Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Búsqueda de Crossing Families para Gráficas Geométricas

Tesis que presenta

Carlos Antonio Bulnes Domínguez

para obtener el Grado de

Maestro en Ciencias en Computación

Directora de la Tesis:

Dra. Dolores Lara Cuevas

Ciudad de México **Octubre 2017**

II

Agradecimientos

Agradezco a Conacyt por el apoyo económico que me brindó para poder realizar mi maestría.

A mi madre porque gracias a ella he logrado desarrollarme en la vida.

A mi mejor amigo Cristian por escucharme cuando necesitaba apoyo moral o me sentía solo o estresado.

A mis compañeros de la maestría, realmente disfruté estos dos años de mi vida. Agradezco especialmente a Santiago quien siempre me ayudó a entender las terribles matemáticas, sobre todo cuando no tenía ningún conocimiento de teoremas y demostraciones.

A mis maestros de los cuales aprendí un montón de cosas interesantes.

Y finalmente un especial agradecimiento a mi asesora Dolores Lara Cuevas, gracias por ser una excelente asesora y maestra, por todas las enseñanzas y paciencia que me tuvo. Realizar mi tesis con ella fue la mejor decisión que pude haber tomado.

Resumen

Una gráfica es un conjunto de objetos llamados vértices, junto con un conjunto de objetos llamados aristas que unen a los vértices. Una gráfica geométrica es una representación en el plano de una gráfica. Una H-crossing family es un conjunto de gráficas geométricas que cumple dos propiedades: 1) cada elemento del conjunto es una representación de una gráfica H, 2) siempre que tomemos dos elementos del conjunto estos se cruzan. Un thrackle es un conjunto de segmentos que se intersectan dos a dos, es decir, siempre que elegimos dos segmentos éstos se cruzan o comparten un punto. El tamaño de una H-crossing family es el número de elementos en el conjunto. Decimos que un thrackle es máximo si su número de segmentos es igual a su número de puntos. Denotamos como $2K_2$ la gráfica que tiene dos aristas disjuntas en vértices. Denotamos como $K_{1,3}$ a la gráfica bipartita completa con 1 y 3 vértices. En esta tesis realizamos búsquedas computacionales para encontrar la $2K_2$-crossing family y la $K_{1,3}$-crossing family más grande de tal forma que exista al menos una $2K_2$-crossing family y una $K_{1,3}$-crossing familiy para cada conjunto con 8, 9 y 10 puntos. También realizamos búsquedas para encontrar thrackles máximos sobre todos los conjuntos de 8, 9 y 10 puntos. Encontramos que existe al menos una $2K_2$-crossing families y $K_{1,3}$-crossing families de tamaño $\lfloor \frac{n}{4} \rfloor$ para cada conjunto de 8, 9 y 10 puntos. Encontramos también que no todos los conjuntos de 8, 9 y 10 puntos tienen un thrackle máximo y encontramos el número de conjuntos que sí lo tienen. Damos algunas características que cumplen los conjuntos que sí tienen thrackle máximo.

Abstract

A graph is defined by a set V of vertices and a set E of edges defined over V. A geometric graph is a drawing of a graph in the plane such that each vertex is represented by a point, and each edge is represented by a segment between the corresponding two points. A *H-crossing family* is a set of geometric graphs where, each element is a drawing of a graph H in the plane and every two elements cross. A *thrackle* is a set of edges where every two edges intersect. The *size* of an H-crossing family is the number of elements in the set. A thrackle is *maximum* if it has as many edges as vertices. In this thesis we present some computational searches to find the maximum size of a $2K_2$-crossing family, and a $K_{1,3}$-crossing family, in point sets with 8, 9 and 10 points. Every set of 8, 9 and 10 points must have a least one $2K_2$-crossing family or a $K_{1,3}$-crossing family of such a maximum size. We also present computational searches to find maximum thrackles in point sets of 8, 9 and 10 points. We find that there is a least one $2K_2$-crossing family and a $K_{1,3}$-crossing family of size $\lfloor \frac{n}{4} \rfloor$ on every set with 8, 9 and 10 points. We also find that not every set with 8, 9 and 10 points have a maximum thrackle. We show which are the point sets of size 8, 9 and 10 that have a maximum thrackle and we characterized such sets.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El trabajo que presentamos en esta tesis se encuentra dentro de las áreas de la teoría de gráficas y de la geometría combinatoria. Una gráfica es un conjunto de objetos llamados vértices, junto con un conjunto de objetos llamados aristas. Las aristas unen a los vértices de una gráfica, estableciendo relaciones entre ellas. Las gráficas son una herramienta muy útil en las ciencias de la computación puesto que éstas modelan las relaciones que existen entre un conjunto de datos. Por otra parte, en la geometría combinatoria se estudian las propiedades combinatorias de objetos geométricos. Una gráfica geométrica es una representación en el plano de una gráfica. En esta tesis estudiamos algunas propiedades combinatorias de las gráficas geométricas. Específicamente, estudiamos el cruce e intersección de gráficas geométricas.

Una H-crossing family es un conjunto de gráficas geométricas que cumplen dos propiedades: 1) todas las gráficas geométricas del conjunto son un dibujo de una gráfica H, donde H es una gráfica (abstracta), 2) siempre que se tomen dos elementos del conjunto éstos se cruzan. Un thrackle es un conjunto de segmentos que se intersectan dos a dos, es decir, siempre que elegimos dos segmentos éstos se cruzan o comparten un punto. El tamaño de una H-crossing family es el número de elementos en el conjunto. Un thrackle máximo es un thrackle donde el número de elementos en el conjunto es igual su número de puntos. Tanto las H-crossing families como los thrackles están definidos sobre un conjunto de n puntos. Existen infinitos puntos en el plano, por lo tanto, existen infinitos conjuntos de n puntos y por consecuencia el espacio de búsqueda del problema de encontrar H-crossing families y thrackles máximos de n puntos es infinito. Existe en la literatura una clasificación combinatoria para los conjuntos de puntos en el plano. Dicha clasificación se encuentra disponible en una base de datos para conjuntos de 3 a 10 puntos.

En esta tesis realizamos búsquedas computacionales para encontrar la H-crossing families más grande tal que exista al menos una en cada conjunto de 8, 9 y 10 puntos. También realizamos búsquedas para encontrar thrackles máximos sobre todos los conjuntos de 8, 9 y 10 puntos. Determinamos que el tamaño máximo para $2K_2$-crossing families y $K_{1,3}$-crossing families es igual a $\lfloor \frac{n}{4} \rfloor$. Además, encontramos que aproximadamente el 30 % de los conjuntos de 8, 9 y 10 puntos tienen un thrackle máximo, y damos algunas características de éstos conjuntos.

Esta tesis se encuentra organizada de la siguiente forma: en el capítulo 2 presentamos los conceptos de teoría de gráficas y geometría combinatoria necesarios para esta tesis. En el capítulo 3 realizamos una revisión de los trabajos previos sobre crossing families. En el capítulo 4 mostramos los resultados que obtuvimos de hacer búsquedas de $2K_2$-crossing families. En el capítulo 5 mostramos los resultados que obtuvimos al hacer búsquedas de $K_{1,3}$-crossing families. En el capítulo 6 mostramos los resultados que obtuvimos de hacer búsquedas de thrackles máximos. Finalmente, en el capítulo 7 presentamos una discusión acerca de los resultados obtenidos, así como algunas conclusiones.

Capítulo 2

Antecedentes

En este capítulo damos las definiciones necesarias para esta tesis. Primero damos definiciones respecto a gráficas y posteriormente respecto a gráficas geométricas.

2.1. Gráficas

Los conceptos presentados en este sección fueron tomados de [2]. Cualquier otra fuente será indicada en el concepto correspondiente.

Definimos una *gráfica* $G = (V, E)$ como un conjunto finito no vacío V de objetos llamados *vértices* junto con un conjunto finito no necesariamente vacío E de subconjuntos de dos elementos de V llamados aristas. A cada arista $\{u, v\} \in V(G)$ la denotamos como uv. Si $e = uv$, entonces decimos que la arista e *une* u y a v. El número de vértices en una gráfica G es el *orden* de G y lo denotamos como n. El *grado* de un vértice $v \in V(G)$ es el número de aristas que inciden en él, y lo denotamos como $d(v)$. Si $uv \in G$, entonces u y v son adyacentes. Decimos que una gráfica G es *completa* si cada dos vértices $v_i, v_j \in V(G)$ son adyacentes. Denotamos como K_n a la gráfica completa de orden n. Notemos que existen $\binom{n}{2}$ aristas en una gráfica completa. Decimos que una gráfica H es una *subgráfica* de la gráfica G si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$.

Decimos que dos conjuntos U y W forman una *partición* de un conjunto P si $U \cap W = \emptyset$ y $U \cup W = P$. Decimos que una *gráfica* G es bipartita si existe una partición U, W de su conjunto de vértices tal que cada arista de la gráfica une sólo a elementos de U con elementos de W. Una gráfica bipartita es *completa* si tiene todas las aristas posibles. Sean $s = |U|$ y $t = |W|$. Denotamos como $K_{s,t}$ a la gráfica bipartita completa con bipartición

$V = U \cup W$. En la figura 2.1 mostramos a la gráfica bipartita completa con $s = 1$ y $t = 3$.

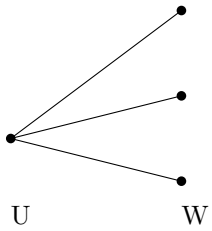

Figura 2.1: Gráfica bipartita $K_{1,3}$

Sea $k \geq 2$, y sea $V = \{v_1, v_2, ..., v_k\}$ un conjunto de k vértices distintos de G. Un *ciclo* $C_k = (v_1, v_2, ..., v_k, v_{k+1} = v_1)$ de G es la secuencia formada por el conjunto V, tal que $v_1 = v_{k+1}$ y cada dos vértices consecutivos en la secuencia son adyacentes en G. En la figura 2.2 mostramos el ciclo $C_5 = (v_1, v_2, v_3, v_5, v_4, v_1)$.

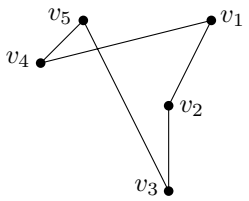

Figura 2.2: Ciclo $C_5 = (v_1, v_2, v_3, v_5, v_4, v_1)$.

El siguiente teorema caracteriza al grado de los vértices de un ciclo.

Teorema 1. *Sea C un ciclo de una gráfica G. Todos los vértices de C son de grado dos.*

Demostración. Podemos distinguir dos casos en la secuencia de vértices que definen a un ciclo:

1. Los vértices v_i, $i \in \{2, ..., k\}$ tal que cada uno de ellos es distinto a cualquier otro.

2. Los vértices v_1, v_k de la secuencia son iguales. Esto es $v_1 = v_{k+1}$.

Consideramos ambos casos por separado para demostrar el teorema.

Por la definición de un ciclo, cada vértice $v_i, i \in \{2, ..., k\}$ aparece exactamente una vez en la secuencia. Notemos también que cada uno de estos

vértices tiene exactamente un vértice que le antecede y un vértice que le procede. Por lo tanto, todos los vértices del caso 1 tiene exactamente dos vecinos.

Para el caso 2, notemos que al vértice v_1 le procede el vértice v_2. De la misma manera al vértice v_{k+1} le antecede el vértice v_k. Por la definición del ciclo $v_1 = v_{k+1}$, por lo tanto los vértices v_1, v_{k+1} de la secuencia tienen exactamente dos vecinos. Con esto demostramos que todos los vértices de un ciclo son de grado dos. □

2.2. Gráficas Geométricas

Sea S un conjunto de n puntos en el plano. Decimos que los puntos en S están en posición general si no existen tres puntos en S sobre una misma recta. Una *gráfica geométrica* G es una pareja (V, E) de conjuntos finitos con las siguientes propiedades:

- $V \subseteq \mathbb{R}^2$ (V es un conjunto de puntos).
- Cada arista de E es un segmento entre dos vértices (dos elementos de V).
- El interior de una arista no contiene ningún vértice.

Nótese que cualquier conjunto S de puntos en posición general en el plano, induce de manera natural una gráfica geométrica completa. Diremos que una gráfica geométrica completa K_n *está definida sobre* S, si K_n tiene como conjunto de vértices a S. A los elementos de V les llamamos indistintamente puntos o vértices. A los elementos de E les llamados indistintamente segmentos o aristas. Un *isomorfismo* entre dos gráficas G y G' es una biyección $f : V \to V'$ en la que $xy \in E(G)$ si y sólo si $f(x)f(y) \in E(G')$, $\forall x, y \in V$. Un *encaje rectilíneo* de una gráfica G en el plano, es un isomorfismo entre G y una gráfica geométrica G. Llamamos a dicho encaje un *dibujo* de G, o una gráfica geométrica de G. Informalmente, se puede decir que un encaje (rectilíneo) de una gráfica G, envía cada vértice de G a un punto en el plano, y cada arista a un segmento entre dos vértices (puntos). En la figura 2.3 mostramos un encaje de una gráfica completa con 7 vértices.

Puesto que una gráfica geométrica es un encaje en el plano de una gráfica, las gráficas geométricas heredan propiedades de las gráficas. Por esta razón los conceptos definidos anteriormente para gráficas se pueden extender, de manera natural, a gráficas geométricas. Sin embargo, al ser encajes en el

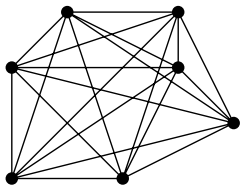

Figura 2.3: Gráfica geométrica completa con siete vértices (K_7)

plano, las gráficas geométricas cuentan con nuevas propiedades derivadas de la geometría del plano. En particular, en este trabajo de tesis estudiaremos el cruce y la intersección de gráficas geométricas. Definimos estos conceptos a continuación.

Sea $\mathsf{G} = (V, E)$ una gráfica geométrica. Decimos que dos aristas de G son *adyacentes* si indicen en un mismo vértice, de lo contrario decimos que las aristas son *disjuntas*. Dos aristas disjuntas (en vértices) de G se *cruzan* si tienen un punto interior en común. Dos aristas de G se *intersectan* si se cruzan o si son adyacentes. En la figura 2.4a mostramos dos aristas que se cruzan, y en la figura 2.4b mostramos dos aristas que se intersectan.

Figura 2.4: (a) Dos aristas que se cruzan. (b) Dos aristas que se intersectan.

Sean H_1 y H_2 dos gráficas geométricas. Decimos que H_1 y H_2 *se cruzan* si existe una arista en H_1 y una arista en H_2 que se cruzan. Decimos que H_1 y H_2 *se intersectan* si existe una arista en H_1 y una arista en H_2 que se intersectan. En la figura 2.5a mostramos dos encajes H_1 y H_2 de la gráfica $K_{1,3}$ que se cruzan. En la figura 2.5b mostramos dos encajes H_1 y H_2 de la gráfica $K_{1,3}$ que se intersectan.

Para n fija hay un número infinito de conjuntos de n puntos en el plano. De la misma manera, dada una gráfica G, hay un número infinito de encajes de G en el plano. Existe una clasificación combinatoria para los conjuntos de puntos en el plano, de tal manera que cualquier conjunto de puntos en la misma categoría es combinatoriamente equivalente. A continuación damos la definición formal de dicha clasificación.

El *tipo de orden* de $\{p_1, ..., p_n\}$, un conjunto de n puntos, es una función que asigna a cada tripleta ordenada i, j, k en $\{1, ..., n\}$ la orientación de la tripleta de puntos p_i, p_j, p_k. Decimos que S_1 y S_2 dos conjuntos de puntos son

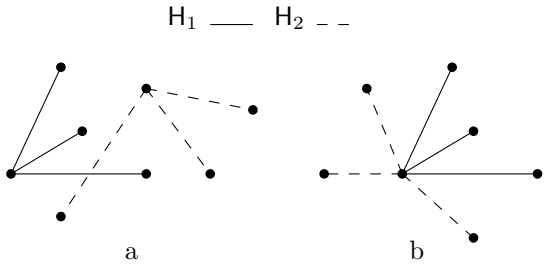

Figura 2.5: (a) Dos encajes de $K_{1,3}$ que se cruzan. (b) Dos encajes de $K_{1,3}$ que se intersectan

combinatoriamente equivalentes si tienen el mismo tipo de orden, es decir, si existe una biyección entre S_1 y S_2 tal que cualquier tripleta en S_1 tenga la misma orientación que su tripleta correspondiente en S_2. En la figura 2.6 mostramos dos conjuntos de 4 puntos con el mismo tipo de orden. En la figura 2.7 mostramos los dos únicos tipos de orden distintos para 4 puntos.

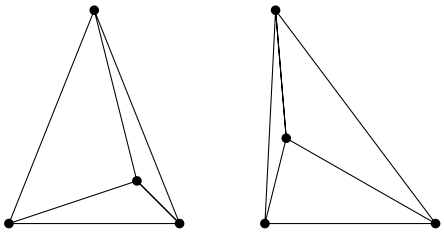

Figura 2.6: Dos conjuntos de 4 puntos con el mismo tipo de orden.

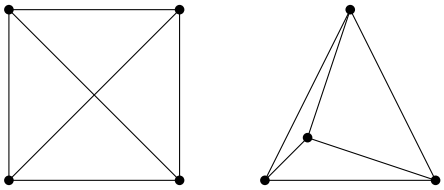

Figura 2.7: Los únicos dos tipos de orden distintos para $n = 4$.

Encontrar, o incluso contar, los tipos de orden para cualquier valor de n no es trivial [3]. Esto es porque el crecimiento del número de tipos de orden distintos es aproximadamente doble exponencial con respecto a n. Los autores de [3] crearon una base de datos [1] donde almacenan todos los tipos de orden para conjuntos de $n \leq 11$ puntos. En la tabla 2.1 mostramos

Número de puntos	Cantidad de tipos de orden	Tamaño de la base de datos
3	1	6 bytes
4	2	16 bytes
5	3	30 bytes
6	16	192 bytes
7	135	1.89 KB
8	3 315	53.04 KB
9	158 817	5.717412 MB
10	14 309 547	572.38188 MB
11	2 334 512 907	96 GB

Tabla 2.1: Tipos de orden para $n \leq 11$ puntos. Tomado de [1].

el número de tipos de orden para $n \leq 11$ así como el tamaño del archivo que los almacena. Debido a que el tamaño de la base de datos para 11 puntos es de 96 GB, esta no se encuentra disponible para descargar en línea.

