no siempre sucede, pues en ocasiones, pequeñas perturbaciones en el comportamiento de elementos de la secuencia aportan mucho al número de cruce final. Entonces ¿cuál es la ventaja de construir las secuencias circulares aumentadas? La ventaja radica en que su construcción resulta ser, en términos computacionales, muy económica. Así podemos aumentar una secuencia millones de veces en busca de nuevas secuencias que nos ayuden en la construcción de conjuntos de puntos con pocos cruces.

La geometrización

Aún después de la construcción de una secuencia circular aumentada Π, debemos construir también el conjunto de puntos P que la genera, a este proceso lo llamaremos *geometrización de* Π. Como se mencionó anteriormente, no hay una relación suprayectiva entre los conjuntos de puntos en el plano y las secuencias circulares. Para evitar caer en ese dilema, el método que utilizamos para geometrizar las secuencias circulares halladas, fue un método de aproximación por exhausción, método en el que si bien, no en todos los casos se obtienen los

conjuntos de puntos que generan las secuencias circulares exactas, se logra que los conjuntos encontrados generen secuencias muy parecidas (que difieren en muy pocas transposiciones en comparación con las secuencias ideales) y que en la mayoría de los casos tienen el mismo número de cruce rectilíneo. Este proceso se describe de manera más precisa en la siguiente sección.

2.2. Geometrizando Conjuntos de Puntos con Elementos Duplicados

Dado un conjunto P de n puntos en posición general en el plano, tal que todos los puntos tienen diferente coordenada en y, construiremos secuencias circulares aumentadas donde $w : P \to \{1, 2\}$, es decir, duplicamos subconjuntos de puntos del conjunto P en busca de secuencias circulares con pocos cruces, para posteriormente geometrizarlas. Si un punto p es duplicado, el nuevo punto p', es llamado *punto gemelo* de p. En esta sección daremos una técnica para geometrizar esta clase de secuencias circulares.
Sea P un conjunto como se describió anteriormente, Π su secuencia circular, y Π' una secuencia aumentada de $m > n$ elementos, generada por Π mediante la duplicación de algunos elementos. Notemos que la forma en la que se genera una secuencia circular induce un orden parcial en las transposiciones, diremos que la transposición $(u, v) < (w, z)$ si la pendiente entre los puntos u y v en menor a la pendiente entre w y z. Notemos también que si la transposición (u, v) genera a la permutación π_i y la transposición (w, z) genera a π_j donde $i < j$, entonces se cumple que $(u, v) < (w, z)$.
Al geometrizar Π', el conjunto resultante P' contendrá al conjunto P y los puntos nuevos serán justamente los gemelos de los puntos que fueron duplicados. Sean $Q = \{q_1, q_2, ..., q_k\} \subset P$ elementos que se duplicaron, denotaremos al gemelo de q_i por q'_i. Podemos suponer (reetiquetando si es necesario) que las transposiciones entre elementos gemelos suceden en el orden de los subíndices en C'. El proceso para dar coordenadas a los puntos gemelos será el siguiente,
Dada la transposición (q_1, q'_1) consideremos a las transposiciones (u, v) y (w, z) tales que

$$(u, v)_1 = max\,\{(x, y) \in C' : (x, y) < (q'_1, q_1), x, y \in P \cup A\},$$
$$(w, z)_1 = min\,\{(x, y) \in C' : (x, y) > (q'_1, q_1), x, y \in P \cup A\},$$

donde el conjunto A es el conjunto de puntos gemelos a los cuales ya se les asignaron coordenadas, es decir, las transposiciones anterior y posterior a (q'_1, q_1) en C' con elementos cuyas coordenadas son conocidas. Así tenemos que $(u, v) < (q'_1, q_1) < (w, z)$, lo cual nos indica entre que pendientes se encuentra la pendiente que debe formar q_1 con q'_1. Por otro lado considerando la manera en la que se eligió π_0 tenemos que q'_1 debe proyectarse justo delante de q_1 y antes de $q + 1$, es decir, la coordenada en X de q'_1 debe ser mayor que la coordenada en X del punto

q_1 pero menor a la del punto $q+1$. Así q_1' estará contenido en la región triangular determinada por las rectas $\ell_1 = \{(x,y) \in \mathbb{R} : y_{q_1} - y = m(u,v)(x_{q_1} - x)\}$, $\ell_2 = \{(x,y) \in \mathbb{R} : y_{q_1} - y = m(w,z)(x_{q_1} - x)\}$ y $\ell_3 = \{(x,y) \in \mathbb{R} : x = x_{q_1+1}\}$ (ver figura 2.1). Por la manera en la que se construyó la secuencia aumentada Π'

