
# Universidad Nacional Autónoma de México

## Posgrado en Ciencia e Ingeniería de la Computación

## Algunos problemas de clasificación en conjuntos de puntos coloreados

# T E S I S

QUE PARA OBTENER EL GRADO DE

Doctor en Ciencias (Computación)

P R E S E N T A:

Crevel Bautista Santiago

Tutor de tesis:
Dr. Jorge Urrutia Galicia

México, D.F., 2011

*To they-who-must-be-named,*
*Eva y Horte*

# Agradecimientos

Qué difícil es expresar estas palabras (llanas)! Agradecer a quienes me han otorgado su apoyo puede ser confuso: hay tantas cosas que se pueden decir, pero no siempre el resultado final es el que se desea transmitir. Tal vez lo mejor será comenzar de una vez y esperar que el canto de la primavera me ayude a fluir.

Sin duda alguna, Jorge ha sido para mí un eje central, tanto académico como personal. La experiencia que me ha transmitido a lo largo de varios años es invaluable, así como también lo son las pláticas que hemos mantenido y los momentos divertidos que compartimos. Definitivamente, creo que no hay mejor manera de corresponder que el poner en práctica lo aprendido.

Otro pilar con el que me topé al inicio del posgrado fue dlara. ¿A quién agradecerle que las *coincidencias* nos hayan puesto juntos en ese dichoso proyecto? ¿Quién diría que horas y horas de discusiones, en ciertos momentos sin sentido, aunado a momentos particulares, nos llevarían a tener una relación tan estrecha? Gracias por todo!

Gracias también a los investigadores/amigos con los que he trabajado, sin los cuales tal vez no estaría escribiendo esta página. A Clemens Huemer y Carlos Seara, quienes en y desde Barcelona me acogieron mucho más de lo que esperaba. En efecto, mi lista no estaría completa sin agradecer a José Miguel, Inma y Pablo: excelentes los momentos, de trabajo y ocio, que pasamos en Sevilla! No puedo olvidar tampoco el agradecimiento hacia Ruy: aunque ha sido muy poco el tiempo que hemos colaborado juntos hasta ahora, cada minuto ha sido una avalancha de conocimientos y *tips*.

Mis sinodales, Sergio, Francisco y Criel, han sido de gran ayuda en momentos decisivos. Les agradezco los consejos, comentarios y tiempo que me proporcionaron a causa de este trabajo.

Por supuesto, mi familia continuó siendo parte fundamental en esta etapa académica. Una vida entera no es suficiente para equiparar lo que han hecho por mí. Horte y Care son las que más sufrieron todos los estados de ánimo por los que, supongo, pasa una persona en un periodo como éste. También, siempre ha sido inspirador y confortable el poder convivir, más que regularmente, con mi primos Leidy y Albert, así como con mis tíos Abel y Lupe.

Y qué decir de mis compañeros estudiantes, Adriana, Marco, Javier y Luis Felipe, quienes

han permeado parte de mi memoria con momentos inolvidables. Ideas acertadas, disparates, desacuerdos, coincidencias, enojos, regaños, risas, superaciones, además de las famosas *marchas*. Espero seguir compartiendo oportunidades como éstas más adelante.

Ahora, son sólo nombres los que vagan en mi cabeza: Toni, Narcis, Martha, Gil, Mónica, José, Víctor, Adrián, Hernán, Lulú, Diana, . . . Seguramente olvido a alguien, como siempre, pero lo que no olvido es el querer expresarles mi agradecimiento.

A todos, MUCHAS GRACIAS!

# Índice general

# Figuras

# Prefacio

Nuestro interés en este trabajo es mostrar los resultados de investigación obtenidos durante los años de estudio de doctorado, ésto dentro del área de las Ciencias de la Computación. Particularmente los problemas que aquí se proponen son de interés en la rama de la Geometría Computacional, por tanto, creemos pertinente plantear como primera pregunta lo siguiente: ¿Qué es la Geometría Computacional?

En general, la Geometría Computacional es un área que trabaja con un conjunto de objetos geométricos en un espacio euclideano. Por un lado, es de interés para el área el calcular propiedades geométricas entre dichos objetos; por ejemplo, dada una recta $l$ y un punto $p$, ambos en $\mathbb{R}^2$, una pregunta clásica que surge es si $p$ se encuentra por debajo, por arriba o sobre la recta $l$. En otro sentido, la Geometría Computacional se ocupa también del diseño y análisis de algoritmos para resolver problemas geométricos; más aún, es de interés para el área la complejidad computacional de los algoritmos para resolver estos problemas, ya sea que se consideren factores comunes como tiempo y espacio, u otros como el saber cuándo un algoritmo es más propenso a errores numéricos.

La Geometría Computacional es vista actualmente como una rama joven dentro de las Ciencias de la Computación. Varios científicos computacionales consideran que el área surge a partir de la presentación de la tesis doctoral de Michael Shamos [64] en 1978, o inclusive unos años antes, cuando Shamos publica en 1975 su artículo *Geometric Complexity* [63]. Algunos otros consideran como año de inicio 1968, en el cual Robin Forrest presentó su tesis doctoral [32], o quizá con alguno de sus artículos subsecuentes, como [33] y [34].

Muchos de los problemas que se estudian en Geometría Computacional representan situaciones que surgen en el mundo real, y es natural que hoy en día, dichos problemas se encuentren agrupados de acuerdo al tipo de elementos y/o técnicas involucradas. Así por ejemplo, existen problemas en los que se requiere la Ubicación de Puntos, o usar un Diagrama de Voronoi, la Triangulación de Delaunay, Estructuras de datos geométricas, Arreglos (de rectas, círculos, etc.), Gráficas de Visibilidad, por sólo mencionar algunas.

Los resultados mostrados a lo largo de este trabajo se encuentran dentro del área que se conoce como *Separación o Clasificación Geométrica*, la cual puede ejemplificarse en su forma más simple al considerar el siguiente problema: Supongamos que tenemos dos conjuntos disjuntos de objetos en algún espacio, clasificados como objetos de tipo $A$ u objetos de tipo $B$. ¿Existirá una superficie

de algún tipo específico, a la que llamaremos *separador*, la cual excluya o separe los objetos del tipo $A$ de los del tipo $B$? La pregunta anterior puede plantearse para diferentes objetos geométricos y separadores en espacios de cualquier dimensión, e inclusive puede generalizarse para cuando se cuenta con más de dos tipos de objetos.

Dentro de la literatura, es vasta la atención que se le ha dado al problema de encontrar superficies separadoras para conjuntos de puntos. Este interés surgió principalmente por la necesidad de clasificar datos en áreas distintas a la Geometría Computacional, tales como el Procesamiento de Imágenes y la Estadística, donde es útil contar con algún separador geométrico. La separación geométrica puede considerarse bajo distintos paradigmas, inducidos principalmente por distintas aplicaciones en las distintas áreas y en este documento señalamos y trabajamos con algunos de ellos.

Con el fin de establecer la amplia variedad de problemas en el área de la Separción Geométrica, en el Capítulo 1 hacemos una pequeña revisión de algunos resultados previos. Estos resultados son representativos de los paradigmas de separación que trataremos y que por tanto forman parte de los antecedentes de nuestro trabajo. Mientras tanto, en el Capítulo 2, describiremos un problema que involucra puntos de tres tipos distintos, y deseamos separar dichos puntos lo "mejor posible" por medio de dos rectas paralelas. Por otro lado, en el Capítulo 3, planteamos un problema de separación entre puntos de dos tipos al hacer uso de la menor cantidad posible de círculos.