A continuación mostramos algunas de las propiedades más importantes de los conjuntos de puntos de la base de datos [1]:

1. Las coordenadas de los puntos están codificadas como enteros sin signo. Para $n \leq 8$ como enteros sin signo de 8 bits y para $n = 9, 10$ con enteros sin signo de 16 bits.

2. Todos los conjuntos de puntos están en posición general.

3. Todos los puntos de un conjunto tienen coordenada x diferente. Todos los puntos de un conjunto tienen coordenada y diferente.

4. Los conjuntos de puntos están ordenados de forma creciente de acuerdo a su menor λ-matriz [3]. En otras palabras, el punto p_1 es un punto extremo (está en el cierre convexo) y los puntos $\{p_2, p_3, ..., p_n\}$ están ordenados en sentido horario alrededor de p_1.

Es de nuestro interés encontrar conjuntos de gráficas que se cruzan definidas sobre conjuntos de puntos combinatoriamente distintos y con menos de 11 puntos. A continuación definiremos dichos conjuntos de gráficas.

Definición 1. *Sea S un conjunto de n puntos y sea H una gráfica (no geométrica). Una H-*crossing family *de S es un conjunto de gráficas geométricas que cumple con las siguientes propiedades:*

- *Cada gráfica es un encaje de H sobre S.*
- *Cada pareja de gráficas se cruzan.*

La *cardinalidad* de una H-Crossing Family es la cantidad de elementos en el conjunto. En la figura 2.8a mostramos una $K_{1,3}$-crossing family de tamaño 2. Para H fija, el número de H-crossing family de S, que denotamos como $\#\text{-}Hcf(S)$, es la cardinalidad de la H-crossing family de S más grande posible. Es decir, si consideramos el conjunto de todas las H-crossing families de S, y tomamos la familia de cardinalidad máxima, a la cardinalidad de dicha familia le llamamos $\#\text{-}Hcf(S)$. Dado un entero n, el número de H-crossing family de n, que denotamos como $\#\text{-}Hcf(n)$, es la cardinalidad de la H-crossing family más grande sobre todos los conjuntos de n puntos. Es decir:

$$\#\text{-}Hcf(n) = max\{\#\text{-}Hcf(S) | S \text{ es un conjunto de } n \text{ puntos } \}$$

Sea K_n una gráfica geométrica completa definida sobre un conjunto S de n puntos. Decimos que $\mathsf{T} \subseteq \mathsf{K}_n$ es un *thrackle* de K_n si cualesquiera dos aristas de T se cruzan o se intersectan. Equivalentemente, un thrackle es un conjunto de segmentos en el plano tal que cada dos segmentos se cruzan o comparten uno de sus extremos. Decimos que T es *máximo* si el número de aristas de T es igual al número de vértices de K_n. En la figura 2.8b mostramos un thrackle máximo con 8 vértices.

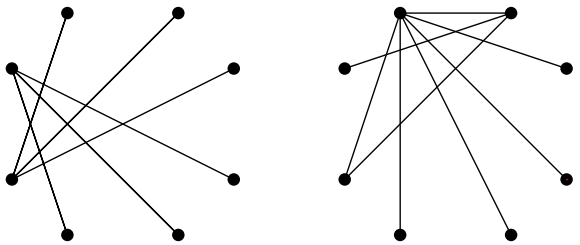

Figura 2.8: $K_{1,3}$-crossing family y thrackle máximo.

Sea P un conjunto de n puntos en el plano. Sea $|p_1p_2|$ la distancia euclidiana entre dos puntos. Decimos que dos puntos $p_a, p_b \in P$ están a *distancia máxima* si $d(p_ap_b) \geq d(p_ip_j), \forall p_i, p_j \in P$ con $i, j \in \{1, ..., n\}, i \neq j$. Decimos que dos puntos $p_a, p_b \in P$ están a *distancia mínima* si $d(p_ap_b) \leq d(p_ip_j), \forall p_i, p_j \in P$ con $i, j \in \{1, ..., n\}, i \neq j$. Sea $M(P)$ el número de parejas de P a *distancia máxima*. Definimos $M(n)$ como

$$M(n) = max\{M(P)|P \text{ es un conjunto de } n \text{ puntos}\} \text{ [4].}$$

A continuación relacionaremos las distancias máximas de un conjunto de puntos con los thrackles. En el siguiente teorema caracterizamos los conjuntos de segmentos de distancias máximas para un conjunto de n puntos. La demostración del siguiente lema y del teorema 2, las tomamos íntegramente de [4] [5].

Lema 1. *Los segmentos de longitud máxima determinados por un conjunto finito de puntos en el plano siempre se intersectan.*

Demostración. Sea m la distancia máxima entre parejas de P. Supongamos que los vértices $a, b, c, d \in P$ forman dos aristas ab y cd tales que, $d(ab) = d(cd) = m$ y además ab y cd no se intersectan, entonces los cuatro vértices forman un cuadrilátero convexo con ab y cd como dos de sus lados. Como los ángulos internos del cuadrilátero suman 360° alguno de los ángulos internos, sin pérdida de generalidad $\angle abc$, es mayor o igual que 90° y por lo tanto la diagonal ac cumple que $d(ac) > d(ab) = m$. Ver figura 2.9. Esto contradice el hecho de que m sea la distancia máxima. Por lo tanto cualesquiera dos segmentos de longitud m se intersectan.

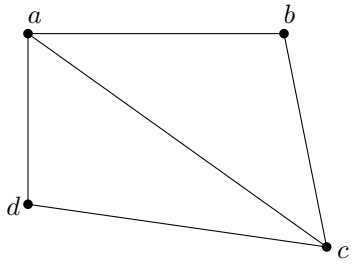

Figura 2.9: Las aristas de distancia máxima ac y cd se intersectan

□

Teorema 2. *Si P es un conjunto de n puntos en el plano entonces la distancia máxima entre cualquier par de ellos ocurre a lo más n veces.*

Demostración. Sea $P = \{p_1, p_2, ..p_n\}$ un conjunto de puntos en el plano y sea m la distancia máxima entre parejas de P. Procederemos por inducción. Si $n = 2$ ó $n = 3$ el teorema es cierto.

Conectamos p_i con p_j siempre que $p_i p_j = m$. Podemos encontrar dos casos:

1. Cada p_i está conectado con a lo más otros dos elementos de P. La suma de los grados de los puntos de P es $2n$. Como cada pareja de puntos fue contada dos veces, entonces el número de parejas de puntos a distancia m es menor o igual a n.

2. Algunos de los elementos de P, supongamos p_1, está conectado con tres o más, supongamos p_2, p_3 y p_4 y además p_1p_3 está entre p_1p_2 y p_1p_4. p_3 no está conectado con ninguno otro p_i, pues de estarlo, p_3p_i tendría que intersectar tanto a p_1p_2 como a p_1p_4, lo cual es imposible.

   Ahora aplicando la hipótesis de inducción a $P - \{p_3\}$ hay a lo más $n-1$ distancias m y p_3 solo contribuye con una más, por lo tanto en P hay a los más n distancias m.

□

Notemos que, como consecuencia del lema 1, un conjunto de distancias máximas es también un thrackle. Por lo tanto, podemos relacionar el teorema 2 con los thrackles. Tomamos el siguiente teorema y su demostración de [6].

Teorema 3. *Cualquier thrackle tiene a los más tantas aristas como vértices.*

Demostración. Sea G un thrackle. Decimos que un vértice $v \in V(G)$ es *puntiagudo* si todas las aristas que inciden en v se encuentran en un semiplano definido por una línea que pasa por v. En la figura 2.10 mostramos un ejemplo de un vértice puntiagudo v, y un vértice que no es puntiagudo, u. Por cada vértice puntiagudo v, borramos de G la arista más a la izquierda incidente en v, ésta es, la primera arista encontrada en el sentido horario en el semiplano de v. Observemos que si hacemos esto eliminamos todas las aristas, entonces la demostración está completa. Puesto que hemos eliminado exactamente una artista por cada vértice puntiagudo. Si nos quedamos con una arista $uv \in E(G)$, entonces uv no era la arista más a la izquierda de v ni la arista más a la izquierda de u. Por lo tanto G contenía dos aristas, uv' y $u'v$ que eran las aristas más a la izquierda de u y v, respectivamente. Además uv' y $u'v$ están en lados distintos del semiplano definido por la línea que pasa por uv, por lo tanto uv' y $u'v$ no se intersectan ver Figura 2.11, lo cual contradice que G era un thrackle.

□

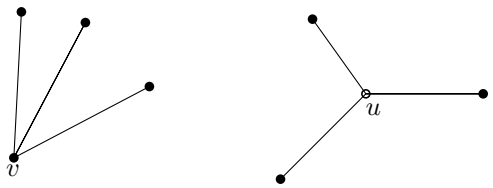

Figura 2.10: (a) Un vértice puntiagudo. (b) Un vértice no puntiagudo

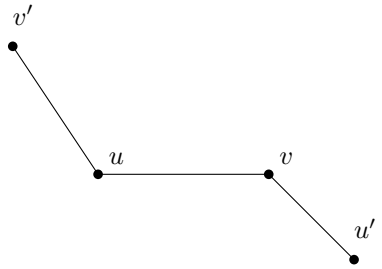

Figura 2.11: uv' y $u'v$ no se intersectan

Capítulo 3

Estado del Arte

Originalmente el problema de las crossing families fue planteado por los autores de [7]. En ese artículo prueban de forma algorítmica que, dado cualquier conjunto de n puntos en el plano, siempre podemos encontrar una crossing family de al menos $\sqrt{n/12}$ segmentos en tiempo $O(n \log n)$. A continuación damos algunas definiciones necesarias para hacer una revisión de los resultados en [7]. Tomamos las definiciones del mismo artículo.

Le llamamos *crossing family* de S a la H-crossing family en la que H es una gráfica completa de dos vértices, es decir, H es una arista. Definimos una *secuencia* como una lista de números tal que cada uno de sus elementos tiene una posición única en la lista. Si los elementos de una secuencia son números reales, decimos que es una *secuencia de números reales*. Decimos que una secuencia de números reales es *descendente* si cada dos elementos e_i, e_{i+1} (consecutivos en la secuencia) cumplen que $e_i > e_{i+1}$. De la misma manera decimos que una secuencia de números reales es *ascendente* si cada dos elementos e_i, e_{i+1} (consecutivos en la secuencia) cumplen que $e_i < e_{i+1}$. Sean A y B dos conjuntos disjuntos de puntos con la misma cardinalidad. Llamamos a los elementos de A puntos *rojos* y a los elementos de B puntos *azules*. Decimos que A y B se *cruzan* si existe una crossing family de $A \cup B$, en la que cada arista conecta un vértice en A con un vértice en B. Si cada arista de una crossing family une vértices de colores distintos entonces decimos que es *coloreada*. Decimos que un conjunto A *evade* a un conjunto B si no existe una línea que pase por dos puntos de A que intersecte el cierre convexo de B. Es decir, que cualquier vértice de B *ve* los puntos de A en el mismo orden. Los conjuntos A y B se *evaden mutuamente* si A evade a B y B evade a A. En la figura 3.1(a) mostramos dos conjuntos A y B de puntos que se evaden mutuamente. En la figura 3.1(a) mostramos una crossing family definida

sobre $A \cup B$.

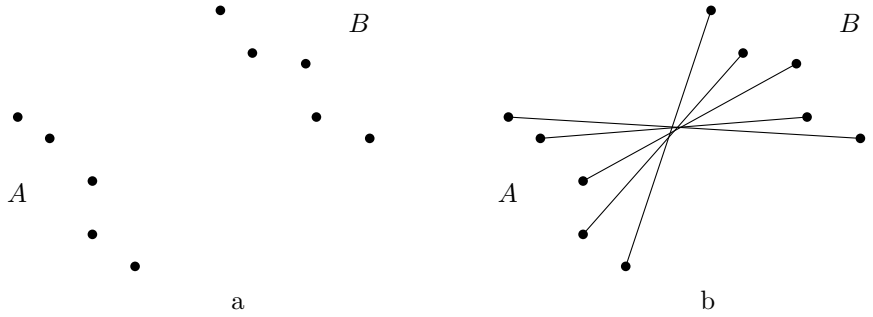

Figura 3.1: (a) dos conjuntos A y B de puntos que se evaden mutuamente. (b) una crossing family definida sobre $A \cup B$.

Los autores de [7] dieron un método para encontrar conjuntos que se evaden mutuamente y que tienen tamaño $\Omega(\sqrt{n})$. Mostraron también que si dos conjuntos A y B se evaden mutuamente y tienen la misma la cardinalidad, entonces se cruzan. A continuación detallamos algunos de sus resultados.

Lema 2 ([7])**.** *Sea S un conjunto finito de puntos en el plano, y sea $\mathcal{L}$ cualquier línea que divide a S en dos subconjuntos. Es posible encontrar otra línea M, que simultáneamente divide a los puntos en dos subconjuntos con cualquier proporción deseada.*

Lema 3 (Erdős - Szekeres [8])**.** *Para cualquier secuencia de números reales, de tamaño n, existe una subsecuencia o bien ascendente o bien descendente, de tamaño $\sqrt{n}$.*

El siguiente teorema es una variante sin colorear del teorema 1 presentado por los autores de [7].

Teorema 4 ([7])**.** *Dado un conjunto S de n puntos en el plano, podemos encontrar dos subconjuntos A y B, con $A \cap B = \emptyset$, de tamaño $\sqrt{n/12}$ que se evaden mutuamente.*

Demostración. Sea $\mathcal{L}$ una línea horizontal tal que todos los puntos de S estén por debajo de $\mathcal{L}$. Movemos $\mathcal{L}$ disminuyendo su coordenada y y manteniendo fija su pendiente, hasta que dejemos al menos $n/2$ puntos de S arriba de $\mathcal{L}$. Llamaremos A al conjunto de puntos arriba de $\mathcal{L}$ y llamaremos B al conjunto de puntos debajo de $\mathcal{L}$. Por el lema 2, existe una línea $\mathcal{M}$ tal que deja $n/12$ puntos de A a su izquierda y $n/12$ puntos de B a su izquierda. Ver figura 3.2.

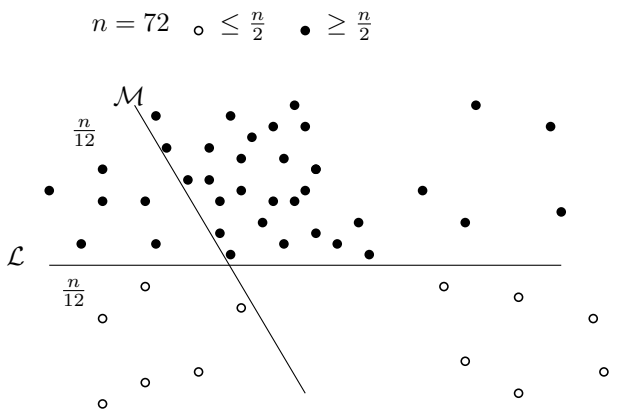

Figura 3.2: Partición del plano usada en el teorema 4. Los puntos rellenos pertenecen al conjunto A. Los puntos vacíos pertenecen al conjunto B.

Sea $\mathcal{N}$ una recta paralela a $\mathcal{M}$ y tal que todos los puntos de S están a su izquierda. Movemos $\mathcal{N}$ disminuyendo su coordenada x y manteniendo fija su pendiente, hasta que dejemos $n/12$ puntos de A a la derecha de $\mathcal{N}$ o dejemos $n/12$ puntos de B a la derecha de $\mathcal{N}$ (lo que suceda primero). Supongamos que dejamos $n/12$ puntos de B a la derecha de $\mathcal{N}$. Ver figura 3.3. En la región $\mathcal{R}$ hay al menos $n/3$ puntos, y en las regiones $\mathcal{B}_1$ y $\mathcal{B}_2$ hay al menos $n/12$ puntos en cada una.

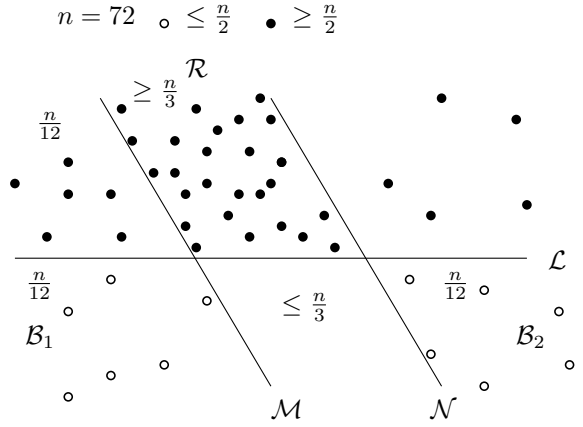

Figura 3.3: Distribución de los puntos de S en 6 regiones.

Sea $\mathcal{V}$ una recta vertical tal que todos los puntos de S se encuentren a su derecha. Movemos $\mathcal{V}$ incrementado su coordenada x y manteniendo cons-

tante su pendiente, y agregamos los puntos de $\mathcal{R}$ a una secuencia SE en el orden en que $\mathcal{V}$ los vaya encontrando. Por el Lema 3 existe una subsecuencia de SE ascendente o descendente en y de tamaño $\sqrt{n/3}$. Supongamos que la subsecuencia es descendente. En la figura 3.4 mostramos con un cuadrado los puntos de la subsecuencia. Observemos que los puntos de la subscuencia evaden a la región $\mathcal{B}_1$, esto es porque para cada punto (p_i, p_j) de la subsecuencia, p_j siempre está a la derecha de p_i, por lo tanto cualquier línea de la subsecuencia tiene como límite a $\mathcal{M}$.