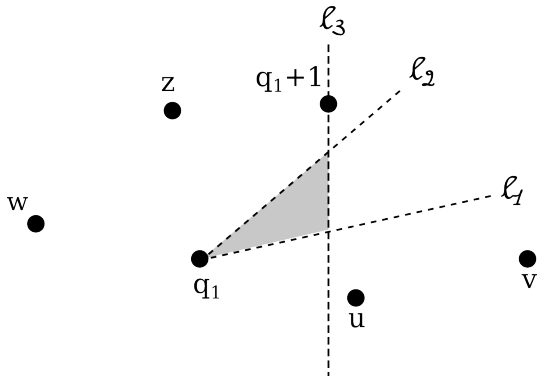


Figura 2.1: Región donde se encuentra k'

la transposición (q', q) ocurrirá lo más centrada posible, así si m es par, la recta determinada por los puntos q y q' es una recta mediana. Mientras que si m es impar la recta determinada por los puntos q y q' dejará $\lfloor (m-2)/2 \rfloor$ elementos de un lado y $\lceil (m-2)/2 \rceil$ del otro lado. También es necesario que q' y q estén lo suficientemente cerca para que q' no genere demasiados cruces con los otros puntos de P. Así repitiendo este proceso con para todos los elementos de Q podemos asignar coordenadas a sus puntos gemelos.

A continuación se muestran algunos de los resultados obtenidos mediante la duplicación y geometrización de puntos. El cuadro 2.1 muestra los números de cruces obtenidos al geometrizar las secuencias circulares generadas por el programa A.2 duplicando subconjuntos del conjunto de 75 puntos reportado por Fabila-Monroy y López en [14]. Mientras que el cuadro 2.2 muestra las cotas logradas por esos conjuntos para el valor asintótico de $\overline{cr}(n)$.

Notemos que el valor asintótico obtenido para K_{150} es un poco mejor (por una muy modesta cantidad) que el que alcanza el K_{75}, el cual es el mejor resultado publicado hasta ahora, sin embargo estamos enterados de la existencia de mejores dibujos aún no publicados.

Cotas generadas duplicando conjununtos K_{75} con $100 < n$							
n	$\overline{cr}(n) \leq$	n	$\overline{cr}(n) \leq$	n	$\overline{cr}(n) \leq$	n	$\overline{cr}(n) \leq$
101	1524077	114	2496456	127	3873123	140	5753206
102	1586549	115	2586808	128	3998431	141	5921639
103	1651075	116	2679548	129	4126904	142	6093952
104	1717423	117	2774663	130	4258605	143	6269987
105	1785614	118	2872550	131	4393112	144	6449453
106	1856144	119	2972971	132	4530832	145	6633338
107	1928526	120	3075565	133	4671821	146	6820727
108	2002870	121	3181432	134	4816303	147	7012052
109	2079687	122	3289678	135	4963571	148	7207471
110	2158463	123	3400602	136	5114861	149	7407147
111	2239417	124	3514569	137	5269160	150	7610243
112	2322834	125	3631157	138	5426814		
113	2408436	126	3750527	139	5588334		

Cuadro 2.1

Cotas para el valor asintótico de $\overline{cr}(n) \leq$							
n	$\overline{cr}(n) \leq$	n	$\overline{cr}(n) \leq$	n	$\overline{cr}(n) \leq$	n	$\overline{cr}(n) \leq$
101	0.38052685	114	0.38052462	127	0.3805104	140	0.38049860
102	0.38051723	115	0.38052428	128	0.38049988	141	0.38049045
103	0.38054041	116	0.38051832	129	0.38050242	142	0.38049473
104	0.38053726	117	0.38050011	130	0.3805181	143	0.38049830
105	0.38050718	118	0.38051464	131	0.38050738	144	0.38048198
106	0.38054006	119	0.38052496	132	0.38050091	145	0.38049812
107	0.38053265	120	0.38048454	133	0.38049884	146	0.38049291
108	0.38050166	121	0.38051646	134	0.38051377	147	0.38048743
109	0.38052971	122	0.38051205	135	0.38048952	148	0.38048689
110	0.38051966	123	0.38050133	136	0.38051264	149	0.38049612
111	0.38050362	124	0.38051731	137	0.3805066	150	0.38047204
112	0.38051944	125	0.38051145	138	0.38049339		
113	0.38051677	126	0.38049665	139	0.38050314		

Cuadro 2.2

CAPÍTULO 3

Resultados y Trabajo a Futuro

3.1. Dos Nuevos Conjuntos

Los resultados más importantes obtenidos a lo largo del presente trabajo son el conjunto de 150 puntos y su cota para el valor asintótico presentado en el cuadro 2.2, así como el hallazgo de dos conjuntos con 96 y 99 puntos con coordenadas enteras, cuyo número de cruce rectilíneo es menor al de los conjuntos con las mismas cardinalidades conocidos. Estos conjuntos se obtuvieron aumentando la secuencia circular del conjunto de 75 puntos dado por Fabila-Monroy. Los mejores conjuntos de puntos reportados en [13] generan las cotas superiores $\overline{cr}(96) = 1238662$ y $\overline{cr}(99) = 1404592$ mientras que los presentados a continuación logran $\overline{cr}(96) = 1238635$ y $\overline{cr}(99) = 1404522$. Estos nuevos conjuntos son,