En el Capítulo 4, nuestro interés será localizar, de un conjunto de puntos de dos tipos distintos, un subconjunto de puntos de un mismo tipo que cumpla una función de optimización en particular. Ya en el Capítulo 5, deseamos separar al menos $k$ puntos, uno de cada tipo, del resto de los elementos por medio de un corredor con forma de $L$, ésto también con base en una función de optimización.

Para tres de los cuatro problemas principales que se proponen a lo largo de este trabajo, se logró plantear algoritmos que nos permiten obtener una solución en tiempo polinomial. Sin embargo, para el problema restante fue posible plantear una demostración de su NP-dureza.

Comencemos entonces a desmenuzar los detalles de nuestros resultados, esperando que lo aquí mostrado sea de interés y utilidad para el lector.

## Publicaciones

La investigación desarrollada durante el doctorado surge como consecuencia directa de la interacción que se genera dentro del grupo que ha formado mi tutor, el Dr. Jorge Urrutia, con sus estudiantes de posgrado y otros investigadores. Cabe mencionar que la mayor parte de los resultados obtenidos pertenecen al tema central de este trabajo, que es la Separación Geométrica, aunque también se colaboró en problemas de otras áreas como lo es la Geometría Combinatoria.

A continuación mencionamos las publicaciones logradas, tanto en revistas como en actas de congresos internacionales, y en las que se presentan los resultados de nuestra investigación doctoral.

- C. Bautista-Santiago, J. M. Díaz-Báñez, D. Lara, P. Pérez-Lantero, J. Urrutia and I. Ventura. *Computing optimal islands*. Operations Research Letters 39(4) : 246-251, 2011.

- C. Bautista-Santiago, J. M. Díaz-Báñez, R. Fabila-Monroy, D. Flores-Peñaloza, D. Lara and J. Urrutia. *Covering moving points with anchored disks*. European Journal of Operational Research 216(2) : 278-285, 2012.

- C. Bautista-Santiago, M. A. Heredia, C. Huemer, A. Ramírez-Vigueras, C. Seara and J. Urrutia. *On the number of edges in geometric graphs without empty triangles*. Graphs and Combinatorics (en revisión , no. de registro GCOM-S-11-00041).

- C. Bautista-Santiago, J. Cano, R. Fabila-Monroy, D. Flores-Peñaloza, H. González-Aguilar, D. Lara, E. Sarmiento and J. Urrutia. *On the diameter of a geometric Johnson type graph*. Proceedings of the 26th European Workshop on Computational Geometry. March. 2010.

- C. Bautista-Santiago, J. Cano-Vila, J. M. Díaz-Báñez, H. González-Aguilar, D. Lara y J. Urrutia *L-corredor k-cromático*. Actas de los XIII Encuentros de Geometría Computacional. Junio, 2009.

- C. Bautista-Santiago, J. M. Díaz-Báñez, D. Lara, P. Pérez-Lantero, J. Urrutia and I. Ventura. *Computing maximal islands*. Proceedings of the 25th European Workshop on Computational Geometry. March. 2009.

- C. Bautista-Santiago, J. M. Díaz-Báñez, R. Fabila-Monroy, D. Flores-Peñaloza, D. Lara and J. Urrutia. *Covering moving points with anchored circles*. Proceedings of the XVII Euro Working Group on Locational Analysis. September, 2008.

# Estado del Arte 1

## 1.1. Introducción

Hoy en día, existen varias áreas de estudio en las que se necesita hacer una separación de los objetos con los que se trabaja. Tal vez dicho estudio se ve motivado en parte por uno de los problemas fundamentales de clasificación en el área de Aprendizaje Automático, en el que, dados dos conjuntos de objetos (conjuntos de prueba), se desea determinar un estimador que permita hacer una clasificación rápida de nuevos objetos que pertenezcan a uno de los dos conjuntos. Otra área en la que se puede apreciar el interés de separar es la Minería de Datos, donde se requiere seleccionar, de un conjunto de datos, prototipos (subconjuntos) que representen diferentes tipos de información.

Varias técnicas estadísticas se han desarrollado para el problema de clasificación, como por ejemplo las Máquinas de Vectores Soporte [15, 43]. Este problema también ha sido atacado desde el punto de vista combinatorio, particularmente en Geometría Computacional, con el fin de desarrollar algoritmos correctos y eficientes. Esta rama de estudio se conoce como *Separación o Clasificación Geométrica*.

Nuestro objetivo en este primer capítulo es plasmar algunos resultados previos referentes al concepto de separación geométrica. Mencionaremos algunos enfoques que se le han dado a este concepto, así como algunos ejemplos.

Consideremos como inicio la siguiente idea: Dada una colección $\mathcal{F}$ de conjuntos convexos, un elemento $F \in \mathcal{F}$ y una subcolección $S$ de $\mathcal{F}$, decimos que una línea $\ell$ separa a $F$ de $S$ si es que $F$ está contenido en uno de los semiplanos cerrados inducidos por $\ell$, mientras que todo conjunto de $S$ yace en el otro semiplano cerrado. Observemos que, en principio, ningún elemento de $\mathcal{F}$ se considera de algún tipo en particular; no obstante, el considerar este atributo para cada elemento será de nuestro interés en este capítulo.

## 1.2. Separación Fuerte

El concepto más simple dentro del área de Separación Geométrica es el que se conoce como *separación fuerte*, el cual se puede definir formalmente de la siguiente manera: Sean $A_1, A_2, \ldots, A_r$ objetos en $\mathbb{R}^d$, tal que $A_i \cap A_j = \emptyset$, $i \neq j$. Decimos que un conjunto finito de objetos geométricos $\mathcal{S}$ forma un separador fuerte para los conjuntos $A_i$, $i = 1, \ldots, r$, si cada componente conexa en $\mathbb{R}^d - \partial(\mathcal{S})$ contiene solamente objetos de uno de los conjuntos $A_i$, donde $\partial(\mathcal{S})$ denota la unión de las fronteras de los objetos en $\mathcal{S}$.

Así, para el caso en el que la separación fuerte sea posible, podemos decir que cada componente conexa inducida por el separador $\mathcal{S}$ es *monocromática*.

A continuación, mencionaremos algunos problemas en los que se requiere de una separación fuerte, ésto al hacer uso de distintos objetos geométricos como separadores. En particular, nos ubicaremos en $\mathbb{R}^2$, y tomaremos dos conjuntos finitos de puntos, $B$ y $R$. Nos referiremos a $B$ como el conjunto de puntos azules, mientras que $R$ representará a un conjunto de puntos rojos.

### 1.2.1. Separación lineal

La noción más natural de separación en el plano considera una línea recta $\ell$ como separador; cuando esto es posible, decimos que $R$ y $B$ son *linealmente separables*. Observemos que $B$ y $R$ son linealmente separables si y sólo si sus cierres convexos no se intersecan [67]. Ver Figura 1.1.