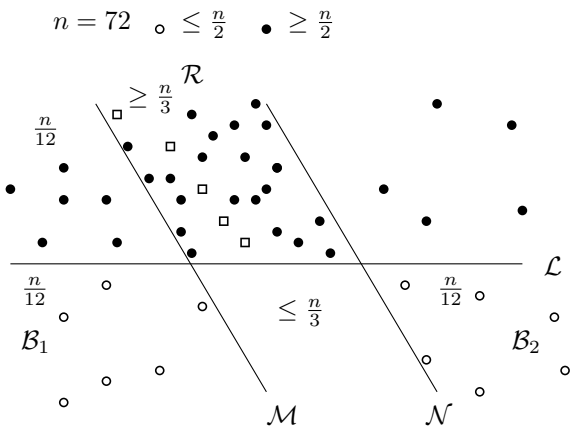

Figura 3.4: Subsecuencia en $\mathcal{R}$ de tamaño $\sqrt{n/3}$.

Aplicamos una transformación de tal manera que las líneas $\mathcal{M}$ y $\mathcal{N}$ se vuelvan verticales. Llamamos x al punto intermedio de la subsecuencia en $\mathcal{R}$ y llamamos $\mathcal{R}_1$ a los puntos que anteceden a x, de la misma manera llamamos $\mathcal{R}_2$ a los puntos que preceden a x. Tomamos los puntos de $\mathcal{B}_1$ en coordenadas polares y los agregamos a una secuencia en orden creciente de acuerdo a su ángulo respecto a x. Por el lema 3 existe una subsecuencia de puntos en $\mathcal{B}_1$ de tamaño $\sqrt{n/12}$ ascendente o descendente respecto su distancia con x. Supongamos que encontramos una subsecuencia descendente. En la figura 3.5 mostramos con una equis a los puntos de la subsecuencia de $\mathcal{B}_1$ así como los segmentos definidos con x para destacar las distancia. Observemos que para cada punto (p_i, p_j) de la subsecuencia descendente en $\mathcal{B}_1$, p_j siempre está debajo de la recta definida por p_i, x, además como p_j está más cerca de x que p_i entonces cualquier línea definida por dos puntos de la subsecuencia en $\mathcal{B}_1$ tiene como límite a la línea definida por (p_1, x). Por lo tanto la subsecuencia descendente de $\mathcal{B}_1$ evade a $\mathcal{R}_1$.

Como ya probamos, $\mathcal{R}$ evade a $\mathcal{B}_1$ y $\mathcal{B}_1$ evade a $\mathcal{R}_1$. Además como $\mathcal{R}$ era de tamaño $\sqrt{n/3}$ y $\mathcal{R}_1$ contiene la mitad de puntos de $\mathcal{R}$ entonces $\mathcal{R}_1$ es de

tamaño $\sqrt{n/12}$. Por lo tanto $\mathcal{R}_1$ y $\mathcal{B}_1$ son del mismo tamaño y se evaden mutuamente.

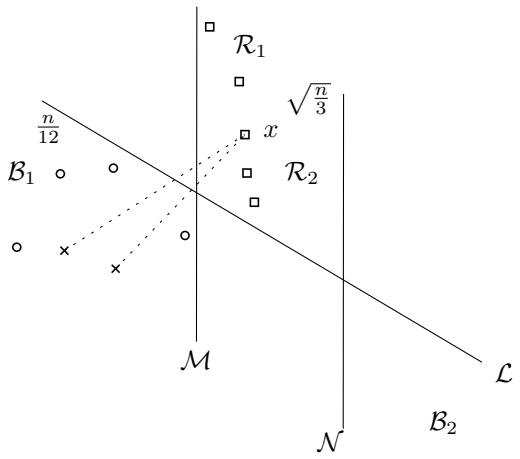

Figura 3.5: Subsecuencia en $\mathcal{B}_1$ de tamaño $\sqrt{n/12}$.

□

En el mismo artículo ([7]), los autores analizan las condiciones que deben cumplir dos conjuntos de puntos que se evaden mutuamente para que se crucen. El procedimiento descrito a continuación es una variación sin colores del descrito por los autores de dicho artículo.

Consideremos dos conjuntos de puntos X y Y los cuales están separados por una recta $\mathcal{L}$. Decimos que un punto $x \in X$ *ve* a un punto $y \in Y$ *con rango* i si y es el i-ésimo punto en sentido antihorario visto por x, y viceversa. Decimos que X y Y cumplen la *condición de rango* si existe una etiquetación $x_1, ..., x_s$ de X y una etiquetación $y_1, ..., y_s$ de Y tal que para cada i, x_i ve a y_i con rango i, y viceversa. Si la etiquetación cumple que x_i ve a y_j con rango j para todo i y para todo j, decimos que la condición de rango de X y Y es *fuerte*. En la figura 3.6 mostramos dos conjuntos X y Y que cumplen la condición de rango fuerte.

Proposición 1 ([7])**.** *Sean X y Y dos conjuntos de puntos separados por una recta, entonces:*

1. *X y Y se pueden cruzar si y sólo si cumplen la condición de rango.*
2. *X y Y se evaden mutuamente si y sólo si cumplen la condición de rango fuerte.*

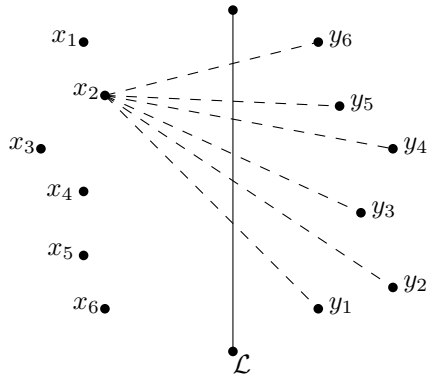

Figura 3.6: Conjuntos X y Y que cumplen la condición de rango fuerte

Debido a que la condición de rango fuerte implica la condición de rango podemos obtener el siguiente corolario.

Corolario 1 ([7])**.** *Un par de conjuntos de puntos se pueden cruzar si y sólo si se evaden mutuamente y tienen la misma cardinalidad.*

Demostración. Sea $\mathcal{L}$ una línea vertical, sea X un conjunto de puntos a la izquierda de $\mathcal{L}$, y sea Y un conjunto de puntos a la derecha de $\mathcal{L}$.

Supongamos que X y Y se cruzan. Sean $l_1, ..., l_i$ los segmentos de una crossing family ordenados de forma creciente con respecto a su pendiente. Etiquetamos como x_i y y_i a los puntos extremos de un segmento l_i. Como los segmentos $l_1, ..., l_{i-1}$ tienen pendiente menor que l_i, y además se intersectan, x_i ve a $y_1, ..., y_{i-1}$ antes que a y_i. De la misma manera, x_i ve a $y_{i+1}, ..., y_s$ después de y_i, por lo tanto x_i ve a y_i con rango i. Por la misma razón, y_i ve a x_i con rango i.

Supongamos que existe una etiquetación $x_1, ..., x_s$ de X, y una etiquetación $y_1, ..., y_s$ de Y, que satisfacen la condición de rango. Probaremos, por inducción sobre s, que los segmentos $l_1, ..., l_i$ son una crossing family.

El caso con $s = 1$ es trivial.

Consideremos la línea l_s definida por los puntos x_s y y_s. Por la condición de rango, $X - x_s$ y $Y - y_s$ están en lados opuestos de l_s. Por lo tanto, la recta $x_s y_s$ intersecta a la recta $x_i y_i$ si intersecta a l_i. Además la pendiente de $x_i y_i$ es menor que la pendiente de $x_s y_s$ para toda $i < s$.

Sea A un conjunto de segmentos que no intersectan a $x_s y_s$. Ordenamos los elementos de A con respecto a su intersección con $\mathcal{L}$. Si A no es vacío entonces, sin pérdida de generalidad, A contiene un segmento que intersecta a $\mathcal{L}$ arriba de $x_s y_s$. Elegimos el segmento de A que intersecte más arriba

a $\mathcal{L}$ y lo nombramos $x_a y_a$. La línea l_a definida por x_a y y_a no intersecta a $x_s y_s$. Como X y Y cumplen la condición de rango, entonces existen $a-1$ puntos de X arriba de l_a y $a-1$ puntos de Y abajo de l_a. Notemos que y_s es parte de los $a-1$ puntos de Y debajo de l_a, pero x_s no es parte de los $a-1$ puntos de X arriba de l_a. Por lo tanto existen dos puntos x_b y y_b que están arriba de l_a. Entonces $x_b y_b$ no intersecta a $x_s y_y$ y además intersecta a $\mathcal{L}$ más arriba que $x_a y_a$, lo que contradice nuestra hipótesis. Por lo tanto A es vacío, y $x_s y_s$ intersecta a todos los segmentos $x_i y_i$.

Para demostrar que $x_i y_i$ intersecta a $x_j y_j$ para toda $i < j < s$, observemos que como la pendiente de $x_s y_s$ es mayor que la de los demás segmentos, la condición de rango se conserva cuando eliminamos $x_s y_s$. Por lo tanto, por inducción, $X - x_s$ y $Y - y_s$ forman una crossing family. En la figura 3.7 mostramos una crossing family que cumple todas las condiciones mencionadas.

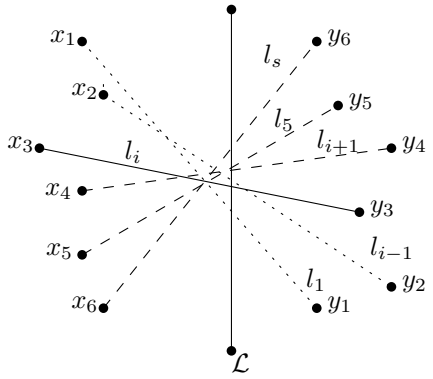

Figura 3.7: Conjuntos X y Y que forman una crossing family

□

Pavel Valtr en [9] encontró que, si un conjunto de puntos en el plano es denso, entonces dicho conjunto tiene una crossing family de tamaño lineal. A continuación mostramos sus definiciones y resultados. Sea $\mathcal{P}$ un conjunto de n puntos en el plano. Decimos que $\mathcal{P}$ es *denso* si el radio entre la distancia máxima y la distancia mínima en $\mathcal{P}$ es de orden $O(\sqrt{n})$. Definimos al radio de un conjunto $\mathcal{P}$ como

$$q(\mathcal{P}) = \frac{max\{d(ab) : a, b \in \mathcal{P}\}}{min\{d(ab) : a, b \in \mathcal{P}, a \neq b\}},$$

donde $d(ab)$ es la distancia euclidiana entre a y b. Para $\alpha > 0$, decimos que $\mathcal{P}$ es α-denso si $q(\mathcal{P}) \leq \alpha\sqrt{n}$. Valtr destaca que podemos encontrar un

conjunto α-denso arbitrariamente grande si y sólo si $\alpha \geq \sqrt{2\sqrt{3}/\pi} \approx 1{,}05$. Uno de los resultados de Valtr y de interés para esta tesis es el teorema 16 presentado en [9]. A continuación enunciamos el teorema.

Teorema 5 ([9]). *Para cualquier $\alpha > \sqrt{2\sqrt{3}}/\pi$ y $\epsilon > 0$, cualquier conjunto $\mathcal{P}$ α-denso de tamaño n tiene una crossing family de tamaño $\Omega(n^{1-\epsilon})$.*

Años más tarde los autores de [10] demostraron que siempre existe una crossing family de tamaño 3 para conjuntos de $9 \leq n \leq 16$ puntos. Además definieron una nueva estructura donde los elementos se cruzan, llamada (j,k)-malla. Las definiciones y resultados encontrados por los autores de [10] son los siguientes.

Sea V_1 un conjunto de j segmentos disjuntos (en vértices) y sea V_2 un conjunto de k segmentos disjuntos (en vértices). Una (j,k)-malla es la unión de V_1 y V_2 tal que cada segmento de V_1 cruza a todos los elementos de V_2. Ver figura 3.8. Definimos como $c(k)$ al mínimo entero tal que cualquier conjunto de $c(k)$ puntos en el plano tiene una crossing family de tamaño k. De la misma manera, definimos como $\#(j,k)$ como el mínimo entero tal que cualquier conjunto de $\#(j,k)$ puntos en el plano tiene una (j,k)-malla.

Los autores de [10] acotaron el número de puntos que se necesitan para encontrar una crossing family de 3 segmentos. Este resultado es $9 \leq c(3) \leq 16$. Además mostraron que cualquier conjunto de 8 puntos en el plano tiene una (1,2)-malla.

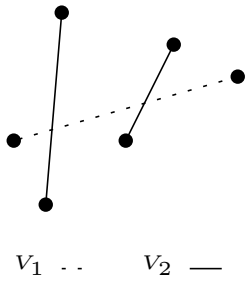

Figura 3.8: (1,2)-malla

El trabajo más reciente es el realizado por los autores de [11]. En este artículo, los autores dan una cota inferior del tamaño de una P_3-crossing family para cualquier conjunto de puntos en el plano. Las definiciones y resultados obtenidos por los autores de [11] se muestran a continuación.

Sea G una gráfica con n vértices y sea S un conjunto de puntos en el plano. Un *camino* de G es una secuencia $P = (v_1, ..., v_{k+1})$ con $k < n$

de vértices de G, tal que no existen vértices repetidos en P y dos vértices consecutivos de P forman una arista. Un k-camino de S es un camino con k aristas definidas sobre los vértices de S. Dos k-caminos de S son *disjuntos en vértices* si no comparten ningún punto en S. Decimos que dos k-caminos se *cruzan* si alguna de sus aristas se cruzan. Dos de los resultados propuestos por los autores en [11] y que son de interés para esta tesis son los siguientes teoremas.

Teorema 6 ([11])**.** *Para cualquier conjunto de n puntos en el plano siempre existe una P_3-crossing family de tamaño $\lfloor n/4 \rfloor$.*

Teorema 7 ([11])**.** *Para cualquier conjunto de n puntos en el plano, existe una C_3-crossing family de tamaño $\lfloor n/6 \rfloor$ (C_3 denota al ciclo de tamaño 3).*

Las demostraciones a dichos teoremas pueden ser consultadas en [11]. Tanto nuestro trabajo como [9], [10] y [11] son variantes del problema planteado por los autores de [7]. Por esta razón solo hemos presentado las demostraciones de [7].

Capítulo 4

Crossing Families de $2K_2$

Denotamos como K_2 a la gráfica completa con dos vértices y como $2K_2$ a la gráfica que tiene dos componentes conexas, cada una de las cuales es isomorfa a K_2. En la figura 4.1 mostramos dos encajes distintos de la gráfica $2K_2$. En este capítulo estudiamos conjuntos de gráficas, isomorfas a $2K_2$, que se cruzan. Sean H_1 y H_2 dos gráficas geométricas. Decimos que H_1 y H_2 se *cruzan* si existe una arista en H_1 y una arista en H_2 que se cruzan. Llamamos $2K_2$-*crossing family* a un conjunto de gráficas isomorfas a $2K_2$ en el que cualesquiera dos de sus elementos se cruzan. En este capítulo estudiamos el problema de encontrar el número de $2K_2$-crossing family para conjuntos con 8, 9 y 10 puntos. Definimos al número de $2K_2$-crossing family como $\#2K_2cf(n) = max\{\#2K_2cf(S)|S$ es un conjunto de n puntos$\}$, donde $\#2K_2cf(S)$ es la cardinalidad de la $2K_2$-crossing family de S más grande posible. En otras palabras, deseamos determinar la $2K_2$-crossing family de cardinalidad máxima sobre todos los conjuntos de tamaño 8, 9 y 10.

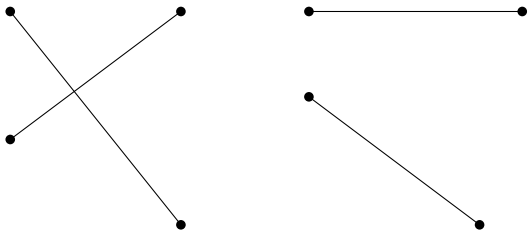

Figura 4.1: Dos gráficas isomorfas a $2K_2$.

Dado n, el siguiente teorema determina el número de parejas de segmentos disjuntos en vértices para un valor de n. A continuación damos algunas definiciones que necesitaremos para el teorema.

Una función $f : A \to B$ es *inyectiva* si y sólo si $f(x) = f(y)$ implica que $x = y$, $\forall x, y \in A$. Decimos que una función $f : A \to B$ es *sobreyectiva* si y solo si $\forall b \in B$ existe $a \in A$ con $f(a) = b$. Por último, decimos que $f : A \to B$ es una biyección si f es inyectiva y sobreyectiva.

Teorema 8. *Sea K_n una gráfica completa. Sea P el conjunto de todas las parejas de aristas de K_n disjuntas en vértices. Considérese cualquier subgráfica de K_n con $n-1$ vértices, sea Q el conjunto de todas las parejas de aristas no necesariamente disjuntas de K_{n-1}. La cardinalidad de P es igual a la cardinalidad de Q.*

Demostración. Para demostrar el teorema daremos una biyección $f : Q \to P$. Daremos la función f en dos partes. La primera parte envía las parejas de aristas disjuntas en vértices, de Q, directamente a parejas de aristas de P. La segunda parte envía las parejas de aristas que no son disjuntas, de Q, a parejas de aristas de P (nótese que las parejas de P son todas disjuntas).

Sean $\{v_1, v_2, ..., v_{n-1}\}$ los vértices de K_{n-1}. Sea K_n la gráfica resultante de agregar un vértice v_n a la gráfica K_{n-1}, y todas las aristas entre v_n y los vértices de K_{n-1}. Es decir, el conjunto de vértices de K_n es $\{v_1, v_2, ..., v_{n-1}, v_n\}$. Consideremos el conjunto Q de K_{n-1} y el conjunto P de K_n. Notemos que, como K_{n-1} es una subgráfica de K_n, entonces todas las parejas de aristas de K_{n-1} también son parejas de aristas de K_n. En particular, todas las parejas de aristas disjuntas en vértices que pertenecen a Q también pertenecen a P. Por lo tanto, la primera parte de nuestra función envía directamente las parejas de aristas de Q que son disjuntas en vértices a parejas de aristas de P. Es decir, $f(\{(v_i, v_j), (v_k, v_l)\} \in Q) = \{(v_i, v_j), (v_k, v_l)\} \in P$, $\forall i, j, k, l \in \{1, ..., n-1\}, i \neq j \neq k \neq l$. Nótese que no tomamos en cuenta el vértice v_n de K_n. Por lo tanto cualquier pareja de aristas que incida en v_n no es utilizada en esta parte de la función.