Conjunto de 96 **puntos con** 1238635 **cruces**

$(-1364755153000000, -2899618565000000)$	$(-495951185000000, 16620108498000000)$
$(-495950185000000, 16620106842232440)$	$(-311149395000000, 16314077753000000)$
$(1044611367000000, 14069644578000000)$	$(1133302705000000, 13923635114000000)$
$(1293365372000000, 2095165431000000)$	$(1474528964000000, 2436685704000000)$
$(1474529964000000, 2436687538927530)$	$(1549775961000000, 2575359287000000)$
$(1679455404000000, 2812631891000000)$	$(1683153691000000, 13003463181000000)$
$(1811935937000000, 12802330604000000)$	$(1889666524000000, 3220648103000000)$
$(1889667524000000, 3220649836120660)$	$(1902291407000000, 12661152660000000)$
$(1902292407000000, 12661150858075020)$	$(1902363904000000, 3245131307000000)$
$(2027617952000000, 3459524378000000)$	$(2067188794000000, 12364750532000000)$
$(2113073755000000, 12281280867000000)$	$(2154725117000000, 3676030999000000)$
$(2195118038000000, 12138376393000000)$	$(2195119038000000, 12138374545105480)$
$(2297590336000000, 3930708704000000)$	$(2297591336000000, 3930710338442910)$

(3168421193000000, 5701152359000000) (3189483730000000, 5743999450000000)
(3189484730000000, 5744000671777890) (3214935430000000, 10605538217000000)
(3253433517000000, 5873175144000000) (3359570710000000, 10389672946000000)
(3400009645000000, 10327277784000000) (3400010645000000, 10327274744752970)
(3570685582000000, 9808713565000000) (3571434484000000, 9806112525000000)
(3648786718000000, 6305728855000000) (3653540692000000, 6310524663000000)
(3722340414000000, 9316231785000000) (3722341414000000, 9316227445929810)
(3798074340000000, 9176659177000000) (3839573186000000, 9100031657000000)
(3955068845000000, 6639763085000000) (3955069845000000, 6639763939874170)
(4012346331000000, 6733970340000000) (4059850707000000, 6811671897000000)
(4106745227000000, 7535480343000000) (4107912992000000, 7542476726000000)
(4112078622000000, 7625130881000000) (4112079622000000, 7625125468799520)
(4113182393000000, 7619250691000000) (4116054424000000, 7605654413000000)
(4338568456000000, 8157953847000000) (4338851382000000, 8157414467000000)
(4338852382000000, 8157412320209190) (4471841261000000, 8674882284000000)
(4473587539000000, 8674070321000000) (4520171724000000, 8637506721000000)
(4520172724000000, 8637505374443400) (4532317105000000, 8633237970000000)
(4538689274000000, 8630906861000000) (4601078091000000, 7662133857000000)
(4601468259000000, 7662169961000000) (4601469259000000, 7662170033977630)
(4897948559000000, 8128714256000000) (4897949559000000, 8128712788822100)
(4899124137000000, 8128629846000000) (5202684700000000, 7706307614000000)
(5207789612000000, 7710691788000000) (5216754785000000, 7718023020000000)
(5277878757000000, 7741749531000000) (5279252153000000, 7742686707000000)
(5279253153000000, 7742685551703490) (5981198967000000, 8072572208000000)
(6032867454000000, 8070589271000000) (6168237700000000, 8065376268000000)
(6168238700000000, 8065375920916000) (6851478487000000, 7943849321000000)
(6888712646000000, 7936512772000000) (7338699912000000, 7861922951000000)
(7338700912000000, 7861922782985890) (7370957968000000, 7863465953000000)
(7493305742000000, 7871610457000000) (8806696260000000, 7963533399000000)
(8806697260000000, 7963533407396890) (8944839519000000, 7965414411000000)
(9000883017000000, 7965096231000000) (9350903224000000, 7955792213000000)
(9630596054000000, 7968154616000000) (10174892708000000, 7993197449000000)
(10174893708000000, 7993197474029010) (10588618608000000, 8002947798000000)
(10634751909000000, 8004278071000000) (11185824774000000, 8018462436000000)
(15041590733000000, 8118237065000000) (15041591733000000, 8118237090716230)

Conjunto de 99 **puntos con** 1404522 **cruces**

(−1364755153000000, −2899618565000000) (−495951185000000, 16620108498000000)
(−311149395000000, 16314077753000000) (−311149385000000, 16314077736396233)
(1044611367000000, 14069644578000000) (1133302705000000, 13923635114000000)
(1293365372000000, 2095165431000000) (1474528964000000, 2436685704000000)
(1474528974000000, 2436685722349275) (1549775961000000, 2575359287000000)
(1549775971000000, 2575359305138083) (1679455404000000, 2812631891000000)