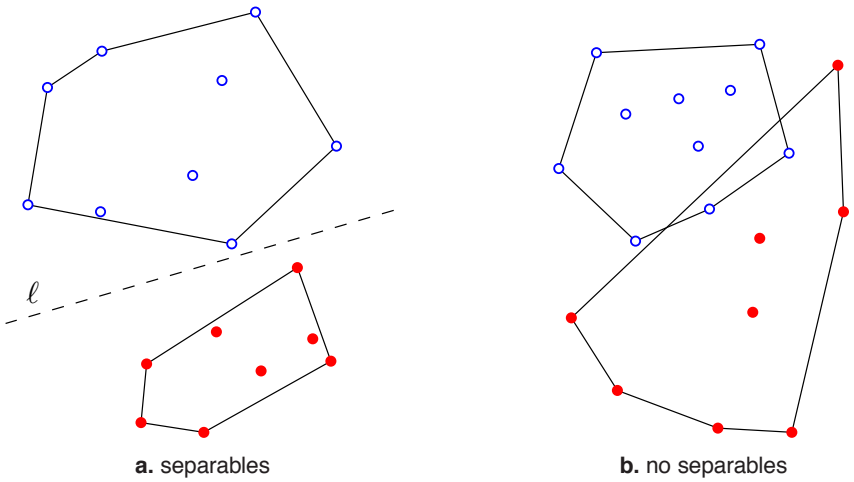


Figura 1.1: En (a), Los cierres convexos de $R$ y $B$ no se intersecan, mientras que en (b) no es posible separar los conjuntos linealmente.

Supongamos que $|B| = n$, $|R| = m$ y que $N = \max\{n, m\}$. En [65] se muestra que el problema de decisión para la separación lineal, dados $R$ and $B$, puede resolverse en $O(N \log N)$ tiempo. De hecho, en dicho artículo se conjeturó, erroneamente, que esta complejidad era óptima. No obstante, años después se demostró en [57] que este problema puede resolverse en $\Theta(N)$ tiempo haciendo

uso de programación lineal.

Cabe mencionar que, en los casos en que $R$ y $B$ son conjuntos de círculos, polígonos o inclusive segmentos de recta, el problema de separación lineal puede resolverse en la misma complejidad, $\Theta(N)$ tiempo [47].

Así también, para un par de conjuntos de puntos linealmente separables, es posible determinar el conjunto de todas las soluciones posibles en $\Theta(N)$ tiempo [62]. Ésto se logra de la siguiente manera:

- Supongamos que $B$ y $R$ son linealmente separables por una recta vertical. Podemos calcular en $\Theta(N)$ tiempo las líneas soporte interiores de ambos conjuntos de puntos, sin necesidad de calcular sus cierres convexos, ya sea usando un algoritmo de podado y búsqueda, descrito en [39], o por medio de programación lineal en dos variables.
- Calculamos el intervalo de pendientes definido por las líneas de soporte . La pendiente de cualquier línea separadora para $R$ y $B$ está dentro de dicho intervalo (Figura 1.2).

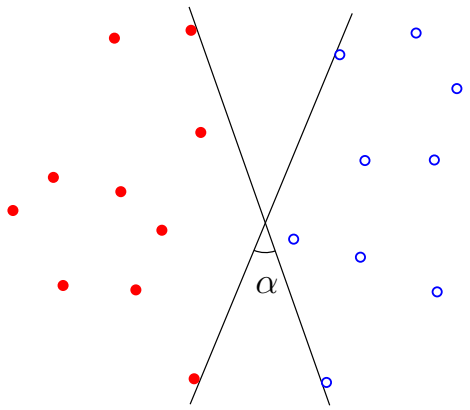


Figura 1.2: Intervalo de pendientes inducido por las líneas soporte interiores de $B$ y $R$.

Más aún, cuando $B$ y $R$ son conjuntos de puntos, segmentos de recta o polígonos, se puede afirmar lo siguiente: Dada una pendiente $m$, podemos determinar todas las líneas separadoras con pendiente $s$ en $O(N)$ por medio de lo siguiente:

- Sin pérdida de generalidad, supongamos que $s$ es la pendiente vertical. Proyectemos los puntos de $B \cup R$ sobre una recta horizontal $l$ y supongamos que todas las proyecciones de los puntos azules están a la izquierda de todas las proyecciones de los puntos rojos.
- Encontramos la proyección del punto azul con la mayor abscisa y la proyección del punto rojo con la menor abscisa. Entonces, podemos determinar el intervalo de $l$ que se encuentra entre dichos puntos. Por tanto, cualquier línea separadora para $B$ y $R$, con pendiente $s$, pasa por un punto de este intervalo.

### 1.2.2. Separación por una cuña

Otro problema de separación fuerte dentro de la literatura refiere a la cuña como superficie separadora de un par de conjuntos en el plano [49]. Decimos que $B = \{b_1, \ldots, b_n\}$ y $R = \{r_1, \ldots, r_m\}$ son separables por una cuña si es que existe una cuña que contiene únicamente a todos los puntos de uno de los conjuntos.

Supongamos que $n, m \geq 3$, además que $R$ y $B$ están en posición general, es decir, no existen tres o más puntos colineales en alguno de los conjuntos. Si los cierres convexos de $B$ y $R$, $CH(B)$ y $CH(R)$ respectivamente, no se itersecan, entonces estos conjuntos son linealmente separables y por tanto también existe una cuña que los separa.

Observemos que una condición necesaria para la separación, por medio de una cuña, es que el cierre convexo de uno de los conjuntos sea monocromático, es decir, que sólo contenga puntos de un sólo color. Podemos obtener el cierre convexo de cada uno de los conjuntos en $O(N \log N)$, y también verificar si alguno de ellos es monocromático. De no cumplirse esta condición, podemos afirmar que dichos conjuntos no son separables por medio de una cuña.

Supongamos ahora que $B$ y $R$ no son linealmente separables y sin pérdida de generalidad, digamos que el cierre convexo de $B$ es monocromático. Dado un punto $p$ en el plano, podemos decidir en $O(n)$ si $B$ y $R$ son separables por una cuña cuyo ápice sea $p$ de la manera siguiente:

- Por medio de programación lineal, podemos decidir si $p$ y $B$ son linealmente separables. En caso de que ésto ocurra, calculamos las líneas soporte entre $p$ y $CH(B)$, ésto por ejemplo, calculando las pendientes de las rectas que pasan por $p$ y cada punto $b_i \in B$, quedándonos al final con los extremos.
- Verificamos que la cuña inducida por las líneas soporte del punto anterior no contenga ningún punto rojo (Figura 1.3).

En [49], también se establecen los resultados siguientes sobre la separación de dos conjuntos de puntos por medio de una cuña.

**Teorema 1.1.** *Sean $B$ y $R$ dos conjuntos disjuntos de puntos en el plano, con $n$ y $m$ elementos respectivamente. Decidir si dichos conjuntos son separables por una cuña y calcular la región en el plano de puntos $p$, tal que $p$ es ápice de una cuña separadora para $B$ y $R$, puede hacerse en $O(N \log N)$ tiempo.*

**Proposición 1.** *Sean $B$ y $R$ dos conjuntos disjuntos en el plano separables por una cuña, con $n$ y $m$ puntos, respectivamente. Supongamos que ya se ha calculado la región del plano inducida por los ápices de las posibles cuñas separadoras, entonces las cuñas con menor y mayor ángulo pueden encontrarse en $O(N)$ tiempo.*

De hecho, estos mismos resultados se han enunciado para cuando $B$ y $R$ son conjuntos de círculos. Así también, el teorema es válido cuando $R$ y $B$ son conjuntos de polígonos y segmentos de recta; sin embargo, para éste último caso de segmentos de recta, el resultado presentado en la Proposición 1 cambia ligeramente:

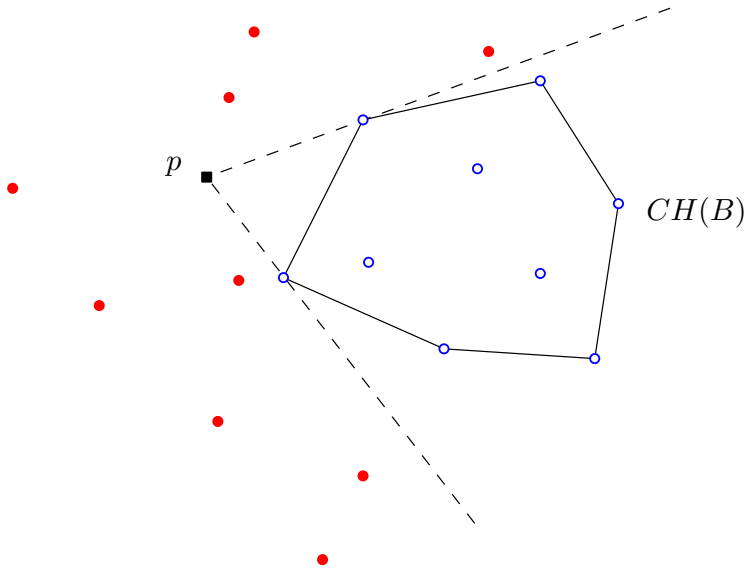


Figura 1.3: Cuña inducida por un punto $p$ y $CH(B)$.