Para la segunda parte de la función utilizamos el vértice v_n para enviar las parejas de aristas de Q que no son disjuntas, como parejas de aristas de P. Es decir, tomaremos cada pareja de Q que comparten vértice y la enviaremos a una pareja de aristas disjuntas de P que incida en v_n. Esto es $f(\{(v_i, v_j), (v_k, v_i)\} \in Q) = \{(v_i, v_j), (v_k, v_n)\} \in P$, $\forall i, j, k \in \{1, ..., n-1\}, i \neq j \neq k$. Ver figura 4.2.

Para lograr esto debemos demostrar que el número de parejas de aristas no disjuntas de Q es igual al número de parejas de aristas que inciden en v_n, de P. Sea $Q' \subset Q$ el subconjunto de las parejas de aristas no disjuntas, de Q. Sea $P_{v_n} \subset P$ el subconjunto de las parejas de aristas de P de la forma $(v_i, v_j)(v_k, v_n)$, de P. Es decir, las parejas de aristas de P que inciden

en v_n. Por lo tanto, para la segunda parte de la función probaremos que $|Q'| = |P_{v_n}|$.

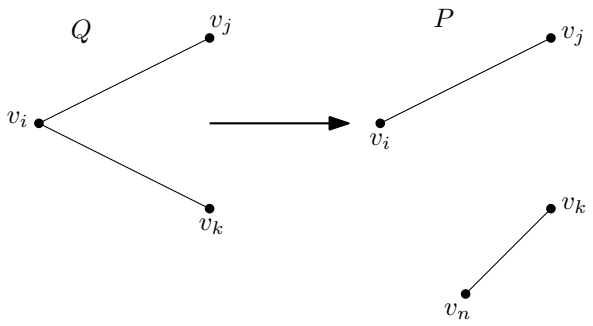

Figura 4.2: Uso del vértice $v_n \in P$ para separar una pareja de aristas de Q.

Para determinar la cardinalidad de Q' contamos la cantidad de parejas de aristas no disjuntas de la gráfica completa K_{n-1}. Esto es,

$$|Q'| = (n-1)\binom{n-2}{2}.$$

Para determinar la cardinalidad de P_{v_n} contamos la cantidad de parejas de aristas de K_n, disjuntas en vértices y que inciden en v_n. Esto es,

$$|P_{v_n}| = (n-1)\binom{n-2}{2}.$$

Como podemos observar $|Q'| = |P_{v_n}|$. Con esto podemos establecer la segunda parte de nuestra función, enviando cada pareja de aristas no disjuntas, de Q, como una pareja de aristas de P que incide en v_n.

Finalmente es claro que la función es tanto inyectiva como sobreyectiva. Por lo tanto la función $f : Q \to P$ es una biyección, lo cuál demuestra el teorema.

□

Para encontrar la $2K_2$-crossing family de cardinalidad máxima realizamos búsquedas sobre la base de datos [1] proporcionada por los autores de [3]. En la siguiente sección analizamos el tamaño del espacio de búsqueda para el problema de determinar $\#2K_2cf(n)$. Además presentamos el algoritmo implementado para determinar $\#2K_2cf(n)$.

4.1. Algoritmo para determinar $\#2K_2cf(n)$

4.1.1. Espacio de Búsqueda

El conjunto de vértices de una gráfica $2K_2$ tiene cardinalidad igual a cuatro. Si un conjunto S de n puntos tiene una $2K_2$-crossing family que contiene al menos dos elementos, entonces el número de puntos en el conjunto debe ser $n \geq 8$. Si una $2K_2$-crossing family contiene al menos tres elementos, entonces $n \geq 12$. La base datos [1] tiene todos los tipos de orden de conjuntos de puntos de tamaño menor o igual a 10. Por lo tanto,

$$0 \leq \#2K_2cf(n \leq 10) \leq 2.$$

El tamaño del espacio de búsqueda para determinar si $\#2K_2cf(n) = k$ está dado por dos factores:

- Las formas distintas de elegir k gráficas geométricas isomorfas a $2K_2$ y,
- la cantidad de tipos de orden que existen para ese valor de n.

A continuación analizamos cuál es el tamaño del espacio de búsqueda para encontrar todas las $2K_2$-crossing families de cardinalidad 2. Dado un conjunto de n puntos hay $\binom{n}{2}$ formas de elegir un segmento. Para cada segmento, hay $\binom{n-2}{2}$ formas de elegir un segundo segmento que no comparta puntos con el primero. Por lo tanto, hay $\binom{n}{2}\binom{n-2}{2}$ formas de elegir una pareja de segmentos disjuntos en vértices. Esto corresponde a las formas de elegir el primer elemento de una $2K_2$-crossing family. Sin embargo, observemos que cada segmento es contado dos veces. Por lo tanto, en realidad hay $\frac{\binom{n}{2}\binom{n-2}{2}}{2}$ formas de elegir el primer elemento de una $2K_2$-crossing family. De la misma manera, cada vez que fijemos el primer elemento de la $2K_2$-crossing family hay $\frac{\binom{n-4}{2}\binom{n-6}{2}}{2}$ formas de definir su segundo elemento. Por lo tanto, el tamaño de nuestro espacio de búsqueda tiene tamaño $\frac{\binom{n}{2}\binom{n-2}{2}\binom{n-4}{2}\binom{n-6}{2}}{4}$, esto es $O(n^8)$. En la tabla 4.1 mostramos el valor exacto del tamaño del espacio de búsqueda total. Es decir el número de parejas distintas que hay que considerar para cada valor de n entre 8 y 10.

4.1.2. Algoritmo

Para determinar $\#2K_2cf(n)$ implementamos un algoritmo que explora de manera exhaustiva el espacio de búsqueda. Es decir, para n fija, nuestro

n	Parejas de $2K_2$	Tipos de orden	Espacio de búsqueda total
8	630	3 315	2 088 450
9	5 670	158 817	900 492 390
10	28 350	14 309 547	405 675 657 450

Tabla 4.1: Tamaño del espacio de búsqueda para determinar $\#2K_2cf(n)$.

algoritmo construye todas las posibles parejas de $2K_2$ correspondientes a un tipo de orden. El algoritmo recibe como entrada un conjunto S de n puntos, correspondiente a un tipo de orden. Para dicho conjunto, el algoritmo construye un arreglo A con los $\binom{n}{2}$ segmentos posibles entre parejas de puntos. Con el arreglo A, el algoritmo construye un arreglo P con todas las parejas de segmentos disjuntas en vértices, de A. Notemos que P es de tamaño a lo sumo $\binom{\binom{n}{2}}{2}$. Cada elemento de P representa una gráfica isomorfa a $2K_2$. El algoritmo itera sobre todas las posibles parejas de P y verifica si se cruzan, de ser así incrementa la variable *n_cruces*. Al terminar de iterar sobre las parejas de P el algoritmo escribe en un archivo cuántos cruces existieron. Véase el Algoritmo 1.

A continuación analizamos la complejidad de nuestro algoritmo de búsqueda (Algoritmo 1). La instrucción de la línea 1 tiene un costo de $O(n^2)$, puesto que construye $\binom{n}{2}$ segmentos. Las líneas 2 y 3 tienen costo constante (debido a que la cantidad de memoria requerida para resolver el problema es menor a la cantidad de memoria RAM disponible en la computadora utilizada). Los ciclos de las líneas 4 y 5 tienen como límite al tamaño de S, observemos que $|S| = \binom{n}{2}$, por lo tanto el número de veces que se ejecutan los ciclos de las líneas 4 y 5 es:

$$\sum_{j=0}^{\binom{n}{2}} \sum_{k=j+1}^{\binom{n}{2}} 1.$$

Recordemos que el binomial $\binom{n}{k}$ es igual a $\frac{n!}{(n-k)!k!}$. Para analizar la complejidad de la suma anterior expresamos el binomial $\binom{n}{2}$ en forma de polinomio, es decir, $\binom{n}{2} = \frac{n(n-1)}{2}$. Por lo tanto, la suma con índice j se ejecuta $\frac{n(n-1)}{2}$ veces. Cuando j sea igual a cero, la suma con índice k se ejecuta $\frac{n(n-1)}{2} - 1$ veces. Cada vez que j se incremente, la segunda suma se ejecutará una vez menos hasta que se ejecute una sola vez. Por lo tanto la suma anterior es igual a la serie $1 + 2 + 3 + ... + \frac{n(n-1)}{2} - 1$. Aplicamos un

Algoritmo 1 Encuentra todas las $2K_2$-crossing families

Entrada: Conjunto S de n puntos correspondiente a un tipo de orden.
Salida: Número de $2K_2$-crossing families de cardinalidad máxima para S.

```
1: Construir un arreglo A con los (n 2) segmentos posibles.
2: i ← 0
3: Crear una arreglo P de tamaño ((n 2) 2)
4: for j ← 0 hasta |A| do
5:     for k ← j + 1 hasta |A| do
6:         if A[j] y A[k] no comparten vértices then
7:             P[i].s1 ← A[j]
8:             P[i].s2 ← A[k]
9:             i ← i + 1
10: n_cruces ← 0
11: for j ← 0 hasta i do
12:     for k ← j + 1 hasta i do
13:         if P[j] y P[k] no comparten vértice then
14:             if P[j] y P[k] se cruzan then
15:                 n_cruces ← n_cruces + 1
16: return n_cruces    ▷ Número de 2K2-crossing families de cardinalidad
    máxima de S
```

cambio de variable a la serie de tal manera que $t = \frac{n(n-1)}{2}$ y reescribimos la serie como $1 + 2 + 3 + ... + t - 1$, esta serie es igual a $\frac{t(t-1)}{2}$. Remplazamos t en la expresión anterior y obtenemos

$$\frac{\frac{n(n-1)}{2}(\frac{n(n-1)}{2} - 1)}{2} = \frac{n^4 - 2n^3 - n^2 + 2n}{8}.$$

La expresión anterior está asintóticamente dominada por $\frac{n^4}{8}$, por lo tanto la complejidad de los ciclos de las líneas 4 y 5 es de $O(n^4)$.

La línea 10 tiene complejidad constante. Observemos que los ciclos de las líneas 11 y 12 tienen como límite a la variable i, la cual es incrementada en dichos ciclos. Un conjunto con n puntos tiene $\binom{\binom{n}{2}}{2}$ parejas de segmentos, esto es $O(n^4)$. Por el teorema 8 sabemos que $i = \binom{\binom{n-1}{2}}{2}$, por lo tanto la suma de los ciclos de las líneas 11 y 12 es

$$\sum_{j=0}^{\binom{\binom{n-1}{2}}{2}} \sum_{k=j+1}^{\binom{\binom{n-1}{2}}{2}} 1.$$

Esta suma es igual a

$$\frac{n^8 - 12n^7 + 58n^6 - 144n^5 + 185n^4 - 84n^3 - 52n^2 + 48n}{128}.$$

Esta expresión está asintóticamente dominada por $\frac{n^8}{128}$, por lo tanto la complejidad de los ciclos de las líneas 11 y 12 es de $O(n^8)$. Por último, la línea 16 tiene costo constante. La suma de los tiempos de ejecución del algoritmo 1 es $O(n^8) + O(n^4) + O(n^2) + 4c$. Los ciclos 14 y 15 son las instrucciones más costosas de nuestro algoritmo, por lo tanto, nuestro algoritmo tiene complejidad $O(n^8)$.

4.1.3. Tiempo de ejecución real

A continuación hacemos una comparación entre el tiempo de ejecución del algoritmo y el tiempo de ejecución calculado con base en el tamaño del espacio de búsqueda combinatorio. En la tabla 4.2 mostramos el tiempo de ejecución calculado con base en el tamaño del espacio de búsqueda. El tiempo mostrado en la tercera columna, de la tabla 4.2, corresponde al tiempo de ejecución promedio de las líneas 13 y 14 del algoritmo 1 cuando recibe como

entrada el tipo de orden en posición convexa. Notemos que las líneas 13 y 14 tendrán un tiempo de ejecución mayor cuando verifica a una pareja de $2K_2$ que si se cruza. En la tabla 4.3 mostramos los tiempos de ejecución reales del algoritmo 1. El algoritmo fue ejecutado en una computadora personal con un procesador AMD A10 a 1.9 GHz.

n	Tamaño del espacio de búsqueda	Tiempo promedio de las líneas 13 y 14	Tiempo Total
8	2 088 450	0.000000928868 seg.	1.9398 seg.
9	900 492 390	0.000000716587 seg.	10.7546 min.
10	405 675 657 450	0.000000650148 seg.	3.05265 días

Tabla 4.2: Tiempos de ejecución calculados con base al tamaño del espacio de búsqueda combinatorio de determinar $\#2K_2cf(n) = 2$.

n	Tiempo de ejecución
8	1.7777 segundos
9	5.5131 minutos
10	23.3003 horas

Tabla 4.3: Tiempos de ejecución reales del algoritmo 1

Como podemos observar nuestro algoritmo se ejecutó más rápido de lo calculado. Esto es porque nuestro cálculo fue hecho sobre los conjuntos de 8, 9 y 10 puntos en posición convexa. Sabemos gracias a la ejecución de nuestro algoritmo que el conjunto de puntos en posición convexa maximiza el número de $2K_2$-crossing families (en la siguiente sección mostramos los resultados más detallados). Por lo tanto, para n fija, el número de $2K_2$-crossing families para los demás tipos de orden es menor o igual al número de $2K_2$-crossing families para posición convexa. Por esta razón, el tiempo de ejecución promedio de las lineas 13 y 14 es menor para los tipos de orden con menor cantidad de $2K_2$-crossing families. Esto causa que nuestro algoritmo se ejecute más rápido de lo que calculamos. Notemos que si todos los tipos de orden tuviesen el mismo número de $2K_2$-crossing families entonces los tiempos de ejecución reales de nuestro algoritmo deberían de ser muy cercanos a los tiempos de ejecución calculados.

4.1.4. Resultados del Algoritmo

Como sabemos que $\#2K_2(n) = 2$, entonces contamos el número de $2K_2$-crossing families de tamaño dos para cada tipo de orden para $8 \leq n \leq 10$. Los tipos de orden de la base de datos [1] están ordenados con base a ciertas propiedades. Por esta razón podemos hablar de los tipos de orden con base en su posición en la base de datos. El tipo de orden 1 siempre corresponde al conjunto de n puntos en posición convexa. En la tabla 4.4 mostramos los tipos de orden con mayor y menor número de $2K_2$-crossing families. En las figuras 4.3, 4.4 y 4.5 mostramos los tipos de orden con mayor y menor número de $2K_2$-crossing families para 8, 9 y 10 puntos. La información para los tipos de orden restante puede ser descargada en `https://computacion.cs.cinvestav.mx/~cbulnes/tesis/2k2_crossing_families.zip`.

n	Tipo de orden con mayor número de $2K_2$-crossing families	Número	Tipo de orden con menor número de $2K_2$-crossing families	Número
8	1	241	2 988	103
9	1	2 169	151 152	961
10	1	10 845	13 413 894	4905

Tabla 4.4: Tipos de orden con mayor y menor número de $2K_2$-crossing families para cada valor de n

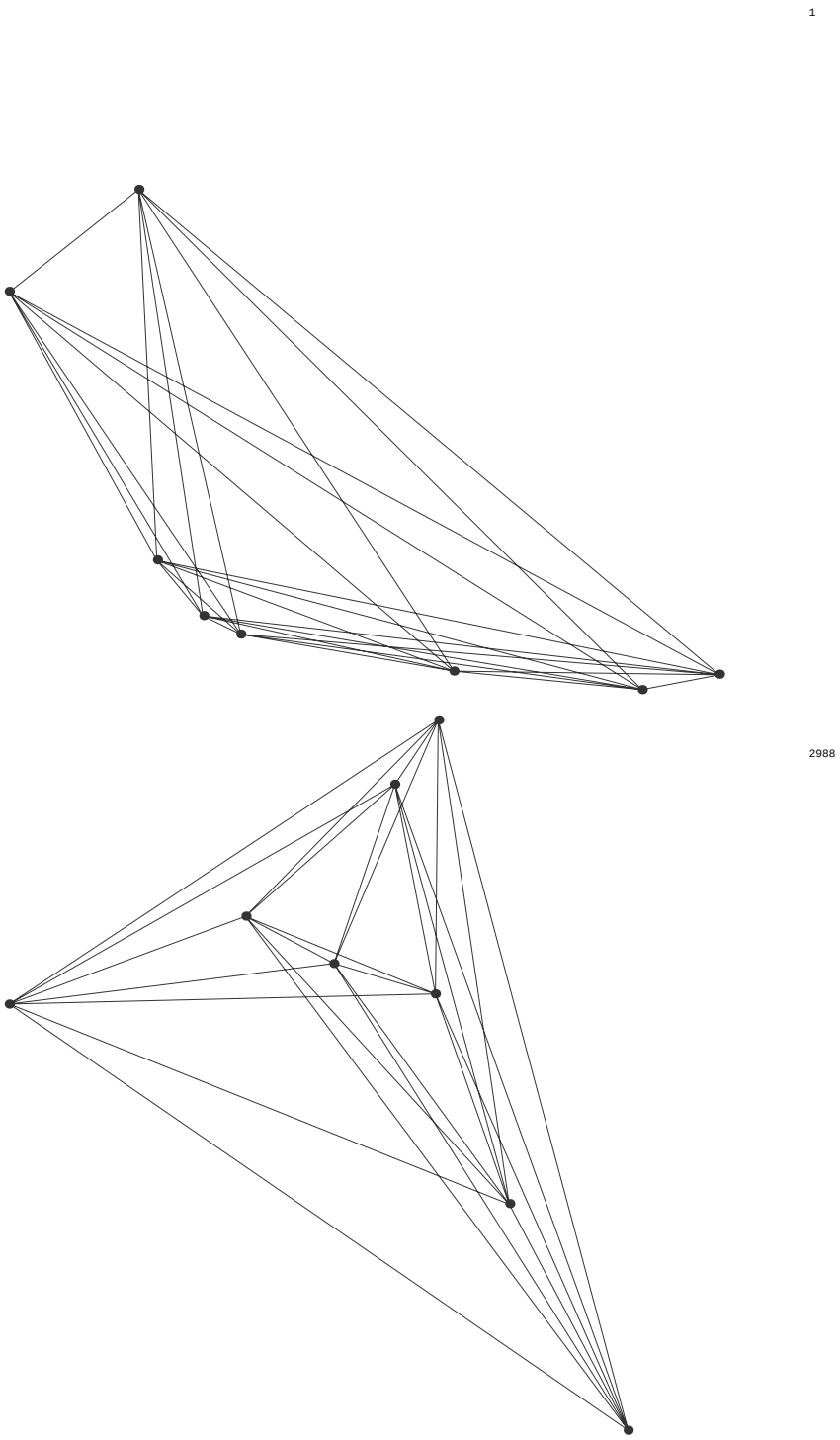

Figura 4.3: Tipos de orden de $n = 8$ con mayor (70) y menor (19) número de $2K_2$-crossing families.