(1683153691000000, 13003463181000000)
(1889666524000000, 3220648103000000)
(1902291407000000, 12661152660000000)
(1902363904000000, 3245131307000000)
(2067188794000000, 12364750532000000)
(2154725117000000, 3676030999000000)
(2195118048000000, 12138376374521063)
(2297590346000000, 3930708720211668)
(3189483730000000, 5743999450000000)
(3214935430000000, 10605538217000000)
(3359570710000000, 10389672946000000)
(3400009655000000, 10327277753598031)
(3571434484000000, 9806112525000000)
(3648786718000000, 6305728855000000)
(3722340414000000, 9316231785000000)
(3798074340000000, 9176659177000000)
(3955068845000000, 6639763085000000)
(4059850707000000, 6811671897000000)
(4106745227000000, 7535480343000000)
(4112078622000000, 7625130881000000)
(4113182403000000, 7619250637778573)
(4338568456000000, 8157953847000000)
(4338851382000000, 8157414467000000)
(4473587539000000, 8674070321000000)
(4532317105000000, 8633237970000000)
(4538689284000000, 8630906846084229)
(4601078101000000, 7662133838112408)
(4897948559000000, 8128714256000000)
(4899124147000000, 8128629832069956)
(5207789612000000, 7710691788000000)
(5277878757000000, 7741749531000000)
(5279252163000000, 7742686695226416)
(6032867454000000, 8070589271000000)
(6168237710000000, 8065376264529160)
(6888712646000000, 7936512772000000)
(7338699922000000, 7861922949266660)
(7493305742000000, 7871610457000000)
(8806696260000000, 7963533399000000)
(8944839519000000, 7965414411000000)
(9350903224000000, 7955792213000000)
(10174892708000000, 7993197449000000)
(10634751909000000, 8004278071000000)
(11185824774000000, 8018462436000000)
(15041590733000000, 8118237065000000)
(1811935937000000, 12802330604000000)
(1889666534000000, 3220648120248215)
(1902291417000000, 12661152641980751)
(2027617952000000, 3459524378000000)
(2113073755000000, 12281280867000000)
(2195118038000000, 12138376393000000)
(2297590336000000, 3930708704000000)
(3168421193000000, 5701152359000000)
(3189483740000000, 5743999462217778)
(3253433517000000, 5873175144000000)
(3400009645000000, 10327277784000000)
(3570685582000000, 9808713565000000)
(3571434494000000, 9806112484663451)
(3653540692000000, 6310524663000000)
(3722340424000000, 9316231741581402)
(3839573186000000, 9100031657000000)
(4012346331000000, 6733970340000000)
(4059850717000000, 6811671904635472)
(4107912992000000, 7542476726000000)
(4113182393000000, 7619250691000000)
(4116054424000000, 7605654413000000)
(4338568466000000, 8157953825502903)
(4471841261000000, 8674882284000000)
(4520171724000000, 8637506721000000)
(4538689274000000, 8630906861000000)
(4601078091000000, 7662133857000000)
(4601468259000000, 7662169961000000)
(4899124137000000, 8128629846000000)
(5202684700000000, 7706307614000000)
(5216754785000000, 7718023020000000)
(5279252153000000, 7742686707000000)
(5981198967000000, 8072572208000000)
(6168237700000000, 8065376268000000)
(6851478487000000, 7943849321000000)
(7338699912000000, 7861922951000000)
(7370957968000000, 7863465953000000)
(7493305752000000, 7871610455887986)
(8806696270000000, 7963533398614029)
(9000883017000000, 7965096231000000)
(9630596054000000, 7968154616000000)
(10588618608000000, 8002947798000000)
(10634751919000000, 8004278071230426)
(11185824784000000, 8018462436250290)

APÉNDICE A

Programas

A.1. Calculando $\overline{cr}(n)$.