**Proposición 2.** *Sean $B$ y $R$ dos conjuntos disjuntos en el plano separables por una cuña, con $n$ y $m$ segmentos de recta, respectivamente. Supongamos que ya se ha calculado la región del plano inducida por los ápices de las posibles cuñas separadoras, entonces las cuñas con menor y mayor ángulo pueden encontrarse en $O(N\alpha(N))$ tiempo, donde $\alpha(\cdot)$ denota la función inversa de Ackerman.*

Otros separadores geométricos han sido también objeto de estudio desde el punto de vista de la separación fuerte, tales como la banda, la doble cuña o una polígonal con giro de tamaño constante. Sin embargo, no es nuestra intención en este trabajo presentar una recopilación de este tipo de resultados. Para más detalles sobre este paradigma de separación, la referencia [62] es un excelente punto de inicio.

## 1.3. Separación débil

Los primeros resultados referentes a problemas de separación en Geometría Computacional lidian con encontrar un separador que aisle completamente los objetos de una misma clase de los de otras clases, como se describió en la sección anterior. No obstante, hablando en términos prácticos, es muy probable que los conjuntos de objetos en cuestión no sean completamente separables, ésto a causa de, por ejemplo, errores de muestreo, redondeo o algún factor de ruido. En dicho escenario, algoritmos como los mostrados en la Sección 1.2 simplemente reportarían que el conjunto de objetos no se puede separar y terminarían su ejecución.

Recordemos, de la sección anterior, que contamos con dos o más conjuntos disjuntos de objetos en $\mathbb{R}^d$: $A_1, \ldots, A_r$. Así también, $\mathcal{S}$ representa un conjunto finito de superficies. En esta sección

consideramos el caso en el que la separación fuerte no siempre es posible, esto es, que forzosamente se generen componentes conexas en $\mathbb{R}^d - \mathcal{S}$ que no sean monocromáticas.

Decimos que una componente conexa es de la clase $A_i$, si la mayor cantidad de objetos en el interior de dicha componente pertenece al conjunto $A_i$. Un objeto en una componente de la clase $A_i$ está *mal clasificado* si dicho objeto pertenece a algún conjunto $A_j$, $j \neq i$. Entonces, diremos que $\mathcal{S}$ forma un *separador débil* para dos o más conjuntos disjuntos de objetos, si en cada componente conexa en $\mathbb{R}^d - \mathcal{S}$ se minimiza el número de objetos mal clasificados.

Con el fin de describir algunos problemas de separación bajo este paradigma, retomemos nuevamente el conjunto $R$ de puntos rojos y el conjunto $B$ de puntos azules, ambos en $\mathbb{R}^2$.

### 1.3.1. Separación lineal débil

Observemos que siempre es posible construir un conjunto $R \cup B$ de $n$ puntos, de manera que no exista una línea recta que separe fuertemente a los puntos rojos de los azules. En este caso, sería deseable determinar una recta $\ell$ que separe lo mejor posible al conjunto de puntos, es decir, que cada uno de los semiplanos inducidos por $\ell$ contenga la mayor cantidad de puntos de una sola de las clases cromáticas (Figura 1.4).

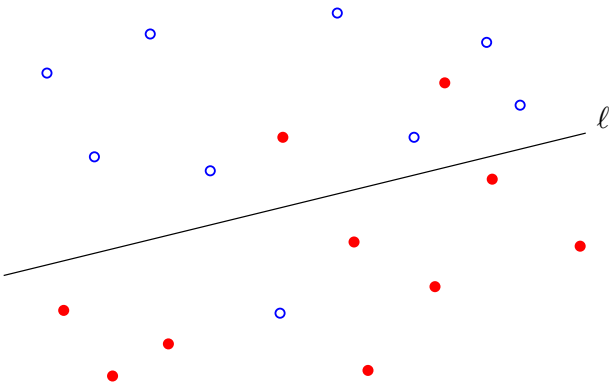


Figura 1.4: Línea separadora débil para $R \cup B$. Hay tres puntos mal clasificados.

Para una $k$ dada, en [12] se presenta un algoritmo que en $O(nk \log^2 n)$ tiempo encuentra los separadores con a lo más $k$ puntos mal clasificados. Ya en [46], se plantea un primer algoritmo que en $O(n^2)$ tiempo permite encontrar una recta $\ell$ que separa débilmente a $R$ de $B$. Cabe mencionar que este último algoritmo hace uso de la *dualidad punto-recta* para representar los puntos en $B$ y $R$; el arreglo de rectas obtenido se explora por medio de un barrido topológico de recta para encontrar la solución.

Posteriormente en [28] se propuso un algoritmo sensible a la salida para este mismo problema. Así, el algoritmo de dicha propuesta, sin pérdida de generalidad, encuentra una recta $\ell$ tal que la mayor cantidad de puntos azules quede en el semiplano superior inducido por $\ell$, mientras que la mayor cantidad de puntos rojos se encuentre en el semiplano inferior. El resultado que se establece

es el siguiente:

**Teorema 1.2.** *Sea B un conjunto de n puntos azules, mientras que R representa un conjunto de n puntos rojos; ambos en el plano. Es posible encontrar una recta que separa débilmente a B de R en $O(nm \log m + n \log n)$ tiempo, donde m representa el número de puntos mal clasificados.*

De igual manera en [28], se propone un algoritmo que permite encontrar una recta que separa débilmente un conjunto de polígonos convexos rojos de un conjunto de polígonos convexos azules.

Para cerrar esta subsección, mencionaremos que en [11] se propone un algoritmo que además de ser sensible a la salida, maneja un enfoque aleatorio para dar solución al problema de la separación lineal débil. Este algoritmo trabaja en $O((n + m^2) \log n)$ tiempo. Otro dato interesante es el que se demuestra en [60], donde se establece que el problema de la separación lineal débil es *3-suma-duro* [37], es decir que el mejor algoritmo que se conoce para solucionarlo toma $\Theta(n^2)$ tiempo, o más, en el peor caso.

### 1.3.2. Separación débil con dos rectángulos isotéticos

Sea $S = R \cup B$ un conjunto de puntos rojos y azules en el plano en posición general, con $|S| = n$. En esta subsección describimos algunos detalles para el siguiente problema: encontrar el subconjunto de $S$ de mayor cardinalidad, tal que éste pueda ser cubierto por la unión de dos rectángulos isotéticos (lados paralelos a los ejes) $\mathcal{R}$ y $\mathcal{B}$, no necesariamente disjuntos, de tal manera que $\mathcal{B}$ (respectivamente $\mathcal{R}$) contenga solamente puntos azules (respectivamente rojos); ver Figura 1.5.