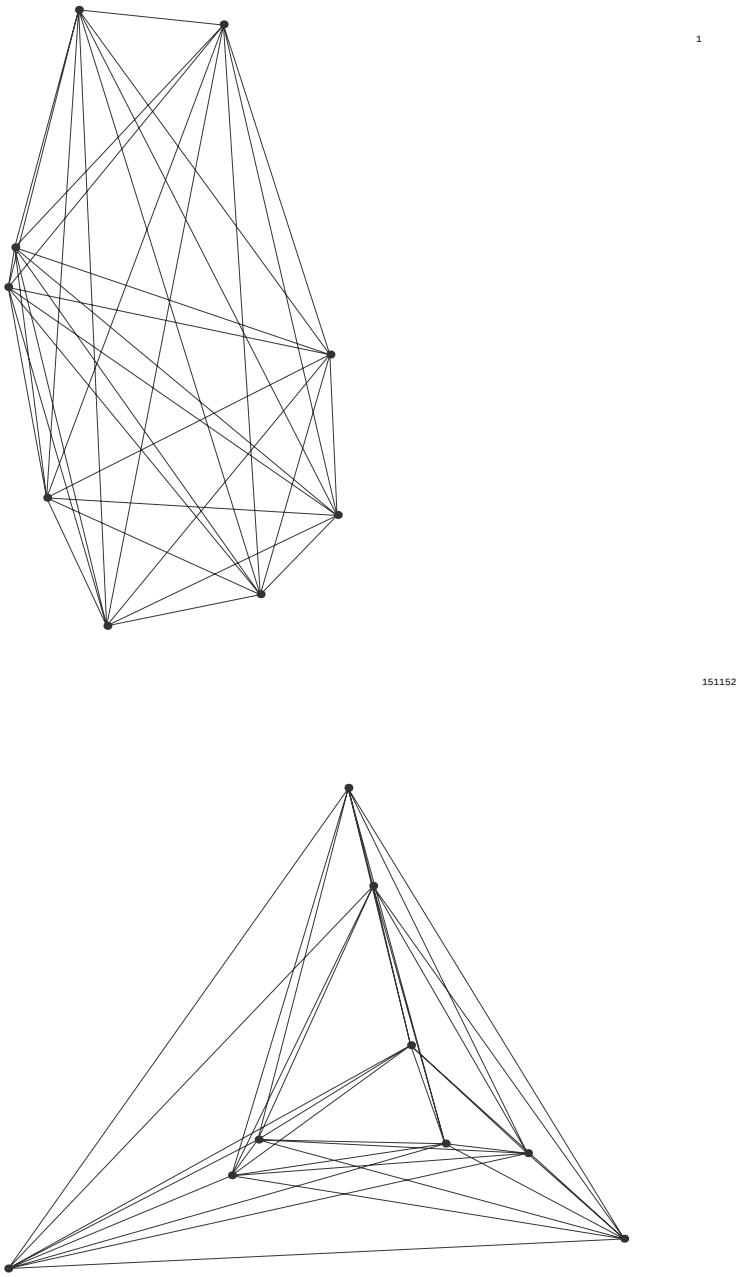

Figura 4.4: Tipos de orden de $n = 9$ con mayor (126) y menor (36) número de $2K_2$-crossing families.

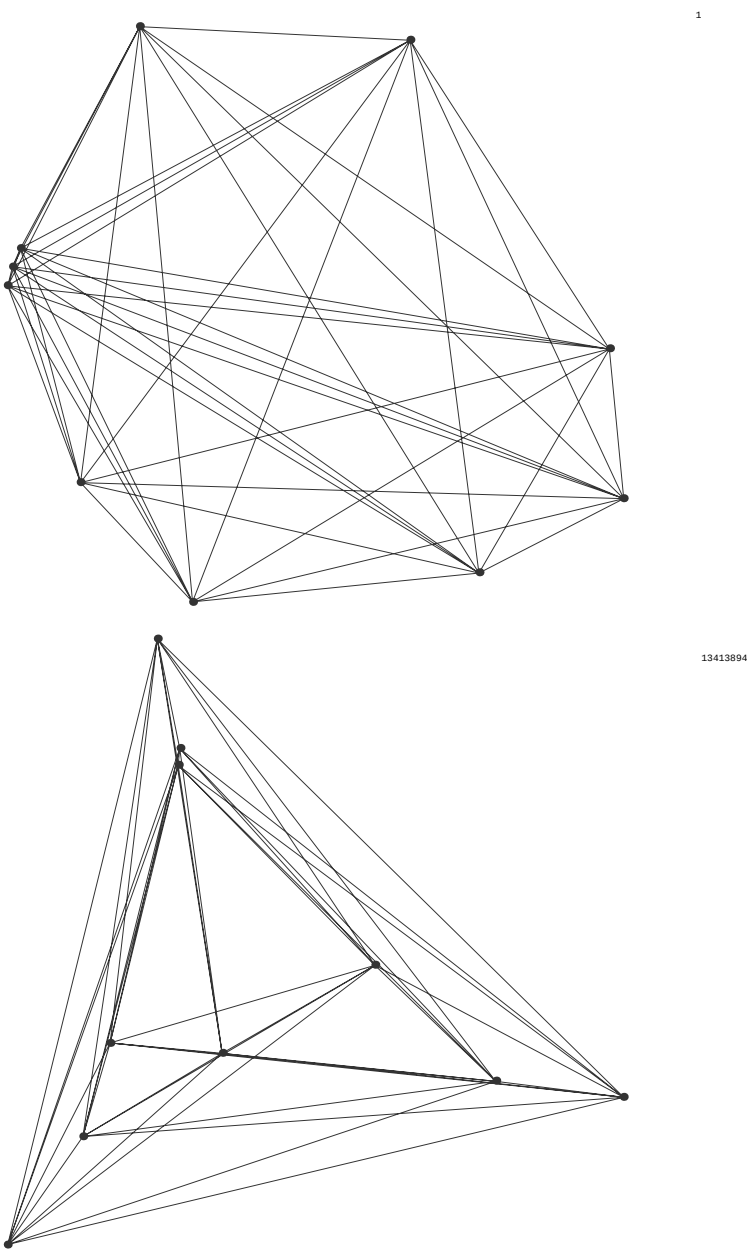

Figura 4.5: Tipos de orden de $n = 10$ con mayor (10845) y menor (4905) número de $2K_2$-crossing families.

Capítulo 5

Crossing Families de $K_{1,3}$

Denotamos como $K_{1,3}$ a la gráfica bipartita completa con 1 y 3 vértices. Llamamos *ápice* al vértice de grado tres en una $K_{1,3}$. En la figura 5.1 mostramos un encaje de una gráfica $K_{1,3}$. Sean H_1 y H_2 dos gráficas geométricas. Decimos que H_1 y H_2 se *cruzan* si existe una arista en H_1 y una arista en H_2 que se cruzan. Llamamos $K_{1,3}$-*crossing family* a un conjunto de gráficas isomorfas a $K_{1,3}$ en el que cualesquiera dos de sus elementos se cruzan. En este capítulo estudiamos el problema de encontrar el número de $K_{1,3}$-crossing family para 8, 9 y 10 puntos. Definimos al número de $K_{1,3}$-crossing family como $\#K_{1,3}cf(n) = max\{\#K_{1,3}cf(S)|S$ es un conjunto de n puntos$\}$, donde $\#K_{1,3}cf(S)$ es la cardinalidad de la $K_{1,3}$-crossing family de S más grande posible. En otras palabras, deseamos determinar la $K_{1,3}$-crossing family de cardinalidad máxima sobre todos los conjuntos de 8, 9 y 10 puntos.

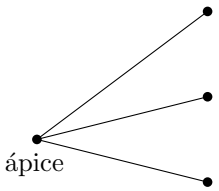

Figura 5.1: Encaje de una gráfica bipartita completa $K_{1,3}$

Dado S un conjunto de n puntos, el siguiente teorema determina el número de parejas de gráficas isomorfas a $K_{1,3}$ de S, fijando únicamente la primera $K_{1,3}$ y uno de los $n-5$ vértices restantes de S.

Teorema 9. *Considérese la gráfica bipartita completa $K_{1,3}$ con bipartición $V = U \cup W$. Sea S un conjunto de n puntos en el plano, sea H una gráfica*

geométrica definida sobre S, que es isomorfa a $K_{1,3}$. Sea $P = S \setminus V(\mathsf{H})$ y sea v_a un punto de P. Sea k la cantidad de segmentos en P que inciden en v_a y que cruzan a H. Sea $f(k)$ el número de gráficas geométricas definidas sobre P isomorfas a $K_{1,3}$ con ápice v_a y que cruzan a H. Entonces:

$$f(k) = \begin{cases} \binom{n-5}{3} \text{ si } k \geq n-7 \\ \sum_{i=1}^{k} \binom{n-5-i}{2} \text{ si } k < n-7 \end{cases}$$

Demostración. La demostración al teorema se divide en dos casos. El primer caso sucede cuando el número de segmentos que inciden a v_a que cruzan H es mayor o igual que $n-7$. El segundo caso sucede cuando el número de segmentos que inciden en v_a que cruzan a H es menor que $n-7$.

Sea K el conjunto de todos los segmentos de la forma (v_a, p), donde $p \in P$, que cruzan a H. Sea $k = |K|$.

Supongamos que $k \geq n-7$. Nótese que hay $n-5$ segmentos de P incidentes en v_a. Si $k \geq n-7$, entonces hay a lo sumo dos segmentos incidentes en v_a que no cruzan a H. Contemos ahora $f(k)$. Cualquier gráfica isomorfa a $K_{1,3}$ tiene exactamente tres aristas. Por lo tanto, cualquier gráfica isomorfa a $K_{1,3}$ con ápice en v_a tiene al menos una arista en K. Por lo tanto, en este caso,

$$f(k) = \binom{n-5}{3}.$$

Supongamos que $k < n-7$. Si hay al menos un segmentos e_1 en K, entonces $f(k)$ es al menos $\binom{n-5-1}{2}$ (número de gráficas con e_1 en su conjunto de aristas). Si hay al menos dos segmentos e_1 y e_2 en K, entonces $f(k)$ es al menos $\binom{n-5-1}{2} + \binom{n-5-2}{2}$. En general, si hay al menos k segmentos en K, entonces:

$$\sum_{i=1}^{k} \binom{n-5-i}{2} \text{ si } k < n-7.$$

□

Para encontrar la $K_{1,3}$-crossing family de cardinalidad máxima realizamos búsquedas sobre la base de datos [1]. En la siguiente sección analizamos el tamaño de búsqueda para el problema de determinar $\#K_{1,3}cf(n)$. Posteriormente presentamos el algoritmo implementado para determinar $\#K_{1,3}cf(n)$.

5.1. Algoritmo para determinar $\#K_{1,3}cf(n)$

5.1.1. Espacio de Búsqueda

El conjunto de vértices de una gráfica $K_{1,3}$ tiene cardinalidad igual a cuatro. Si un conjunto S de n puntos tiene una $K_{1,3}$-crossing family que contiene al menos dos elementos, entonces el número de puntos en el conjunto debe ser $n \geq 8$. Si una $K_{1,3}$-crossing family contiene al menos tres elementos, entonces $n \geq 12$. Como hemos mencionado antes, la base de datos [1] tiene todos los tipos de orden de conjuntos de puntos de tamaño menor o igual a 10. Por lo tanto,

$$0 \leq \#K_{1,3}cf(n \leq 10) \leq 2.$$

El tamaño del espacio de búsqueda para determinar si $\#K_{1,3}cf(n) = k$ está dado por dos factores:

- Las formas distintas de elegir k gráficas geométricas isomorfas a $K_{1,3}$ y,
- la cantidad de tipos de orden que existen para ese valor de n.

A continuación analizamos el tamaño del espacio de búsqueda para encontrar todas las $K_{1,3}$-crossing families de cardinalidad 2. Dado un conjunto de n puntos hay n formas de elegir el ápice de la primera $K_{1,3}$. Para cada ápice hay $\binom{n-1}{3}$ formas de elegir tres puntos que formen una $K_{1,3}$ con dicho ápice. Por lo tanto, hay $n\binom{n-1}{3}$ formas de elegir el primer elemento de la $K_{1,3}$-crossing family. De la misma manera, cada vez que fijemos el primer elemento de la $K_{1,3}$-crossing family hay $(n-4)\binom{n-5}{3}$ formas de definir su segundo elemento. Sin embargo, notemos que cada $K_{1,3}$ es contada dos veces. Por lo tanto, el tamaño de nuestro espacio de búsqueda es $\frac{n\binom{n-1}{3}(n-4)\binom{n-5}{3}}{2}$, esto es $O(n^8)$. En la tabla 5.1 mostramos el tamaño exacto del espacio de búsqueda para cada valor de n entre 8 y 10.

n	Parejas de $K_{1,3}$	Tipos de orden	Tamaño del espacio de búsqueda
8	560	3 315	1 856 400
9	5 040	158 817	800 437 680
10	25 200	14 309 547	360 600 584 400

Tabla 5.1: Tamaño del espacio de búsqueda para determinar $\#K_{1,3}cf(n)$.

5.1.2. Algoritmo

Para determinar $K_{1,3}cf(n)$ implementamos un algoritmo que explora el espacio de búsqueda sin construirlo todo. El algoritmo recibe como entrada un conjunto S de n puntos, correspondientes a un tipo de orden. Para dicho conjunto, el algoritmo construye las $n\binom{n-1}{3}$ gráficas isomorfas a $K_{1,3}$ correspondientes al primer elemento de la $K_{1,3}$-crossing family. Por cada uno de estos, el algoritmo fija cada uno de los $n-4$ vértice restantes para considerarlo como ápice. Por cada vértice, el algoritmo verifica cuántos segmentos que inciden en dicho vértice cruzan a la $K_{1,3}$ fija. Por el teorema 9 sabemos el número de $K_{1,3}$ con dicho ápice que cruzan a la $K_{1,3}$ fija. El algoritmo acumula en la variable *n_cruces* el número de parejas de $K_{1,3}$ que se cruzan cada vez que fija la primera $K_{1,3}$ y uno de los $n-4$ vértices. Ver algoritmo 2.

Nótese que el cálculo del número de parejas de $K_{1,3}$ que se cruzan se encuentra en código duro. Esto debido a que el número de casos es manejable en nuestro algoritmo. Por lo tanto, resultaba más rápido hacer el incremento directo que el cálculo en cada iteración.

A continuación analizamos la complejidad de nuestro algoritmo de búsqueda. La instrucción de la línea 1 tiene un costo constante. Notemos que el algoritmo tiene 6 ciclos anidados que abarcan desde la línea 3 hasta la línea 36. Como todas las instrucciones dentro de estos ciclos son constantes, analizaremos la complejidad del algoritmo resolviendo los ciclos. Por lo tanto el número de veces que se ejecutan los ciclos de las líneas 3 a 36 es:

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=j+1}^{n-1}\sum_{l=k+1}^{n-1}\sum_{m=0}^{n-1}\sum_{p=0}^{n-1}1$$

Esta suma es igual a

$$\frac{n^6}{6}-\frac{n^5}{2}+\frac{n^4}{3}.$$

Esta expresión es asintóticamente dominada por $\frac{n^6}{6}$, por lo tanto la complejidad de los ciclos es de $O(n^6)$. Por último la línea 37 tiene complejidad constante. Por lo tanto, la suma de los tiempos de ejecución del algoritmo 2 es $O(n^6)+2C$, por lo tanto, nuestro algoritmo tiene complejidad $O(n^6)$.

Cabe destacar que la complejidad de nuestro algoritmo es menor que a del espacio de búsqueda del problema.

Algoritmo 2 Encuentra todas las $K_{1,3}$-crossing families

Entrada: Conjunto S de n puntos correspondiente a un tipo de orden.
Salida: Número de $K_{1,3}$-crossing families de cardinalidad máxima para S.

```
1: n_cruces ← 0
2: Declara una estructura llamada K_{1,3}
3: for i ← 0 hasta n do
4:     Selecciona S[i] como ápice
5:     for j ← 0 hasta n do
6:         if S[i] ≠ S[j] then
7:             Selecciona a S[i]S[j] como la primera arista de K_{1,3}.
8:         for k ← j + 1 hasta n do
9:             if S[i] ≠ S[k] then
10:                Selecciona a S[i]S[k] como la segunda arista de K_{1,3}.
11:            for l ← k + 1 hasta n do
12:                if S[i] ≠ S[l] then
13:                    Selecciona a S[i]S[l] como la tercera arista de K_{1,3}.
14:                for m ← 0 hasta n do
15:                    if S[m] ≠ S[i], S[j], S[k], S[l] then
16:                        Selecciona a S[m] como el segundo ápice.
17:                    intersecciones ← 0
18:                    for p ← 0 hasta n do
19:                        if S[p] ≠ S[i], S[j], S[k], S[l], S[m] then
20:                            if S[m]S[p] intersecta a K_{1,3} then
21:                                intersecciones ← intersecciones + 1
22:                    if n = 8 then
23:                        if intersecciones > 0 then
24:                            n_cruces ← n_cruces + 1
25:                    if n = 9 then
26:                        if intersecciones = 1 then
27:                            n_cruces ← n_cruces + 3
28:                        if intersecciones ≥ 2 then
29:                            n_cruces ← n_cruces + 4
30:                    if n = 10 then
31:                        if intersecciones = 1 then
32:                            n_cruces ← n_cruces + 6
33:                        if intersecciones = 2 then
34:                            n_cruces ← n_cruces + 9
35:                        if intersecciones ≥ 3 then
36:                            n_cruces ← n_cruces + 10
37: return n_cruces/2 ▷ Número de K_{1,3}-crossing families de cardinalidad
    máxima de S.
```

5.1.3. Tiempo de ejecución real

A continuación hacemos una comparación entre el tiempo de ejecución del algoritmo y el tiempo de ejecución calculado con base en el tamaño del espacio de búsqueda combinatorio. En la tabla 5.2 mostramos el tiempo de ejecución calculado con base en el tamaño del espacio de búsqueda combinatorio. El tiempo de ejecución mostrado en la tercera columna, de la tabla 5.2, corresponde al tiempo de ejecución de determinar si una pareja de $K_{1,3}$ se cruza. En la tabla 5.3 mostramos los tiempos de ejecución reales del algoritmo 2. El algoritmo fue ejecutado en una computadora personal con un procesador AMD A10 a 1.9 GHz.

n	Tamaño del espacio de búsqueda	Determinar si una pareja de $K_{1,3}$ se cruza	Tiempo Total
8	1 856 400	0.000001s	1.8564 seg.
9	800 437 680	0.000001s	13.340628 min.
10	360 600 584 400	0.000001s	4.1736 días

Tabla 5.2: Tiempos de ejecución calculados con base en el tamaño del espacio de búsqueda combinatorio para determinar $\#K_{1,3}cf(n) = 2$

n	Tiempo de ejecución del algoritmo 2
8	2.146709 segundos
9	4.62 minutos
10	18.31244 horas

Tabla 5.3: Tiempos de ejecución del algoritmo 2

Notemos que nuestro algoritmo se ejecutó notablemente más rápido que lo calculado. Esto gracias a que la complejidad de nuestro algoritmo es menor a la complejidad combinatoria. Mientras que la complejidad combinatoria es $O(n^8)$ la complejidad de nuestro algoritmo es $O(n^6)$.