Este programa recibe como entrada un conjunto $P = [p_1, p_2, ..., p_n]$ de puntos con coordenadas enteras, y genera a partir de éste la sucesión circular correspondiente. Decidimos utilizar el lenguaje de programación Python versión 2.7.6, por la fácilidad de uso, y la legibilidad de su sintaxis. El código es el siguiente.

```
def Compare_Slope(P1,P2,P3,P4):
    if (P2[1]-P1[1])*(P4[0]-P3[0])<(P4[1]-P3[1])*(P2[0]-P1[0]):
        return -1
    else:
        return 1
def Changes_sort(list):
    for i in range(1,len(list)):
        currentv = list[i]
        pos = i
        while pos>0 and Compare_Slope(list[pos-1][0],
        list[pos-1][1],currentv[0],currentv[1])==1:
            list[pos]=list[pos-1]
            pos = pos-1
            list[pos]=currentv
    return list
def Changes_list(list):
    Couples=[]
    for i in range(len(list)-1):
        for j in range(i+1,len(list)):
            Couples.append([list[i],list[j]])
    aux=Couples[0]
    Changes_sort(Couples)
    return Couples
```

```
def Switch(x,A):
    list=A+[[0,0]]
    k=0
    for i in range(len(list)-2):
        if list[i]==x:
            key=list[i]
            list[i]=list[i+1]
            list[i+1]=key
            break
    list.pop(len(list)-1)
    return list
def Changes(list):
    A=list
    Tem=[0]*(len(A)-1)
    for i in range(len(A)-1):
        for j in range(len(A[i])):
            if A[i][j]!=A[i+1][j]:
                Tem[i]=[str(A[i+1][j]) , str(A[i+1][j+1])]
                break
    return Tem
def CRUCE(A):
    N=[0]*(len(A[0])/2)
    for i in range (len(A)-1):
        for j in range(len(A[0])):
            if A[i][j]!=A[i+1][j]:
                if j<len(A[0])/2:
                    N[j]=N[j]+1
                if j>=len(A[0])/2:
                    N[len(A[0])-j-2]=N[len(A[0])-j-2]+1
                break
    Cr=0
    for i in range (len(N)):
        Cr=Cr+float(N[i]*(float(float(float(len(A[0])-2)/2)-i)**2))
    Cr=Cr-(float(((len(A[0]))*(len(A[0])-1)*(len(A[0])-2))/6))*(
    float(3)/4)
    return Cr
def Compare(A,B):
    Temp=[0]*len(A)
    for i in range(len(A)):
        for j in range(len(A)):
            if A[j][0]==B[i]:
                Temp[i]=A[j][1]
    return Temp
def main():
# El input sera el conjunto de puntos P.
    P=[p1,p2,...,pn]
    P1=[]
    for i in range(len(P)):
        P1.append([P[i],str(i+1)])
    P.sort()
    ChangesList=Changes_list(P)
    PF=[P]+[0]*(len(ChangesList))
    for i in range(len(PF)-1):
```

```
            PF[i+1]=Switch(ChangesList[i][0],PF[i])
77      Temp=P1
        CSC=[0]*len(PF)
79      for i in range(len(CSC)):
            CSC[i]=Compare(Temp,PF[i])
81      Tem=Changes(CSC)
        Cr=CRUCE(CSC)
83  # Output
        print 'El numero de cruce rectilineo es:',Cr
85  main()
```

A.2. Generando Secuencias Circulares Aumentadas.