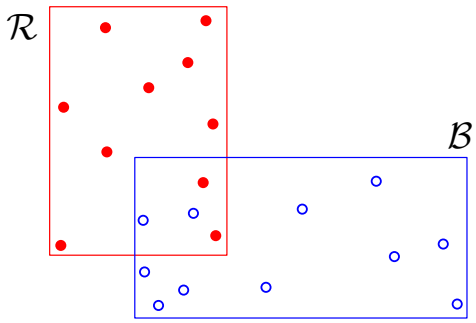


Figura 1.5: La solución con dos rectángulos isotéticos en este caso tiene $n - 4$ puntos.

Este problema fue introducido por primera vez en [14], donde se propone un algoritmo de $O(n^3)$ tiempo y espacio para resolverlo. En [13], en cambio, se da una mejora al mostrar que este problema puede resolverse en $O(n^2 \log n)$ tiempo y $O(n)$ espacio. El algoritmo propuesto en [13] se basa principalmente en resolver la versión dinámica del problema de la *subsecuencia consecutiva de suma máxima*, propuesto por Bentley en [6]. Para lograr dicha complejidad, los autores hacen uso de una estructura de datos, a la cual denominan *árbol MCS*, para representar y operar los datos de interés que se encuentran de manera implícita en el conjunto $S$.

En contraste con la separación fuerte, la separación débil ha sido mucho menos estudiada; no obstante, podemos agregar a los dos problemas de separación con puntos mal clasificados descritos anteriormente otros resultados, tales como [48], en el que el objeto separador en cuestión es una esfera, mientras que en [42] se trabaja con un conjunto de hiperplanos que optimizan una función particular con respecto a los puntos.

## 1.4. Separación por cubrimiento

El paradigma de separación por cubrimiento puede verse como una versión que surge a partir de la separación fuerte; a diferencia de este último enfoque, en la separación por cubrimiento el interés se centra totalmente sobre una sola clase cromática en un conjunto bicoloreado de puntos. Este paradigma puede expresarse de manera general en $\mathbb{R}^2$ como sigue.

Sea $S = R \cup B$ un conjunto de puntos rojos y azules en posición general en el plano. Decimos que un conjunto finito de objetos $A_1, A_2, \ldots, A_m$ forman un *separador por cubrimiento* o *cubren* una clase cromática, digamos $R$, si $m$ es el mínimo número de elementos tal que todo punto rojo está contenido en algún $A_i$ y todo $A_i$ no contiene ningún punto azul, con $1 \leq i \leq m$.

En realidad no hay muchos resultados que abordan este enfoque de separación, pero mencionaremos algunos que son representativos en la siguiente sección.

### 1.4.1. Separación por cubrimiento con rectángulos isotéticos

Un ejemplar dentro del paradigma de separación por cubrimiento es el problema que requiere el uso de rectángulos para llevar a cabo la separación: dado $S = R \cup B$, deseamos encontrar un conjunto de cardinalidad mínima $\mathcal{R}$ de rectángulos isotéticos y abiertos, tal que cada uno de sus elementos no contenga puntos azules en su interior y cada punto rojo esté cubierto por al menos un rectángulo de $\mathcal{R}$; ver Figura 1.6.

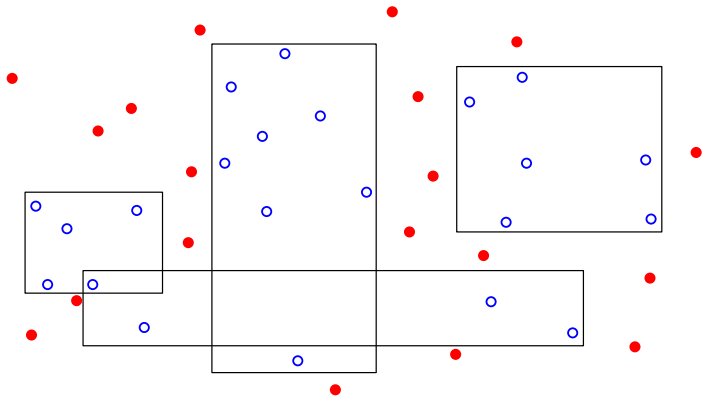

Figura 1.6: Separación por cubrimiento haciendo uso de rectángulos isotéticos.

En [7] se ha demostrado que este problema es $NP$-duro, además de plantearse un algoritmo de aproximación logarítmica para dar una solución, es decir, un algoritmo cuya solución al problema es a lo más logaritmo veces más grande que la solución exacta. Así también se muestra que la versión en la que sólo se usan cuadrados isotéticos admite un algoritmo de aproximación con un factor constante. De hecho, en dicho trabajo se tratan distintas versiones de separación por cubrimiento inspiradas por el uso de rectángulos, y una de las pocas que puede resolverse en tiempo polinomial, además de ejemplificar algunas de las técnicas que se utilizan al resolver este tipo de problemas, se comenta a continuación.

### 1.4.2. Separación por cubrimiento con bandas

Los tipos de objetos con los que se desea separar la clase roja de la azul, en esta versión propuesta también en [7], son bandas horizontales o verticales y semi-planos alineados a los ejes coordenados, los cuales son llamados también bandas para simplificar la notación. Sea $\mathcal{R}^*$ el conjunto formado por todas las bandas más anchas que no contienen puntos azules en su interior y cuyos lados pasan por puntos azules o están en el infinito, como es el caso de un semi-plano. Observemos que si una banda participa en una solución para nuestro problema, entonces ésta puede hacerse más ancha hasta que cada una de las rectas que definen la banda contengan al menos un punto azul, es decir, cualquier solución al problema puede puede verse como un subconjunto de $\mathcal{R}^*$; véase la Figura 1.7.

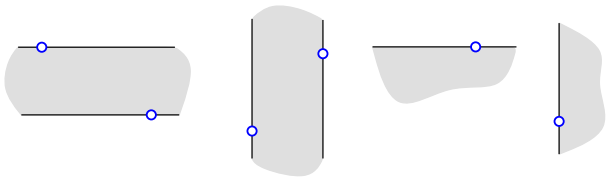

Figura 1.7: Tipos de *bandas* que participan en una versión de separación por cubrimiento.

Sea $|R| = r$ y $|B| = b$. Los puntos en $S = R \cup B$ pueden ordenarse por coordenada $x$ y $y$ en $O(r \log r + b \log b)$ tiempo, una vez hecho ésto $\mathcal{R}^*$ puede obtenerse en tiempo lineal.

Observemos que hay una solución para este problema si y sólo si para cada punto rojo existe una recta, paralela a algún eje coordenado, que pase por dicho punto rojo y no contenga ningún punto azul. Ésto puede verificarse en tiempo lineal.

Supongamos entonces que una instancia dada tiene solución. Si un punto rojo y uno azul yacen sobre una misma recta horizontal (respectivamente vertical), entonces dicho punto rojo puede cubrirse por una sola banda en $\mathcal{R}^*$. Agregamos este tipo de bandas a nuestra solución y removemos los puntos rojos que fueron cubiertos. Ésto puede hacerse también en tiempo lineal. Cada uno de los puntos rojos que permanece en nuestro conjunto puede cubrirse por dos bandas en $\mathcal{R}^*$ y cómo resolver la selección de una de ellas puede hacerse de la siguiente manera.