5.1.4. Resultados del algoritmo

Obtuvimos el número de $K_{1,3}$-crossing families para cada tipo de orden para $8 \leq n \leq 10$. Recordemos que los tipos de orden de la base de datos [1] están ordenados con base en ciertas propiedades. Por esta razón podemos hablar de los tipos de orden con base en su posición en la base de datos. El tipo de

orden 1 siempre corresponde al conjunto de n puntos en posición convexa. En la tabla 5.4 mostramos los tipos de orden con mayor y menor cantidad de $K_{1,3}$-crossing families. En las figuras 5.2, 5.3 y 5.4 mostramos los tipos de orden con mayor y menor número de $K_{1,3}$-crossing families para 8, 9 y 10 puntos. La información para los tipos de orden restante puede ser descargada en `https://computacion.cs.cinvestav.mx/~cbulnes/tesis/k13_crossing_families.zip`.

n	Tipo de orden con mayor cantidad de $K_{1,3}$-crossing families	Número	Tipo de orden con menor cantidad de $K_{1,3}$-crosing families	Número
8	1	496	2991	276
9	1	4464	151148	2522
10	1	22320	13413894	12862

Tabla 5.4: Tipos de orden con mayor y menor cantidad de $K_{1,3}$-crossing families para cada valor de n

Descubrimos que el tipo de orden con menor número de $K_{1,3}$-crossing families para 8 y 9 puntos coincide con el tipo de orden con el menor número de crossing number para 8 y 9 puntos reportados en la base de datos [1]. El *crossing number* de K_n es el número mínimo de cruces entre parejas de segmentos de K_n.

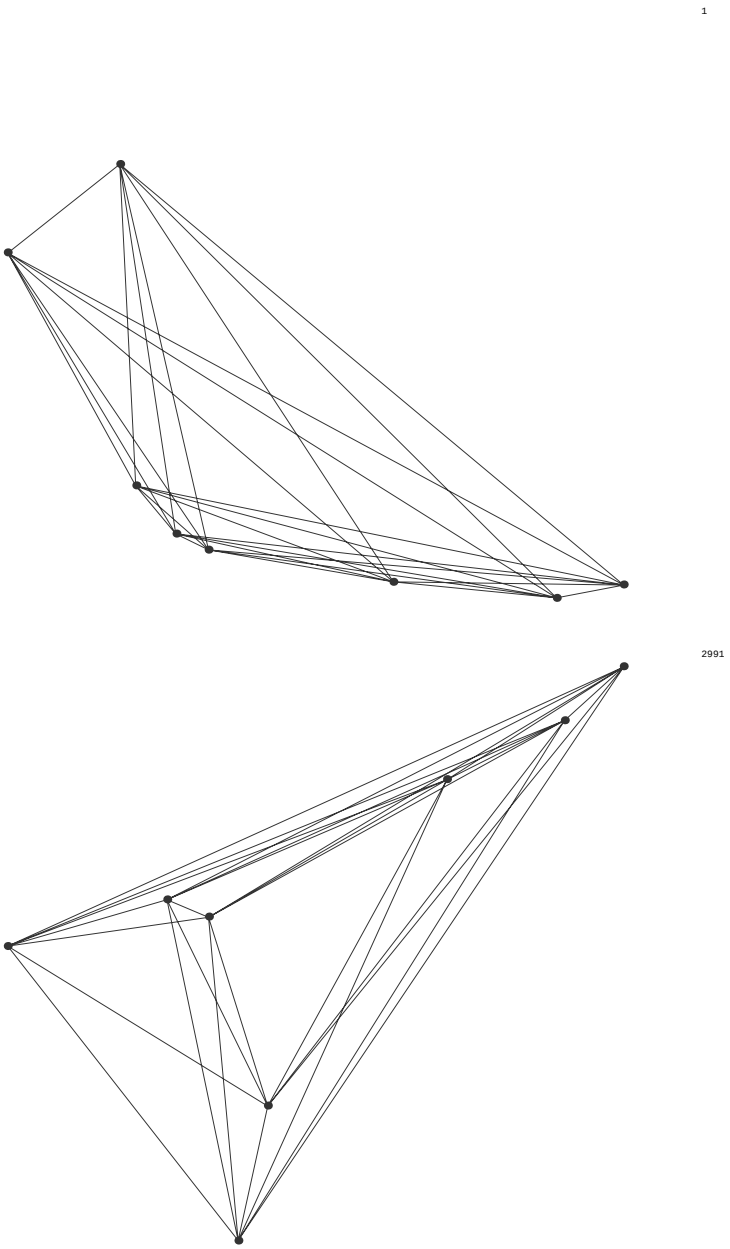

Figura 5.2: Tipos de orden de $n = 8$ con mayor (496) y menor (276) número de $K_{1,3}$-crossing families.

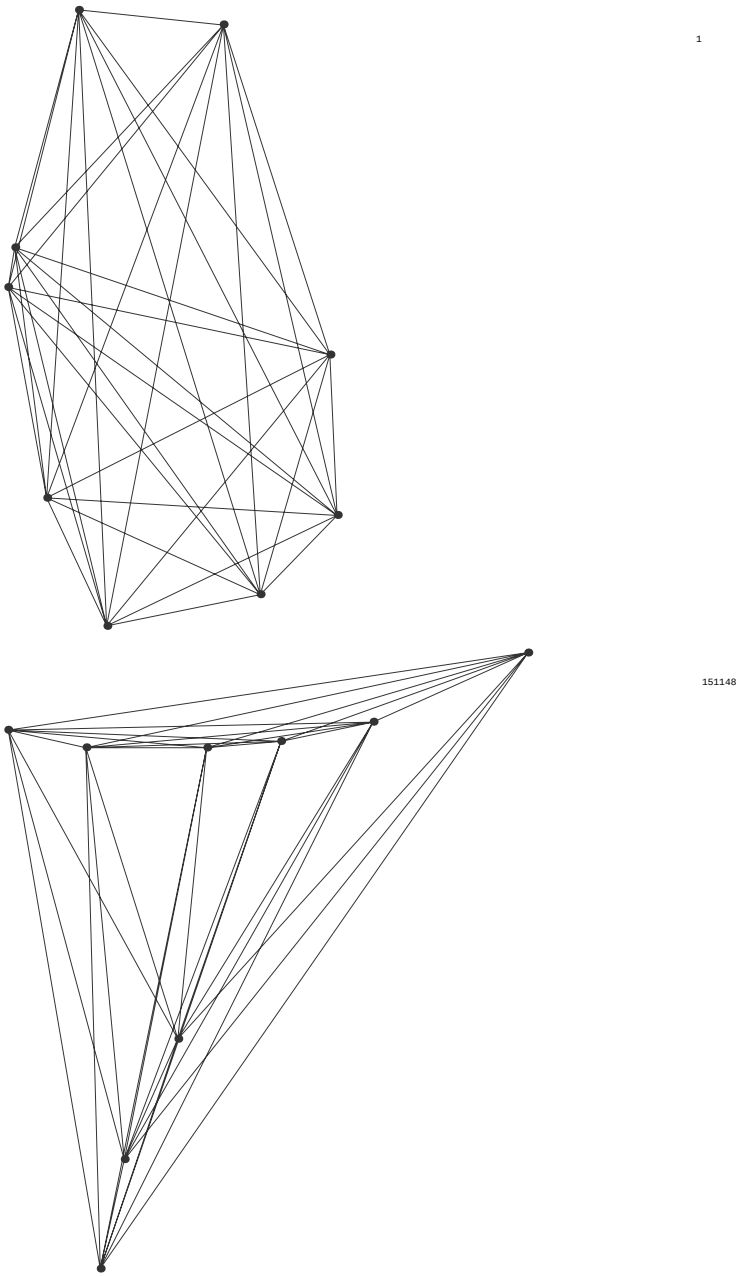

Figura 5.3: Tipos de orden de $n = 9$ con mayor (4464) y menor (2522) número de $K_{1,3}$-crossing families.

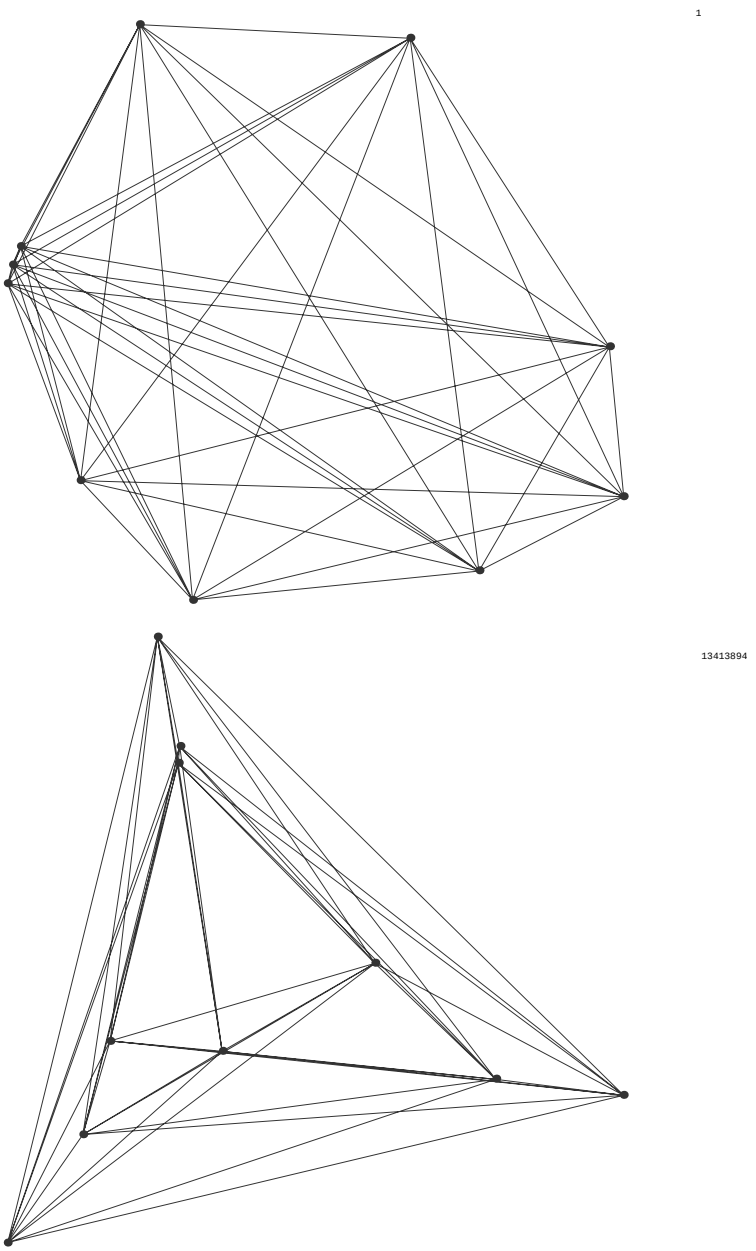

Figura 5.4: Tipos de orden de $n = 10$ con mayor (22320) y menor (12862) número de $K_{1,3}$-crossing families.

Capítulo 6

Thrackles

Sea K_n una gráfica geométrica completa definida sobre un conjunto S de n puntos. Decimos que $\mathsf{T} \subseteq \mathsf{K}_n$ es un *thrackle* de K_n si cualesquiera dos aristas de T se cruzan o se intersectan. Decimos que T es *máximo* si el número de aristas de T es igual al número de vértices de K_n. En la figura 6.1 mostramos un thrackle máximo con 8 vértices.

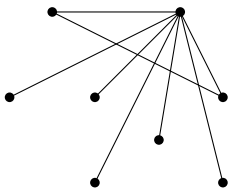

Figura 6.1: Thrackle máximo con 8 vértices.

En esta sección damos un teorema que define la estructura que cumple el conjunto de puntos de cualquier thrackle máximo. Antes es necesario demostrar el siguiente lema.

Lema 4. *Sea S un conjunto de n puntos y sea T un thrackle definido sobre S. T no contiene dos ciclos disjuntos.*

Demostración. Sean C_1 y C_2 dos ciclos de T. Si C_1 no es un thrakle o C_2 no es un thrakle, entonces claramente esto contradice la suposición de que T es un thrakle y la demostración estaría completa.

Supongamos C_1 es un thrakle y C_2 es un thrakle y supongamos por contradicción que T contiene dos ciclos C_1 y C_2.

Para demostrar el teorema probaremos que existe un par de aristas $e_1 \in \mathsf{C}_1$ y $e_2 \in \mathsf{C}_2$ que no se intersectan.

Tomemos una arista $e \in \mathsf{C}_2$ que intersecta a C_1. Esta arista define una línea que divide a los vértices de C_1 en dos semiplanos. Como C_1 es impar entonces hay un semiplano que tiene al menos un vértice más. Llamaremos A al semiplano con el mayor número de vértices y B al semiplano con el menor número de vértices.

Observemos que cualquier vértice $v_i \in C, 1 \leq i \leq k-1$ tiene como vecinos a los vértices v_{i-1}, v_{i+1}, sabemos por la definición del ciclo que v_i es adyacente con v_{i-1} y v_{i+1}, como todos los vértices v_i son diferente, entonces v_i sólo es adyacente con v_{i-1} y v_{i+1}, por lo tanto v_i es de grado 2. Además cómo $v_0 = v_k$ y v_0 es adyacente con v_1 y v_k es adyacente con v_{k-1} entonces v_0 y v_k también son de grado 2.

Como C_1 es isomorfo a C. Los vértices de C_1 son de grado 2. Por lo tanto, podemos dirigir las aristas de C_1 de tal manera que cada uno de sus vértices tenga una arista que llega al vértice y una arista que sale del vértice. Ver figura 6.2.

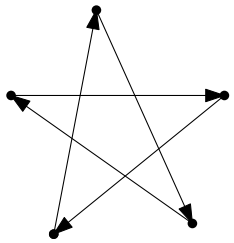

Figura 6.2: Ciclo de 5 vértices dirigido.

Todos los vértices en A tienen una arista que sale a los vértices de B, como en A hay al menos un vértice más que en B, éste vértice tiene una arista que sale a otro vértice de A, ésta arista está totalmente contenida en el semiplano A entonces no puede intersectar a e, lo cuál contradice la definición de un thrackle.

□

Sea r la recta defida por dos vértices u, v. Un *vértice simétrico* v' con respecto a u es un vértice sobre r tal que la distancia del vértice v' al vértice u es igual a la distancia del vértice v al vértice u.

Sea $\mathsf{C}(\mathsf{T})$ un ciclo de T y sea p un vértice de $\mathsf{C}(\mathsf{T})$, definimos una *cuña* $W_T(p)$ como la región acotada por p y sus dos vértices vecinos en $\mathsf{C}(\mathsf{T})$, el vértice p es llamado *ápice* de $W_T(p)$ [12]. Sean o y q los vértices vecinos de p en $\mathsf{C}(\mathsf{T})$ y sea o' el vértice simétrico de o y q' el vértice simétrico de q. Definimos una cuña inversa $W'_T(p)$ como la región acotada por p con los

vértices simétricos o' y q'. En la figura 6.3 mostramos un ejemplo de una cuña inversa.

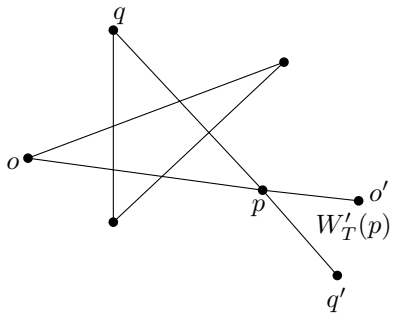

Figura 6.3: Ejemplo de una cuña inversa de un ciclo $\mathsf{C}(\mathsf{T})$

Teorema 10. *Sea* T *un thrackle máximo y sea* T' *el conjunto de vértices de* T *que no pertenecen a* $\mathsf{C}(\mathsf{T})$*, es decir,* $\mathsf{T}' = \mathsf{T} \setminus \mathsf{C}(\mathsf{T})$*. Todos los vértices de* T' *habitan en alguna cuña* $W_T(p)$*.*

Demostración. Por el *teorema 20* de [12] sabemos que todo thrackle máximo tiene un ciclo impar. Por lo tanto T tiene un ciclo impar y T' existe.