El siguiente código se utilizó para aumentar secuencias circulares cuya cardinalidad es múltiplo de tres, cambiando algunos de sus puntos por bloques de hasta 5 puntos (fácilmente se puede editar para bloques de cardinalidad mayor). En este caso elegimos utilizar el lenguaje *C* por su rapidez, ya que era necesario correr millones de simulaciones.
El programa recibe como entrada la cardinalidad del conjunto base, la lista de cambios que se realiza en la secuencia circular inducida por el conjunto base y la secuencia inicial de la misma, y devuelve en pantalla, una lista con datos de las mejores secuencias circulares aumentadas obtenidas, el número de cruce rectilíneo que se obtuvo en cada una, y el vector de pesos que las genera, es decir, el vector cuya $i-$ésima entrada indica cuantos puntos tiene el bloque que remplazará al $i-$ésimo elemento de la secuencia inicial, en la secuencia circular aumentada. Además genera un archivo *.txt* con esta información.

```
#include <stdio.h>
#include <time.h>
#define TAM 150
//Pasa un natural a base k y lo regresa en un vector de tamano len(array).
void BaseK(int k, int array[], int numero,int len_array){
        int  residuo,i;
        for (i=0;i<len_array;i++){
                array[i]=0;
        }
        i=0;
        while(numero>=1){
                residuo=numero%k;
                array[i]=residuo;
                numero=numero/k;
                i++;
        }
}
//Remplaza en un array los 0-4 por a-d.
void Remplazo(int a, int b, int c, int d, int e, int array[], int len_array){
        int i;
        for(i=0;i<len_array;i++){
                if(array[i]==0){
                        array[i]=a;
                }
                else if(array[i]==1){
                        array[i]=b;
                }
                else if(array[i]==2){
                        array[i]=c;
```

```
                }
31              else if(array[i]==3){
                        array[i]=d;
33              }
                else if(array[i]==4){
35                      array[i]=e;
                }
37      }
}
39 //Genera un array de tamano len_array, con digitos 'x' y 'y'.
void Nuevo_Pesos(int array[], int len_array, int a, int b, int c,
    int d, int e, int base_k, int numero){
41      int nuevo_array[len_array/3],i;
        BaseK(base_k, nuevo_array, numero,len_array/3);
43      Remplazo(a,b,c,d,e,nuevo_array,len_array/3);
        for (i=0;i<len_array/3;i++){
45              array[i]=nuevo_array[i];
                array[len_array/3+i]=nuevo_array[i];
47              array[2*len_array/3+i]=nuevo_array[i];
        }
49 }
//Divide dos enteros.
51 float division(int a,int b) {
    if(b != 0) {return (float)a/b;}
53      printf("La division entre cero no esta permitida. ");
        return -1;
55 }
//Cambia el elemento k por su sucesor en AInicial regresando estos
    cambios en el array ACambiado.
57 void Mueve(int ACambiado[], int AInicial[], int k, int len){
        int aux,i;
59      for(i=0;i<len;i++){
                ACambiado[i]=AInicial[i];
61      }
        for(i=0;i<len;i++){
63              if(ACambiado[i]==k){
                        aux=ACambiado[i];
65                      ACambiado[i]=ACambiado[i+1];
                        ACambiado[i+1]=aux;
67                      break;
                }
69      }
}
71 //Genera el vector de cambio de las Ns al intercambiarse dos puntos
    inflados por a y b.
int minimo(int a, int b){
73    if (a < b)
        return a;
75    return b;
}
77 void Vector(int a,int b,int Vec[]){
        int min,i;
79      for(i=0;i<TAM;i++){
```

```
                Vec[i]=0;
        }
    min = minimo(a, b);
        for (i=0;i<min;i++){
                Vec[i]=i+1;
                Vec[a+b-2-i]=i+1;
        }
        for(i=min;i<a+b-min;i++){
                Vec[i]=min;
        }
}
//Obtiene el peso de una variable x, dada una lista de pesos y la
    secuencia inicial.
int Peso(int x,int Inicial[],int Pesos[], int len){
        int k=0,i=0;
        while(i<len && k==0){
                if(Inicial[i]==x){
                        k=i;
                }
                i++;
        }
        return Pesos[k];
}
//Saca el indice desde el que comenzaremos a sumar el vector de
    cambio en las Ns.
int Indice(int x, int lista[], int Inicial[], int Pesos[], int
    len_lis, int Suma_Pesos){
        int ind=0,i;
        for (i=0;i<len_lis;i++){
                if(lista[i]==x){
                        break;
                }
        ind=ind+Peso(lista[i],Inicial,Pesos,len_lis);
        }
        return ind;
}
//Saca el cr(n).
int Cruce(int N[],int len_N){