Sea $G = (V, E)$ la gráfica cuyo conjunto de vértices es el conjunto de bandas en $\mathcal{R}^*$ que cubren al menos un punto rojo, y donde hay una arista entre los vértices que representan a las bandas $R_1$

y $R_2$ si y sólo si $R_1 \cap R_2$ contiene un punto rojo. No es difícil ver que $G$ es una gráfica bipartita con $O(b)$ vértices y $O(r)$ aristas, y además puede construirse en $O(r + b)$ tiempo.

Ya que cada punto rojo se cubre por exactamente dos bandas, el problema se reduce a encontrar una *cubierta de vértices* de cardinalidad mínima en $G$ [38]. Sin embargo, cuando la gráfica en cuestión es bipartita, el problema de la cubierta de vértices es equivalente al problema del *apareamiento máximo* (teorema de König), y éste último puede ser resuelto en $O(\sqrt{|V|}\,|E|) = O(\sqrt{b}\,r)$ tiempo [45]. Así, se concluye el siguiente resultado:

**Teorema 1.3.** *El problema de separación por cubrimiento con bandas isotéticas puede ser resuelto en* $O(r \log r + b \log b + \sqrt{b}\,r)$ *tiempo.*

De hecho, el planteamiento de los problemas en estas dos últimas subsecciones está inspirado en un problema clásico en el área de Minería de Datos que se conoce como el *problema de la cubierta de clase* [10, 18, 55], donde el separador por cubrimiento está formado por un conjunto de círculos. En [10] se demuestra que este problema es $NP$-duro, y además se presenta un algoritmo que obtiene una $(1 + \ln n)$-aproximación para espacios métricos en general. Ya en particular, para puntos en $\mathbb{R}^d$ con la métrica Euclideana, se proporciona un esquema de aproximación de tiempo polinomial (PTAS).

Por otro lado, en [2] se plantea también un problema con el paradigma de separación por cubrimiento, pero en éste se requiere del menor número de triángulos disjuntos. En dicho trabajo se demuestra que este problema resulta ser también $NP$-duro, ademas se muestra que se puede encontrar en tiempo polinomial un conjunto de $O(K_o \log K_o)$ triángulos ajenos que cubren al conjunto de puntos rojos sin cubrir ninguno azul, donde $K_o$ denota el número de triángulos en una solución óptima.

Un último ejemplar de este paradigma de separación puede consultarse en [58]. En esta versión, el objetivo es calcular un polígono simple en el plano con el menor número de vértices, de tal manera que dicho polígono separe a la clase roja de la azul. En particular se proporciona un algoritmo que en tiempo polinomial obtiene una aproximación logarítmica para este problema.

## 1.5. Separación $k$-cromática

La separación $k$-cromática es el último de los paradigmas relacionados con los resultados desarrollados en este trabajo. Este enfoque surge como una versión natural después del trabajo hecho en la separación fuerte o débil, sumado a resultados previos que requieren separar, de un conjunto de $n$ puntos sin colores, un subconjunto de $k$ puntos que cumplan con una función de optimización. Este paradigma puede describirse de la siguiente manera.

Sea $S = S_1 \cup S_2 \cup \ldots \cup S_k$ un conjunto de $n$ puntos $k$-coloreado, es decir, cada punto tiene asociado un único color $i$ y pertenece a $S_i$, con $1 \leq i \leq k$. Decimos que un objeto $A$ es *$k$-cromático* con respecto de $S$ si $A$ contiene al menos un punto de cada $S_i$. Si además $A$ satisface una función de optimización, entonces $A$ forma un *separador $k$-cromático* en $S$.

Los resultados que existen sobre este paradigma son realmente recientes, de inicios de la década pasada. De la misma manera que en las secciones anteriores, describiremos un par de resultados representativos y citaremos otros de interés a continuación.

### 1.5.1. Círculo $k$-cromático de radio mínimo

Bajo este paradigma de separación, uno de los problemas que surgen de manera natural es el preguntarse por el círculo de radio mínimo $k$-cromático en el plano; ver Figura 1.8. En [50] se muestra que dicho círculo puede encontrarse al hacer uso del siguiente concepto. Dado un conjunto $S = \{p_1, p_2, \ldots, p_n\}$ de puntos en $\mathbb{R}^2$, el *Diagrama de Voronoi* de $S$ es la descomposición del plano en celdas (de Voronoi), $C_1, \ldots, C_n$, tal que cada celda $C_i$ contiene a los puntos en $\mathbb{R}^2$ cuya distancia euclideana a $p_i$ es menor o igual que su distancia a cualquier otro punto en $S$. Definimos la *superficie de Voronoi* de $S$ como la gráfica de la siguiente función,

$$f(x) = \min_{p_i \in S} d(x, p_i), \quad x \in \mathbb{R}^2,$$

donde $d(\cdot, \cdot)$ denota la distancia euclideana entre un par de puntos.

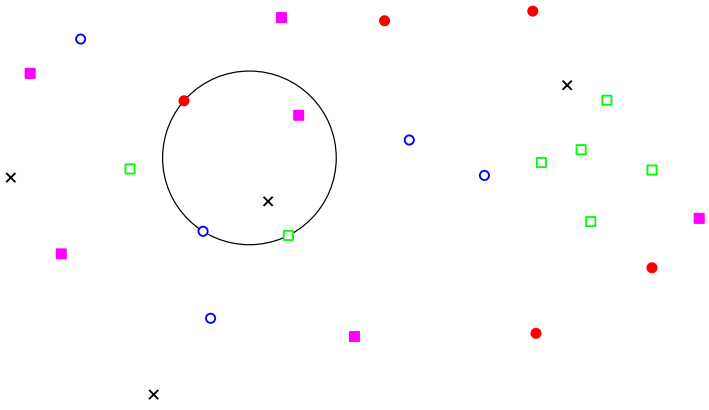

Figura 1.8: Círculo $k$-cromático de radio mínimo.

Ahora, sea $S = S_1 \cup \ldots \cup S_k$ un conjunto $k$-coloreado. Denotamos por $f_i$ la superficie de Voronoi del conjunto $S_i$. La *envolvente superior* de estas superficies es (la gráfica de) la función

$$F(x) = \max_{1 \leq i \leq k} f_i(x)$$

Lo que hay que observar en este punto es que $F(x)$ proporciona la distancia más grande de un punto $x$ a sus $k$ vecinos más cercanos, uno de cada color. Estamos interesados en los mínimos locales de $F(x)$, ya que ahí es de donde podemos obtener una solución para el problema del círculo $k$-cromático de radio mínimo. En particular, se demuestra en [50] que estos mínimos locales son

vértices o aristas de la envolvente superior, donde un vértice es un punto en el cual tres superficies de Voronoi se intersectan. Se muestra también cuál es la complejidad combinatoria (número de vértices y aristas) de la envolvente superior, así como un algoritmo para obtenerlos y cuya complejidad está dada por el siguiente teorema :

**Teorema 1.4.** *La envolvente superior de un conjunto de k superficies de Voronoi puede calcularse en* $O(kn \log kn)$ *tiempo.*

Entonces, al construir la envolvente superior, se pueden revisar los vértices y las aristas para determinar el mejor mínimo local sin costo extra.

### 1.5.2. Banda $k$-cromática de ancho mínimo

Dado un conjunto $k$-coloreado de $n$ puntos en el plano, el objetivo en uno de los problemas estudiados en [1] es buscar dos rectas paralelas (banda), de tal manera que haya al menos un punto de cada color entre el par de rectas y además la distancia entre ellas sea mínima.