Supongamos por contradicción que algún vértice $v \in \mathsf{T}'$ no está en ninguna cuña de $\mathsf{C}(\mathsf{T})$, v necesariamente debe estar en alguna cuña inversa. Notemos que tenemos cinco posibles casos, formar aristas con los vértices de la cuña ($\overline{vo}$, $\overline{vq}$ y $\overline{vp}$), formar una arista con algún vértice a la izquierda de $\vec{po}$ y formar una arista con algún vértice a la derecha de $\vec{pq}$. La arista $\overline{vo}$ no cruza a la arista $\overline{pq}$, de la misma manera cualquier arista formada con v y algún vértice a la izquierda de $\vec{po}$ no cruza a $\overline{pq}$. Ver figura 6.4A. La arista $\overline{vq}$ no cruza a la arista $\overline{po}$, así como cualquier arista formada con v y algún vértice a la derecha de $\vec{pq}$ no cruza a $\overline{po}$. Ver figura 6.4B. Estos 4 casos contradicen la definición de un thrackle. Observemos que solo es posible formar un thrackle con la arista $\overline{vp}$, pues comparte vértice con las aristas $\overline{po}$ y $\overline{pq}$. Por lo tanto, la arista $\overline{vp}$ cumple con la definición de un thrackle, sin embargo no existe ninguna arista de $\mathsf{C}(\mathsf{T})$ que pase por $W'_T(p)$, por lo tanto la arista $\overline{vp}$ no cruza ninguna arista de $\mathsf{C}(\mathsf{T})$. Ver figura 6.4C. Esto contradice la definición de un thrackle, por lo tanto v no puede habitar en una cuña inversa, lo que nos lleva a una contradicción.

□

Ya sabemos que todo thrackle máximo tiene un ciclo impar. Además, por el teorema 10 sabemos la forma del conjunto de puntos de un thrackle

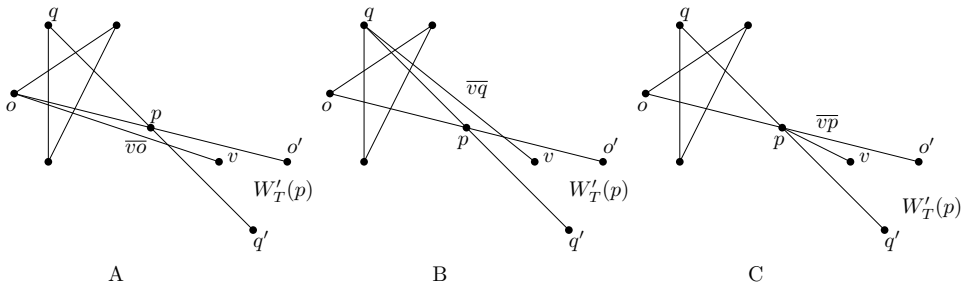

Figura 6.4: Aristas posibles con un vértice en la cuña inversa y los vértices de la cuña

máximo. En el lema 5 caracterizamos a las aristas de un ciclo impar que es un thrackle máximo. En el teorema 11 damos una forma de contar el número de thrackles máximos para conjuntos de puntos en posición convexa.

Lema 5. *Sea P un conjunto de n puntos en el plano, con n impar. P tiene exactamente un ciclo C que ocupa todos los vértices y que es un thrackle máximo.*

Demostración. Sea e una arista de C. Sea A el conjunto de puntos con cardinalidad mayor que están a un lado e. Sea B el conjunto de puntos con cardinalidad menor que están a un lado e. Como $|C|$ es impar A y B siempre existen. Observemos que en cada vértice extremo de e incide una arista que tiene su otro extremo en A (dichas aristas no pueden tener su vértice extremo en B puesto que B tiene menos vértices). Además, como C es un ciclo que es un thrackle, en cada vértice de B inciden dos aristas que tienen su otro extremo en A. Por lo tanto, el número de aristas de C es igual a $2|B| + 3$. Como C tiene n aristas, la cardinalidad de B es igual a $\frac{n-3}{2}$. Observemos que las únicas aristas que cumplen que $|B| = \frac{n-3}{2}$ son las aristas de distancia máxima. Por lo tanto, C está formado por aristas de distancia máxima. Como $|C|$ es impar hay n aristas de distancia máxima. Como solo hay una forma de elegir n aristas de distancia máxima, entonces solo existe un ciclo que es un thrackle máximo formado por aristas de distancia máxima. Por lo tanto, P tiene exactamente un ciclo que ocupa todos los vértices y que es un thrackle máximo.

□

Teorema 11. *Sea S un conjunto de n puntos en posición convexa. El núme-*

ro de thrackles máximos de S es igual

$$\sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} \binom{n}{2i+1}.$$

Demostración. Sea C_i un ciclo de S de tamaño impar que es un thrackle. Sea S' el conjunto de vértices de S que no pertenecen a C_i, es decir, $S' = S \setminus V(C_i)$. Sea c un vértice de C_i y sea $W(c)$ una cuña definida por c y sus dos vértices vecinos en C_i. Sea $W'(c)$ una cuña inversa definida por c. Sea $P \subseteq S$ de cardinalidad impar. Para demostrar el teorema probamos que para cualquier subconjunto P existe exactamente un ciclo impar que es un thrackle. Además, para cada uno de estos ciclos, los vértices de S' están en cuñas.

Por el lema 5 sabemos que P tiene exactamente un ciclo que ocupa todos sus vértices y que es un thrackle máximo. Por lo tanto el número de ciclos impares que son un thrackle máximo de S es igual al número de subconjuntos P de S, esto es

$$\binom{n}{3} + \binom{n}{5} + ... + \binom{n}{n} \text{ si } n \text{ es impar}$$
$$\binom{n}{3} + \binom{n}{5} + ... + \binom{n}{n-1} \text{ si } n \text{ es par}$$

Podemos considerar ambos casos por medio de la suma

$$\sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} \binom{n}{2i+1}.$$

Ahora solo queda demostrar que para cada ciclo impar que es un thrackle, C_i, de S, los vértices de S' están en cuñas. Supongamos por contradicción que un vértice $v \in S'$ no está en ninguna cuña $W(c)$ de C_k. Entonces v necesariamente está en alguna cuña inversa $W'(c)$ de C_i. Notemos que si v esta en alguna cuña inversa $W'(c)$, entonces el conjunto de vértices $\{v \cup V(W'(c))\}$ no están en posición convexa. Ver figura 6.5. Lo cual contradice que S está en posición convexa. Por lo tanto, los vértices de S' necesariamente están en una cuña de C_i. Lo cual demuestra el teorema.

□

Para encontrar los thrackles máximos realizamos búsquedas sobre la base de datos [1] proporcionada por los autores de [3]. En la siguiente sección analizamos el tamaño del espacio de búsqueda para el problema de encontrar los

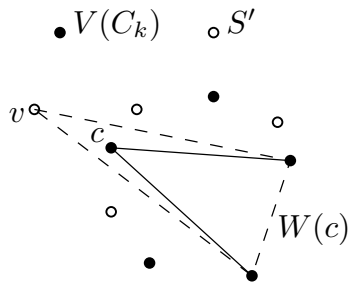

Figura 6.5: El conjunto de puntos $\{v \cup V(W'(c))\}$ no está en posición convexa.

thrackles máximos para un conjunto de puntos. Posteriormente presentamos el algoritmo implementado.

6.1. Algoritmo para determinar thrackles máximos

6.1.1. Espacio de Búsqueda

Dados n puntos en el plano, hay $\binom{n}{2}$ segmentos que conforman a K_n. Como estamos buscando conjunto de n segmentos, entonces hay $\binom{\binom{n}{2}}{n}$ formas de elegir n segmentos de K_n. Por lo tanto, el tamaño del espacio de búsqueda para el problema de encontrar los thrackles de tamaño máximo para un conjunto de puntos es

$$\binom{\binom{n}{2}}{n}.$$

En la tabla 6.1 mostramos el tamaño del espacio de búsqueda para cada valor de n entre 8 y 10.

n	Formas de elegir n segmentos	Polinomio equivalente al binomial	Orden asintótico
8	$\binom{\frac{n(n-1)}{2}}{8}$	$\frac{\prod_{k=0}^{7}(n^2-n-2k)}{2*8!}$	$O(n^{16})$
9	$\binom{\frac{n(n-1)}{2}}{9}$	$\frac{\prod_{k=0}^{8}(n^2-n-2k)}{2*9!}$	$O(n^{18})$
10	$\binom{\frac{n(n-1)}{2}}{10}$	$\frac{\prod_{k=0}^{9}(n^2-n-2k)}{2*10!}$	$O(n^{20})$

Tabla 6.1: Tamaño del espacio de búsqueda para encontrar thrackles máximos sobre conjuntos de 8, 9 y 10 puntos.

En la tabla 6.2 mostramos el espacio de búsqueda exacto considerando los tipos de orden para cada valor de n utilizado.

n	Formas de elegir n segmentos	Tipos de orden	Espacio de búsqueda
8	3 108 105	3 315	10 303 368 075
9	94 143 280	158 817	$1{,}49515533 * 10^{13}$
10	3 190 187 286	14 309 547	$4{,}565013491 * 10^{16}$

Tabla 6.2: Tamaño exacto del espacio de búsqueda para encontrar thrackles máximos sobre todos los tipos de orden de conjuntos de 8, 9 y 10 puntos.

6.1.2. Algoritmo para determinar los thrackles de tamaño máximo para un conjunto de puntos

Debido a que la complejidad del tamaño del espacio de búsqueda es muy alta no implementamos un algoritmo exhaustivo. En cambio, nuestro algoritmo descarta todos los subconjuntos de segmentos que sabemos que no formarán un thrackle. A continuación describimos nuestro algoritmo. Dado un conjunto S de n puntos correspondientes a un tipo de orden, el algoritmo construye un arreglo con los $\binom{n}{2}$ segmentos de S. Con dicho arreglo, el algoritmo explora las $\binom{\binom{n}{2}}{n}$ formas de elegir n puntos (como si fuese el algoritmo exhaustivo). El algoritmo verifica si el segmento que se está agregando para formar parte de un thrackle máximo, intersecta a los ya agregados. De no ser así, el algoritmo descarta todos los subconjuntos que incluyan al segmento que no intersecta a los demás. En el mejor de los casos, nuestro algoritmo descarta subconjunto desde que agrega el segundo segmento.

El algoritmo varía para cada valor de n en cuanto al número de ciclos anidados. Por esta razón presentamos el algoritmo 3, el cual busca thrackles máximos sobre conjuntos de 8 puntos.

A continuación analizamos la complejidad de nuestro algoritmo de búsqueda. Véase Algoritmo 3. La línea 1 tiene un costo de $O(n^2)$, puesto que construye $\binom{n}{2}$ segmentos. La línea 2 tiene costo constante. Los ciclos de las líneas 3 y 26 tienen como límite al tamaño de E, observemos que $|E| = \binom{n}{2}$, por lo tanto el número de veces que se ejecutan los ciclos de las líneas 3 a 26 es:

$$\sum_{i_1=0}^{\binom{n}{2}} \sum_{i_2=i_1+1}^{\binom{n}{2}} \sum_{i_3=i_2+1}^{\binom{n}{2}} \sum_{i_4=i_3+1}^{\binom{n}{2}} \sum_{i_5=i_4+1}^{\binom{n}{2}} \sum_{i_6=i_5+1}^{\binom{n}{2}} \sum_{i_7=i_6+1}^{\binom{n}{2}} \sum_{i_8=i7+1}^{\binom{n}{2}} 1.$$

Algoritmo 3 Encontrar los thrackles máximos de tamaño 8 para un tipo de orden

Entrada: Conjunto S de n puntos correspondiente a un tipo de orden.
Salida: Número de thrackles máximos para S.

```
1: Construir un arreglo E con los (n 2) segmentos posibles.
2: n_cruces ← 0
3: for i1 ← 0 hasta |E| do
4:     for i2 ← i1 + 1 hasta |E| do
5:         if S[i2] y S[i1] no se intersectan then
6:             Salta el ciclo i2 a su siguiente iteración.
7:         for i3 ← i2 + 1 hasta |E| do
8:             if E[i3] no intersecta a E[i1] o E[i2] then
9:                 Salta el ciclo i3 a su siguiente iteración.
10:            for i4 ← i3 + 1 hasta |E| do
11:                if E[i4] no intersecta a E[i1], E[i2] o E[i3] then
12:                    Salta el ciclo i4 a su siguiente iteración.
13:                for i5 ← i4 + 1 hasta |E| do
14:                    if E[i5] no intersecta a E[i1], E[i2], E[i3] o E[i4] then
15:                        Salta el ciclo i5 a su siguiente iteración.
16:                    for i6 ← i5 + 1 hasta |E| do
17:                        if E[i6] no intersecta a E[i1], E[i2], E[i3], E[i4] o
   E[i5] then
18:                            Salta el ciclo i6 a su siguiente iteración.
19:                        for i7 ← i6 + 1 hasta |E| do
20:                            if E[i7] no intersecta a E[i1], E[i2], E[i3], E[i4],
   E[i5] o E[i6] then
21:                                Salta el ciclo i7 a su siguiente iteración.
22:                            for i8 ← i7 + 1 hasta |E| do
23:                                if E[i8] no intersecta a E[i1], E[i2], E[i3],
   E[i4], E[i5], E[i6] o E[i7] then
24:                                    Salta el ciclo i8 a su siguiente iteración.
25:                                else
26:                                    n_cruces ← n_cruces + 1
27: return n_cruces
```

Esta suma es igual a

$$\frac{n^{16}}{10321920}-\frac{n^{15}}{1290240}-\frac{n^{14}}{860160}+\frac{n^{13}}{46080}-\frac{11n^{12}}{737280}-\frac{7n^{11}}{30720}+\frac{121n^{10}}{368640}+\frac{341n^{9}}{322560}$$

$$-\frac{1441n^{8}}{688128}-\frac{1969n^{7}}{1290240}+\frac{211n^{6}}{36864}-\frac{29n^{5}}{7680}-\frac{4127n^{4}}{645120}+\frac{1079n^{3}}{80640}+\frac{11n^{2}}{4480}-\frac{n}{112}$$

Esta expresión está asintóticamente dominada por $\frac{n^{16}}{10321920}$, por lo tanto la complejidad de los ciclos de las líneas 3 a 26 es de $O(n^{16})$. Por último, la línea 27 tiene costo constante. La suma de los tiempos de ejecución del algoritmo 3 es $O(n^{16}) + O(n^2) + 2c$. Los ciclos de las líneas 3 a 26 son las instrucciones más costosas de nuestro algoritmo, por lo tanto, en el peor de los casos nuestro algoritmo tiene complejidad $O(n^{16})$.

Recordemos que en realidad nuestro algoritmo no explora todo el espacio de búsqueda puesto que descarta todos aquellos conjuntos que no forman un thrackle. Descartamos estos conjuntos gracias a las condiciones de las líneas 5, 8, 11, 14, 17, 20 y 23. Por lo tanto el análisis de complejidad anterior sucede cuando las condiciones de las líneas 5, 8, 11, 14, 17, 20 y 23 nunca se cumplen. El mejor de los casos de nuestro algoritmo sucede cuando la condición de la línea 5 siempre se cumple. De ser así, entones nuestro algoritmo nunca ejecutará los ciclos de las líneas 7 a 26. Por lo tanto el número de veces que se ejecutan los ciclos de las líneas 3 a 26 si la condición de la línea 5 siempre se cumple es:

$$\sum_{i_1=0}^{\binom{n}{2}} \sum_{i_2=i_1+1}^{\binom{n}{2}} 1.$$

Esta suma es igual a

$$\frac{n^4}{8}-\frac{n^3}{4}-\frac{3n^2}{8}-\frac{n}{4}.$$

Esta expresión está asintóticamente dominada por $\frac{n^4}{8}$. Por lo tanto, en el mejor de los casos nuestro algoritmo tiene complejidad $\omega(n^4)$.

6.1.3. Tiempos de ejecución

A continuación hacemos una comparación entre el tiempo de ejecución del algoritmo y el tiempo de ejecución calculado con base en el tamaño del espacio de búsqueda combinatorio. En la tabla 6.3 mostramos el tiempo de

ejecución calculado con base en el tamaño del espacio de búsqueda combinatorio. El tiempo de ejecución mostrado en la tercera columna, de la tabla 6.3, corresponde al tiempo de ejecución promedio de determinar si un conjunto de n aristas es un thrackle. En la tabla 6.4 mostramos los tiempos de ejecución reales del algoritmo 3. El algoritmo fue ejecutado en una computadora personal con un procesador AMD A10 a 1.9 GHz.

n	Tamaño del espacio de búsqueda	Verificar si n aristas son un thrakle	Tiempo Total
8	10 303 368 075	0.000002 seg.	5 horas
9	14 951 553 299 760	0.000002 seg.	11 meses
10	45 650 134 907 819 440	0.000003 seg.	43 siglos

Tabla 6.3: Tiempos de ejecución calculados con base en el tamaño del espacio de búsqueda combinatorio para encontrar thrackles máximos

n	Tiempo de ejecución
8	8.974019 segundos
9	25.94 minutos
10	5.64 días

Tabla 6.4: Tiempos de ejecución reales del algoritmo 3

Como podemos observar nuestro algoritmo se ejecutó notablemente rápido (comparado con el tiempo de ejecución calculado con base en el tamaño del espacio de búsqueda). Esto sucedió gracias a la capacidad de nuestro algoritmo de descartar subconjuntos que no son thrackles. En la siguiente sección mostramos de forma más detallada los factores que causaron que nuestro algoritmo terminara así como otros resultados.

6.1.4. Resultados del algoritmo

Encontramos que, dado $n \leq 10$ no siempre existe un thrackle máximo (un conjunto de n segmentos que se intersectan dos a dos). En la tabla 6.5 mostramos la cantidad exacta de tipos de orden que no tienen thrackle máximo para 8, 9 y 10 puntos. Recordemos que los tipos de orden de la base de datos [1] están ordenados con base en ciertas propiedades. Por esta razón podemos hablar de los tipos de orden respecto a su posición en la base de datos. El tipo de orden 1 siempre corresponde al conjunto de n puntos en posición convexa.

n	Total de tipos de orden	Número de tipos de orden sin thrakle máximo	Porcentaje
8	3 315	1 175	35,44 %
9	158 817	53 758	33,84 %
10	14 309 547	4 654 339	32,52 %

Tabla 6.5: Número y porcentaje de tipos de orden sin thrackle máximo para 8, 9 y 10 puntos

Descubrimos que el tipo de orden que maximiza la cantidad de thrackles máximos es el de posición convexa con 120, 143 y 190 thrackles máximos para 8, 9 y 10 puntos respectivamente. La información para los tipos de orden restante puede ser descargada en `https://computacion.cs.cinvestav.mx/~cbulnes/tesis/thrackles.zip`.