        int i,Ns[len_N/2];
        double aux, K, Cr;
        for(i=0;i<len_N/2;i++){
                Ns[i]=0;
        }
        for(i=0;i<len_N/2;i++){
                Ns[i]=N[i]+N[len_N-2-i];
        }
        aux=division(len_N-2,2);
        K=0;
        for(i=0;i<len_N/2;i++){
                K=K+Ns[i]*((aux-i)*(aux-i));
        }
        Cr=K-(division((len_N)*(len_N-1)*(len_N-2),6)*3)/4;
        return (int)Cr;
```

```
}
int Inflar(int *Cambios, int *Inicial, int *Pesos, int len_fila){
    int Cambiosi1, Cambiosi0;
        int Sec_Cir[(len_fila*(len_fila-1)/2)+2][len_fila],i,j;
        int Cambios_Mismo_Bloque
    [5][4]={{0,0,0,0},{1,0,0,0},{1,2,0,0},{1,3,2,0},{1,7,2,0}};
        for(i=0;i<len_fila;i++){
                Sec_Cir[0][i]=Inicial[i];
        }
        for(i=0;i<1+(len_fila*(len_fila-1)/2);i++){
        Cambiosi1 = *(Cambios + 2*i + 1);
                Mueve(Sec_Cir[i+1],Sec_Cir[i],Cambiosi1,len_fila);
        }
        int Vec[TAM],Suma_Pesos=0,k, indice;
    for (i=0;i<len_fila;i++){
                Suma_Pesos=Suma_Pesos+Pesos[i];
        }
    int N[Suma_Pesos];
    for (i=0;i<Suma_Pesos-1;i++){
                N[i]=0;
        }
        for (i=0;i<(len_fila*(len_fila-1)/2);i++){
        Cambiosi0 = *(Cambios + 2*i + 0);
        Cambiosi1 = *(Cambios + 2*i + 1);
                Vector(Peso(Cambiosi0,Inicial,Pesos,len_fila),Peso(
    Cambiosi1,Inicial,Pesos,len_fila),Vec);
                indice = Indice(Cambiosi1,Sec_Cir[i],Inicial, Pesos,
    len_fila,Suma_Pesos);
                for(j=indice;j<TAM;j++){
                        N[j]=N[j]+Vec[j-indice];
                }
        }
        //Saca los cambios entre elementos del mismo bloque.
        int Vec2[5],l,aux;
        for (i=0;i<5;i++){
                Vec2[i]=0;
        }
        for(i=0;i<len_fila;i++){
                for (j=0;j<3;j++){
                        Vec2[j]= Cambios_Mismo_Bloque[Peso(Inicial[i
    ],Inicial,Pesos,len_fila)-1][j];
                }
                l=0;
                aux=0;
                while(Vec2[l]!=0){
                        aux=l+1;
                        l++;
                }
                for (k=0;k<3;k++){
                        N[Suma_Pesos/2-(aux/2)+k-1]=N[Suma_Pesos/2-(
    aux/2)+k-1]+Vec2[k];
                }
        }
```

```
        for (i=0;i<Suma_Pesos-1;i++){
179     }
        int aux1=0;
181     for(i=0;i<TAM;i++){
                aux1=aux1+N[i];
183     }
    return Cruce(N,Suma_Pesos);
185 }
int main(){
187     clock_t start = clock();
        printf("Comenzamos :)");
189     int puntos=54,i,j,aux, aux2;
        int Cambios[puntos*(puntos-1)/2][2]={{}, {}, {}...{}} //Aqui
    van los cambios
191     int Inicial[puntos]={1,...,n} //Aqui va la secuencia inicial
        int len_fila=sizeof(Inicial)/sizeof(int);
193     int record
    [30]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

    int numeros
    [30]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

195

        for (j=0;j<200000000;j++){
197         int array[puntos];
            int Narray[puntos];
199         int suma=0;
            Nuevo_Pesos(array, puntos, 1,2,3,4,5, 5, j);
201         for (i=0;i<puntos;i++){
                    Narray[i]=array[puntos-1-i];
203         }
            for (i=0;i<puntos/3;i++){
205             suma=suma+Narray[i];
            }
207         if((suma>=21)&&(suma<=23)){
                    aux=Inflar(&Cambios[0][0],Inicial, Narray,
    len_fila);
209                 aux2=suma-17;
                    if ((record[aux2]>aux) || (record[aux2]==0)){
211                         record[aux2]=aux;
                            numeros[aux2]=j;
213                 }
            }
215     }
        for (i=0;i<30;i++){
217             int array[puntos],l;
                printf("\nEl record de num de cruce para %i es de:
    %i\n\nEl vector de pesos es: ", puntos+3*i, record[i]);
219             if(record[i]!=0){
                    Nuevo_Pesos(array, puntos, 1,2,3,4,5,3,i);
221                 for (l=0;l<puntos;l++){
                        printf("%i, ",array[l]);
223                 }
```

```
                }
225
        }
227     FILE* fichero;
    fichero = fopen("Mejores1_2.txt", "wt");
229 for(i=0;i<30;i++){
                fprintf(fichero,"\nPara %i usamos el numero %i, y
    el cr es es: %i",puntos+3*i,numeros[i], record[i]);
231 }
    fprintf(fichero,"Tiempo transcurrido: %f", ((double)clock() -
    start) / CLOCKS_PER_SEC);
233 fclose(fichero);
        printf("\n\nTiempo transcurrido: %f", ((double)clock() -
    start) / CLOCKS_PER_SEC);
235     return 0;
}
```