Notemos que la banda solución debe tener tres puntos en su frontera, todos de distinto color, ya que si fuesen sólo dos o se repitiera un color, podría obtenerse una nueva banda más angosta al hacer una rotación.

Un primer intento para obtener una solución es el siguiente. Consideremos las $O(n^2)$ rectas definidas por dos puntos de distinto color y ordenémoslas por pendiente, lo cual puede hacerse en $O(n^2 \log n)$ tiempo. Dada una de las rectas, $\ell$, podemos proyectar sobre ella el resto de los puntos, ésto de manera perpendicular a $\ell$. Al ordenar los puntos proyectados y recorrerlos, podemos encontrar una solución en $O(n \log n)$ tiempo, cuyas dos rectas tienen pendiente $\theta + \frac{\pi}{2}$ y donde $\theta$ es la pendiente de $\ell$; Figura 1.9

Así, en cada cambio de dirección, es decir, cuando escogemos la siguiente recta dada por el orden de pendientes, lo único que necesitamos es actualizar el orden de los puntos proyectados, lo cual puede hacerse en $O(1)$ tiempo, y recorrerlos nuevamente en $O(n)$ tiempo para encontrar una nueva solución para esta nueva dirección. Entonces, este algoritmo trabaja en $O(n^3)$ tiempo.

Cuando la dirección cambia, el conjunto de puntos que forman parte de la solución óptima puede ser completamente distinto. Esta es la razón por la cual en [1] se decide no abordar una nueva manera de actualizar la solución. Como alternativa, se hace uso de algunas técnicas como la inversión círculo-rectas y la envolvente exterior, ésto para dar un mejor algoritmo, como lo establece el siguiente resultado.

**Teorema 1.5.** *Dado un conjunto k-coloreado de n puntos en el plano, es posible hallar en* $O(n^2 \alpha(k) \log k)$ *tiempo la banda k-cromática más angosta, donde* $\alpha(\cdot)$ *denota la función inversa de Ackerman.*

Por lo tanto, el algoritmo descrito anteriormente toma $O(n^2 \log n + n^2 \alpha(k) \log k)$ tiempo, en total, para encontrar una solución al problema de la banda $k$-cromática más angosta. Posteriormente,

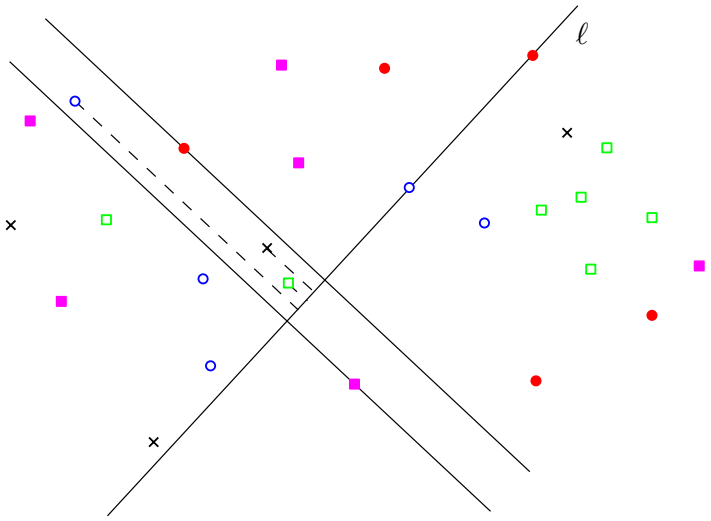


Figura 1.9: Dada la línea $\ell$, podemos obtener la banda $k$-cromática más angosta que es perpendicular a $\ell$.

en [16], se estudió nuevamente este mismo problema, obteniéndose una mejora al proponer un algoritmo que toma $O(n^2 \log n)$ tiempo para hallar una solución.

Con respecto a otros objetos que han sido estudiados bajo este mismo paradigma de separación, encontramos los siguientes. En [1] se estudia el problema de encontrar el rectángulo isotético $k$-cromático de mínima área o perímetro, y se plantea un algoritmo de $O(n\,(n-k)\log^2 k)$ tiempo para encontrarlo. Nuevamente en [16], se retoma este problema y se obtiene una mejora al presentarse un algoritmo de $O(n\,(n-k)\log k)$ tiempo para resolverlo. Más aún, en dicho artículo se proporciona también un algoritmo para encontrar en $O(n^3 \log k)$ tiempo el rectángulo $k$-cromático, no necesariamente isotético, de menor área.

Una versión dinámica del problema de identificar la banda $k$-cromática más angosta se estudia en [17], donde un nuevo punto puede agregarse al conjunto original y la banda solución en la instancia resultante puede reportarse en $O(k\,n(\alpha(n))^2 \log k)$ tiempo.

Por último, sólo mencionaremos que los resultados presentados en la subsección 1.5.1 se generalizan en [50] para usar otro tipo de métricas distintas a la euclideana. Así, al hacer uso de la métrica Manhattan, es posible encontrar el cuadrado $k$-cromático de mínima área en $O(kn \log kn)$ tiempo.

De esta manera, damos por terminada la descripción de todos los paradigmas de separación que son de nuestro interés, ésto para presentar los resultados de investigación obtenidos durante el doctorado en los capítulos subsecuentes.

# Separación débil con bandas | 2

## 2.1. Introducción

Uno de los principales problemas de clasificación en el área de *Aprendizaje Automático* (Machine Learning) es el siguiente: dados dos conjuntos de objetos, que se conocen como *conjuntos de prueba*, se desea establecer un *criterio* que facilite la clasificación de un nuevo objeto, ésto al considerar ciertos parámetros que se presentan en los conjuntos de prueba. De hecho, esta problemática es uno de los motivos que fomentó el estudio de problemas de separación geométrica.

Como mencionamos en el Capítulo 1, el paradigma de separación débil puede ser de mucha utilidad cuando no es posible separar totalmente las distintas clases de objetos involucrados. Esta imposibilidad puede tener varios orígenes, por ejemplo, una fuerte similitud entre los datos obtenidos. En otros casos, los conjuntos de prueba pueden poseer datos erroneamente obtenidos o contienen valores fuera de rangos permitidos, lo cual ocurre en casos como en el análisis de datos médicos [41]. En este tipo de situaciones, puede ser útil la eliminación de algunos datos de entrada con el fin de obtener una mejor clasificación, por lo que seleccionar el mínimo número de estos datos es importante para no perder mucha información.

Sea $S$ un conjunto de $n$ puntos en el plano en posición general, tal que $S = R \cup B \cup G$. Supondremos que los elementos en $R$ tienen asociado únicamente el color *rojo*, mientras que los elementos en $B$ poseen sólo el color *azul* y los elementos en $G$ sólo el color *verde*. Diremos entonces que $S$ es un conjunto *3-coloreado*. Dado $X \subset S$, denotamos como $Ro(X)$, $Az(X)$ y $Ve(X)$, respectivamente, al número de puntos rojos, azules y verdes en el conjunto $X$. Sean $\ell_1$ y $\ell_2$ un par de rectas paralelas en el plano, las cuales dividen $\mathbb{R}^2$ en tres componentes conexas o *bandas*. Denotamos por $S_1, S_2$ y $S_3$ a los conjuntos de puntos de $S$ que yacen en la banda izquierda, centro y derecha, respectivamente.

Sea $Q$ una de las bandas inducidas por $\ell_1$ y $\ell_2$. Si el número de puntos rojos contenidos en $Q$ es mayor que el número de puntos azules y verdes, entonces diremos que la banda $Q$ es de color rojo. Más aún, diremos que los puntos azules y verdes en $Q$ están *mal separados* o *mal clasificados* .