Otro resultado importante es que nuestro algoritmo terminó su ejecución (pese a que el cálculo con base en el tamaño del espacio de búsqueda con 10 puntos indicaba que le tomaría 43 siglos ejecutarse). Recordemos que nuestro algoritmo descarta todos los subconjuntos que no formarán un thrackle. Debido a que solo un porcentaje muy pequeño del espacio de búsqueda es un thrackle máximo nuestro algoritmo logró terminar de ejecutarse. En la tabla 6.6 mostramos el porcentaje del espacio de búsqueda que si es un thrackle máximo para posición convexa. Estos porcentajes son la cantidad de veces que nuestro algoritmo ejecuta los n ciclos. En los demás tipos de oren el número de veces que el algoritmo ejecuta los n ciclos es menor o igual a los presentados en dicha tabla.

n	Número de conjuntos de n segmentos	Número de conjuntos que son un thrakle máximo	Porcentaje
8	3 108 105	120	0,00386 %
9	94 143 280	247	0,00026 %
10	3 190 187 286	502	0,000015736 %

Tabla 6.6: Tamaño del espacio de búsqueda que si es un thrackle máximo para posición convexa

Finalmente, desarrollamos un programa en OpenGL el cual permite visualizar todos los thrackles máximos para cada tipo de orden para 8, 9 y 10 puntos. Dicho programa se encuentra disponible en `https://github.com/carlosbulnes/busquedas_de_crossing_families`.

Capítulo 7

Discusión y Conclusiones

En esta tesis estudiamos el problema de encontrar la $2K_2$-crossing family y la $K_{1,3}$-crossing family más grande de tal manera que exista al menos una $2K_2$-crossing family y una $K_{1,3}$-crossing family para cada conjunto de 8, 9 y 10 puntos. También buscamos determinar si todos los conjunto de 8, 9 y 10 puntos tienen un thrackle máximo. Para resolver dichos problemas realizamos búsquedas computacionales sobre los tipos de orden almacenados en la base de datos [1].

En este trabajo logramos encontrar H-crossing families de tamaño máximo, es decir, H-crossing families de tamaño $\lfloor n/|H| \rfloor$, para valores pequeños de n. Como mencionamos anteriormente, la primera vez que se planteó el problema de las crossing families fue en [7]. El problema de encontrar una crossing family de aristas de tamaño máximo ($\lfloor n/2 \rfloor$) para cualquier valor de n aún se encuentra abierto. Sin embargo, en este trabajo hemos mostrado que es posible encontrar H-crossing families de tamaño máximo cambiando la gráfica, es decir, cambiando H. Como las gráficas que estudiamos en esta tesis, $2K_2$ y $K_{1,3}$, tienen cardinalidad igual a cuatro, las H-crossing families que encontramos en esta tesis son de tamaño $\lfloor n/4 \rfloor$.

También logramos determinar que no todos los conjuntos de n puntos tienen un thrackle máximo. De hecho encontramos que aproximadamente el 70 % de los conjuntos de 8, 9 y 10 puntos tienen un thrackle máximo. En los conjuntos de puntos restantes no es posible encontrar thrackles máximos. Por lo tanto, podemos concluir que para conjuntos de 8, 9 y 10 puntos no siempre existe un thrackle máximo. Es decir, dado un valor de n, entre 8 y 10, no es posible encontrar un thrackle máximo para todos los conjuntos de n puntos.

Uno de los resultados de esta tesis es que el conjunto de puntos en posi-

ción convexa siempre tiene el mayor número de $2K_2$-crossing families, $K_{1,3}$-crossing families y thrackles máximos. Por la base de datos [1] sabemos que para 3 a 9 puntos, el conjunto que tiene el mayor número de cruces entre todas sus posibles aristas es el de posición convexa. Es decir, los conjuntos de 3 a 9 puntos en posición convexa maximizan el crossing number. Como en esta tesis estudiamos gráficas geométricas que se cruzan, parece lógico pensar que los resultados serán similares a los del problema de determinar el crossing number. Entonces una pregunta importante es ¿por qué el conjunto en posición convexa también maximiza el número de $2K_2$-crossing families, $K_{1,3}$-crossing families y thrackles máximos?.

Notemos que el problema del crossing number y los problemas estudiados en esta tesis tienen algo en común, todos estudian cruces de estructuras geométricas. En el crossing number se estudia el número de cruces de segmentos en una gráfica geométrica completa con n puntos. Que el conjunto en posición convexa tenga mayor crossing number implica que el número de segmentos que se cruzan en dicho conjunto sea mayor. Este fenómeno se extiende a las gráficas $2K_2$ y $K_{1,3}$, ya que éstas se componen de segmentos. Como el cruce de dos $2K_2$ o dos $K_{1,3}$ ocurre si uno o más de sus segmentos se cruzan, entonces todas las gráficas que contengan dos segmentos que se cruzan en el problema del crossing number también se cruzan en los problemas que estudiamos. En los thrackles la relación es más clara, puesto en que dicho problema se consideran todos los segmentos que se cruzan agregando también los segmentos que comparten puntos. Por estas razones creemos que el comportamiento del problema del crossing number se relaciona con otros problemas que estudian cruces entre otras estructuras geométricas.

Recordemos que cuando estudiamos las $K_{1,3}$-crossing families encontramos que, en efecto, los tipos de orden que minimizan el número de $K_{1,3}$-crossing families, para 8 y 9 puntos, coinciden con los tipos de orden que minimizan el crossing number (para 8 y 9 puntos) reportados en la base de datos [1]. Sin embargo, cuando estudiamos las $2K_2$-crossing families esto no sucedió. El tipo de orden para 8 puntos que minimiza el número de $2K_2$-crossing families no es el mismo que minimiza el crossing number. Sin embargo, se encuentra cerca en el orden de la base de datos del tipo de orden que minimiza el crossing number. Creemos que el orden de los tipos de orden en la base datos guarda cierta relación con la estructura que tienen los conjuntos de puntos. En particular, creemos que se relaciona con el número de puntos en cierre convexo. En la figura 7.1 mostramos el tipo de orden que minimiza el número de $2K_2$-crossing families junto con el tipo de orden que minimiza el crossing number. Como podemos ver en dicha figura, los tipos de orden para ambos problemas tienen 3 puntos en su cierre convexo,

además, si nos fijamos en los puntos restantes, se forma otro cierre convexo con 3 puntos.

Para 9 puntos encontramos que hay 10 tipos de orden que minimizan el crossing number (con 36 cruces). Encontramos que, aunque el tipo de orden que minimiza el número de $2K_2$-crossing families y el tipo de orden que minimiza el número de $K_{1,3}$-crossing families es distinto, ambos tipos de orden pertenecen a los 10 que minimizan el crossing number. En la figura 7.2 mostramos los dos tipos de orden que minimizan dichas crossing families.

Además, con este trabajo se da a conocer más información acerca de estructuras geométricas que se cruzan. Siendo nuestro trabajo una variante de los problemas estudiados por los autores de [10] y por los autores de [11]. Estos trabajos a su vez son una variante del problema estudiado por los autores de [7]. Además, obtuvimos que algunos de nuestros resultados coinciden con los resultados del crossing number. Por lo tanto, puede ser que algunos problemas tanto de la literatura como trabajos futuros puedan ser estudiados considerando su relación con el crossing number.

Finalmente, como pudimos ver en los tiempos de ejecución de nuestros algoritmos, los problemas estudiados en esta tesis requieren de uno o varios días para terminar su ejecución. Esto nos da una idea de la complejidad de resolver dichos problemas manualmente así como de la importancia del uso de computadoras para resolver problemas matemáticos con grandes tamaños de espacio de búsqueda.

Como trabajo futuro proponemos diseñar un mejor algoritmo para la búsqueda de $2K_2$-crossing families. También proponemos estudiar de manera más profunda la relación que existe entre los conjuntos que minimizan nuestros problemas con los que minimizan el problema del crossing number, así como dar características de dichos conjuntos. También puede ser interesante saber como se comportan estos problemas para conjuntos de 11 puntos.

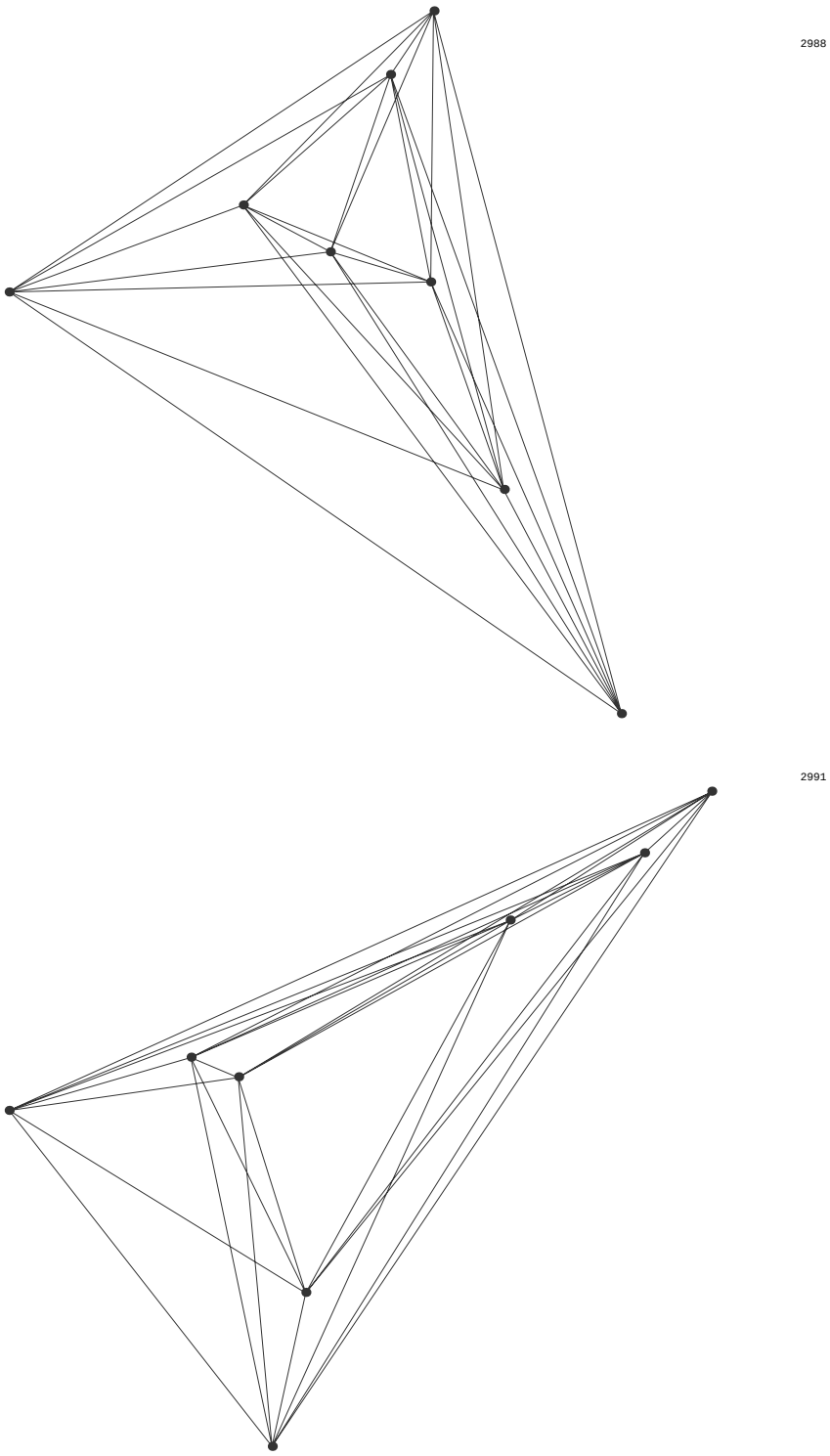

Figura 7.1: Tipo de orden (2988) que minimiza el número de $2K_2$-crossing families (primer imagen) contra el tipo de orden (2991) que minimiza el crossing number (segunda imagen).

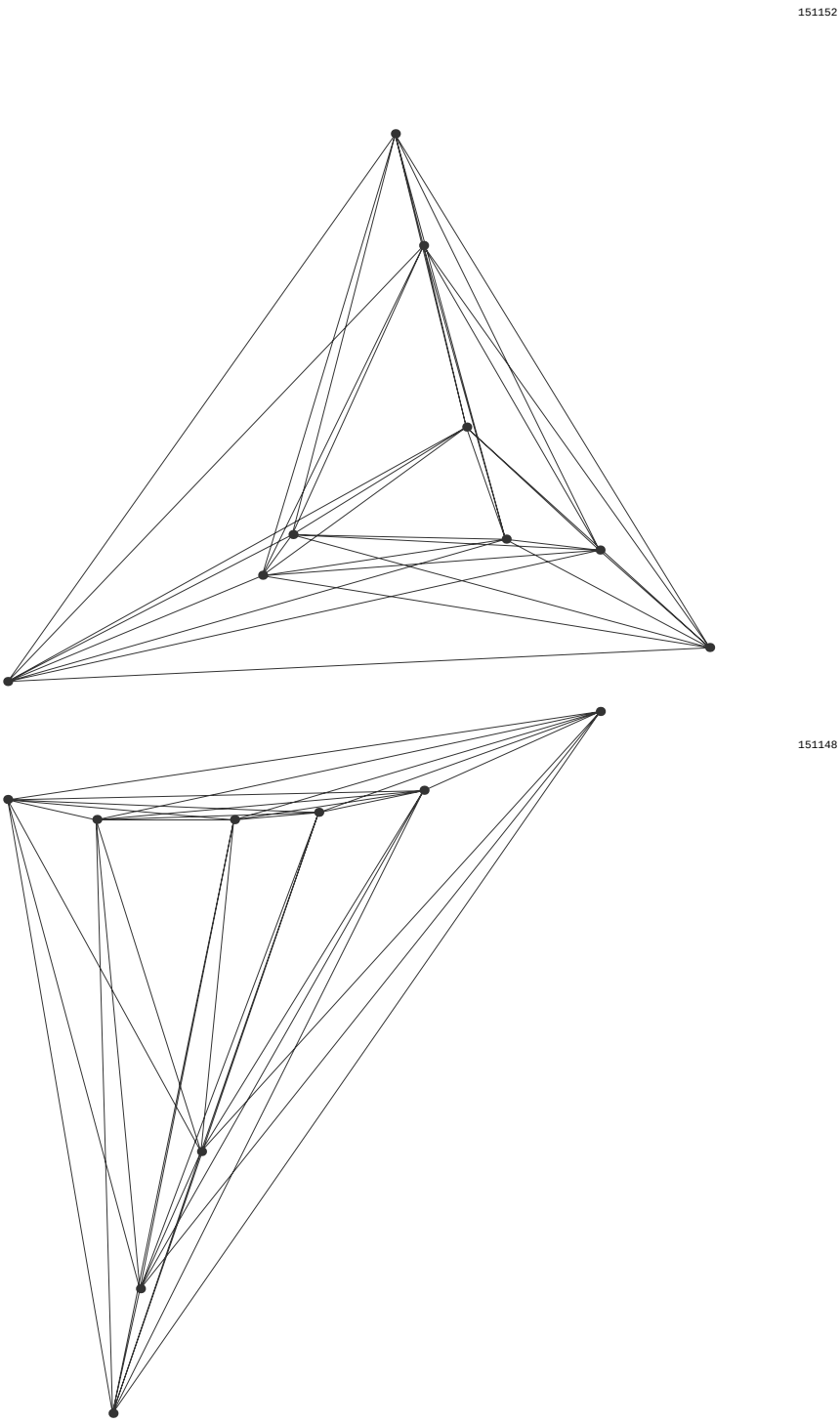

Figura 7.2: Tipo de orden (151152) que minimiza el número de $2K_2$-crossing families (primer imagen) contra el tipo de orden (151148) que minimiza el número de $K_{1,3}$-crossing families (segunda imagen).

Bibliografía

[1] O. Aichholzer, "Enumerating order types for small point sets with applications." `http://www.ist.tugraz.at/aichholzer/research/rp/triangulations/ordertypes/`, julio 2017.

[2] G. Chartrand and P. Zhang, *Chromatic graph theory*. CRC press, 2008.

[3] O. Aichholzer, F. Aurenhammer, and H. Krasser, "Enumerating order types for small point sets with applications," *Order*, vol. 19, no. 3, pp. 265–281, 2002.

[4] B. M. Ábrego, *Problemas combinatorios sobre conjuntos finitos de puntos*.

[5] P. Erdos, "On sets of distances of n points," *The American Mathematical Monthly*, vol. 53, no. 5, pp. 248–250, 1946.

[6] J. Pach and E. Sterling, "Conway's conjecture for monotone thrackles," *American Mathematical Monthly*, vol. 118, no. 6, pp. 544–548, 2011.

[7] B. Aronov, P. Erdős, W. Goddard, D. J. Kleitman, M. Klugerman, J. Pach, and L. J. Schulman, "Crossing families," in *Proceedings of the seventh annual symposium on Computational geometry*, pp. 351–356, ACM, 1991.

[8] P. Erdős and G. Szekeres, "A combinatorial problem in geometry," *Compositio Mathematica*, vol. 2, pp. 463–470, 1935.

[9] P. Valtr, "Lines, line-point incidences and crossing families in dense sets," *Combinatorica*, vol. 16, no. 2, pp. 269–294, 1996.

[10] M. J. Nielsen and W. Webb, "On some numbers related to the erdős-szekeres theorem," *Open Journal of Discrete Mathematics*, vol. 3, no. 03, p. 167, 2013.

[11] J. U. José Luis Alvarez Rebollar, Jorge Cravioto Lagos, "Crossing families and self crossing hamiltonian cycles," July 2015.

[12] S. C. Matias, "Índice cromático para gráficas geométricas completas," tesis para obtener el grado de maestro en ciencias de la computación, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, octubre 2015.