A.3. Geometrizando Secuencias Circulares Aumentadas.

Por ultimo presentamos el código que se utilizó para la geometrización y obtención del nuevo conjunto de puntos. Este programa utiliza las ideas de la sección 2.2 y los resultados obtenidos por el algoritmo de la sección anterior. El programa recibe como entrada el conjunto de puntos en coordenadas enteras que se tomaron originalmente (P), la secuencia con los elementos a duplicar ($Inicial$) con los cambios que se generan en la misma (obtenidos del algoritmo de la sección anterior), y una lista con los cambios de cada elemento duplicado con su gemelo ($Inflados$), siguiendo el orden en el que aparecen en Lista_de_cambios.

```
P=[[-1364755153, -2899618565], [-495951185, 16620108498],...]
Inicial=['1', '2', '2a', '3', '4', '5', '5a', ...]
Lista_Cambios=['14', '13'],['18', '17'],['12', '11'],['24', '23a'],
    ...]
Inflados=[['38', '38a'], ['32', '32a'], ['28', '28a'],...]
for i in range(len(P)):
        P[i]=[P[i][0]*10**11,P[i][1]*10**11]
Indices=[]
for i in range(len(P)):
        Indices.append(str(i+1))
from decimal import *
def gcd(a, b):
    while b:
        a, b = b, a%b
    return a

def sumaf(f1,f2):
        r=[f1[0]*f2[1]+f1[1]*f2[0],f1[1]*f2[1]]
        Mcd=gcd(r[0],r[1])
        return [r[0]/Mcd,r[1]/Mcd]

def divf(f1,f2):
        return [f1[0]*f2[1],f1[1]*f2[0]]

def Pendiente(P1, P2):
    m=float((Decimal.from_float(P2[1]-P1[1]))/(Decimal.from_float(P2
    [0]-P1[0])))
    return [int(m*10**11),10**11]

def Pend_Laterales2(Lista_Cambios,P,Indices,A):
    i=Lista_Cambios.index(A)
    l=1;r=1
    auxl=0
    auxr=0
    while auxl!=1:
        a0=Lista_Cambios[i-l][0]in Indices
        a1=Lista_Cambios[i-l][1]in Indices
        if a0==True and a1==True:
```

```
            auxl=1
        else:
            l=l+1
    while auxr!=1:
        b0=Lista_Cambios[i+r][0] in Indices
        b1=Lista_Cambios[i+r][1] in Indices
        if b0==True and b1==True:
            auxr=1
        else:
            r=r+1
    x=[P[Indices.index(Lista_Cambios[i-l][0])],P[Indices.index(
    Lista_Cambios[i-l][1])]]
    y=[P[Indices.index(Lista_Cambios[i+r][0])],P[Indices.index(
    Lista_Cambios[i+r][1])]]
    return [Pendiente(x[0],x[1]),Pendiente(y[0],y[1])]

def Ordenada(list):
        aux=0
        i=0
        while (aux!=16 and i<len(list)-1):
                if(list[i+1]-list[i]<0):
                        aux=aux+1
                        i=i+1
                else:
                        i=i+1
        if i==len(list)-1:
                return 1
        else:
                return 0

Lista_Checados=[]
for i in range(len(Inflados)):
        Checado=Inflados[i]
        M=Pend_Laterales2(Lista_Cambios,P,Indices,[Checado[1],
    Checado[0]])
        pendiente=divf(sumaf(M[0],M[1]),[2,1])
        P.append([P[int(Checado[0])-1][0]+pendiente[1],P[int(Checado
    [0])-1][1]+pendiente[0]])
        Indices.append(Checado[1])

print 'P:  ', P
```

Bibliografía

[1] Ábrego, B. M., Cetina, M., Fernández-Merchant, S., Leaños, J., & Salazar, G. (2010). *3-symmetric and 3-decomposable geometric drawings of K_n*. Discrete Applied Mathematics, 158(12), 1240 − 1258.

[2] Ábrego, B. M., & Fernández-Merchant, S. (2007). *Geometric drawings of K_n with few crossings*. Journal of Combinatorial Theory, Series A, 114(2), 373 − 379.

[3] Ábrego, B. M., Balogh, J., Fernández-Merchant, S., Leaños, J., & Salazar, G. (2008). *An extended lower bound on the number of $(\leq k)-$edges to generalized configurations of points and the pseudolinear crossing number of K_n*. Journal of Combinatorial Theory, Series A, 115(7), 1257-1264.

[4] Lovász, L., Vesztergombi, K., Wagner, U., & Welzl, E. (2004). *Convex quadrilaterals and $k-$sets. Contemporary Mathematics*, 342, 139 − 148.

[5] Beineke, L., & Wilson, R. (2010). *The early history of the brick factory problem*. The Mathematical Intelligencer, 32(2), 41 − 48.

[6] Blazek, J., & Koman, M. (1964). *A minimal problem concerning complete plane graphs*. Theory of graphs and its applications, Czech. Acad. of Sci, 113 − 117.

[7] Pfiefer, R. E. (1989). *The historical development of JJ Sylvester's four point problem*. Mathematics Magazine, 62(5), 309 − 317.

[8] Blaschke, W. (1917). *Über affine Geometrie XI: Lösung des "Vierpunktproblems"von Sylvester aus der Theorie der geometrischen Wahrscheinlichkeiten*. Leipziger Berichte, 69, 436 − 453.

[9] Bárány, I. (2001). *A note on Sylvester's four-point problem*. Studia Scientiarum Mathematicarum Hungarica, 38(1 − 4), 73 − 77.

[10] Goodman, J. E., & Pollack, R. (1980). *On the combinatorial classification of nondegenerate configurations in the plane*. Journal of Combinatorial Theory, Series A, 29(2), 220 − 235.

[11] Zarankiewicz, C. (1955). *On a problem of P. Turán concerning graphs*. Fundamenta Mathematicae, 41(1), 137 − 145.

[12] Scheinerman, E. R., & Wilf, H. S. (1994). *The rectilinear crossing number of a complete graph and Sylvester's "four point problem" of geometric probability*. The American mathematical monthly, 101(10), 939 − 943.

[13] Oswin Aichholzer. *On the rectilinear crossing number*. http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/crossing/, 2011.

[14] Fabila-Monroy, R., & López, J. (2014). *Computational search of small point sets with small rectilinear crossing number*. Journal of Graph Algorithms and Applications, 18(3), 393 − 399.