En este capítulo nos enfocaremos en trabajar con el *problema de separación débil ordenada con tres bandas*. Dado un orden específico de tres colores, por ejemplo *rav* (rojo-azul-verde), este

problema consiste en encontrar dos rectas paralelas $\ell_1$ y $\ell_2$, tales que los conjuntos $S_1, S_2$ y $S_3$, contenidos en cada una de las bandas inducidas, maximicen el valor de:

$$Ro(S_1) + Az(S_2) + Ve(S_3)$$

entre todo par de rectas paralelas; ver Figura 2.1.

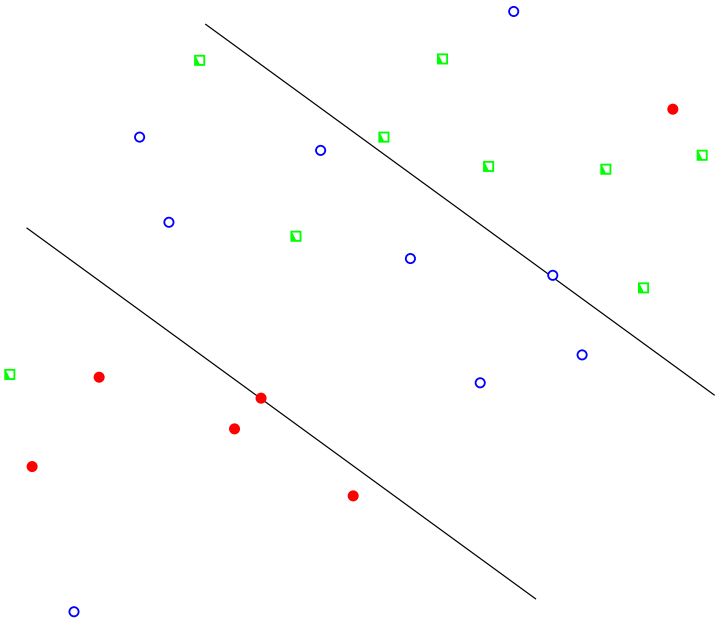

Figura 2.1: Solución al problema de separación débil con tres bandas, dado el orden rav.

En otras palabras, nuestro problema consiste en encontrar dos rectas paralelas $\ell_1$ y $\ell_2$, de tal manera que minimizemos el número de puntos mal clasificados de acuerdo al orden de color impuesto.

La motivación de este problema proviene principalmente de aplicaciones en el área de *Ubicación de servicios*, en la cual generalmente se desea ubicar de manera óptima algún servicio, como líneas de gas, agua o fibra óptica, de acuerdo a restricciones impuestas por elementos previamente observados en el terreno en cuestión. Así, una interpretación de nuestro problema en dicha área sería la siguiente: Una compañia farmaceutica requiere decidir la posición de dos líneas principales, una de gas y una de agua, de tal manera que el mayor número de estaciones de trabajo que requieren de ambos elementos para llevar a cabo sus procesos queden entre ambas tuberías, mientras que la mayor parte de las que sólo requieren de gas o agua queden en los costados. Todo ésto con el fin de elaborar un plan de ahorro al hacer la conexión de las estaciones con las líneas de distribución.

## 2.2. Bandas paralelas al eje $y$

Sea $S$ un conjunto de $n$ puntos en el plano en posición general, tal que $S$ es un conjunto 3-coloreado. Dado un orden de tres colores, denotamos como $m$ al máximo número de puntos que podemos clasificar correctamente por medio de las bandas inducidas por dos rectas $\ell_1$ y $\ell_2$, ambas paralelas al eje $y$.

En esta sección describiremos un algoritmo que en $O(n \log \log n)$ tiempo nos permite determinar la posición de las rectas $\ell_1$ y $\ell_2$, con la garantía de que se minimiza el número de puntos mal clasificados, de acuerdo a un orden de color predeterminado. Sin pérdida de generalidad, de aquí en adelante, asumiremos en todo momento que el orden de color impuesto para una solución es *rojo-azul-verde*.

Un elemento que será fundamental para nuestra solución es el de la *subsecuencia creciente más larga* de una secuencia de números, por lo que dedicaremos un breve espacio a continuación para introducir los conceptos necesarios.

### 2.2.1. Subsecuencia creciente más larga

Consideremos una secuencia $\mathfrak{a} = (a_1, a_2, \ldots, a_n)$ de $n$ números (no necesariamente distintos). Una subsecuencia creciente de $\mathfrak{a}$, $(a_{i_1}, a_{i_2}, \ldots, a_{i_q})$, satisface que $a_{i_1} \leq a_{i_2} \leq \ldots \leq a_{i_q}$, donde $i_1 < i_2 < \ldots < i_q$. El entero $q$, con $1 \leq q \leq n$, representa la *longitud* de la subsecuencia creciente.

Uno de los primeros artículos en los que se requería determinar la subsecuencia creciente más larga surgió a principios de los años 60 [61]. Sin embargo, en dicho trabajo no era de particular interés el establecer la complejidad del algoritmo inducido por los resultados mostrados. Fue hasta casi quince años después que en [36] se planteó un algoritmo que en $O(n \log n)$ tiempo permitía establecer únicamente la longitud de dicha subsecuencia. Ambos artículos basan sus resultados en el uso de una estructura que se conoce como *Young tableau*, la cual es un arreglo de columnas y renglones de tal manera que los números en cada renglón (columna) forman una secuencia creciente.

Años después, en [22, 19] , se planteó otro algoritmo que en el mismo tiempo, y usando un paradigma de Programación Dinámica, obtiene los elementos de una subsecuencia creciente más larga. Posteriormente, en [59] se publicó una modificación al algoritmo propuesto en [36], el cual también permite determinar una solución pero con una complejidad distinta. Este último algoritmo es de hecho bastante simple y puede apreciarse en el Listado 2.1.

Listado 2.1: Obtención de una subsecuencia creciente más larga en una secuencia

```
1 Entrada. La secuencia a = (a_1, a_2, ..., a_n). La secuencia b y la pila P_1, ambas
      inicialmente vacías;
2
3 Procedimiento.
4 i := 1; Colocar a_1 en el tope de la pila P_1, y también como primer
      elemento de b;
```

```
5
6  /* Al inicio de cada iteración en el siguiente ciclo, las pilas P_1, ..., P_m son no vacías. La secuencia
       b consiste de m elementos a_{i_1}, ..., a_{i_m}, los cuales ocupan los topes de las pilas B_1, ..., B_m,
       respectivamente */
7  mientras (i < n) {
8      i := i + 1;
9      Si a_i ≤ a_{i_m}, entonces r := r + 1; a_{i_m} := a_i y creamos una nueva pila B_m
           con a_{i_m} en el tope; creamos el apuntador a_i ← a_{i_{m-1}};
10     De lo contrario, encontramos el mínimo j tal que a_i < a_j;
           remplazamos a_{i_j} por a_i en b y metemos a_i en el tope de la pila P_j
           ; creamos el apuntador a_i ← a_{i_{j-1}};
11 }
12
13 /* Para obtener una subsec. crec. más larga */
14 Tomar un elemento en B_m y seguir la secuencia de apuntadores.
```

La complejidad en tiempo del algoritmo anterior está dada por el siguiente teorema, cuya demostración puede consultarse en [59]:

**Teorema 2.1.** *El algoritmo del Listado 2.1 permite extraer en $O(n \log m)$ tiempo una subsecuencia creciente más larga, $\mathfrak{a}'$, de la secuencia de entrada $\mathfrak{a}$, dónde m representa la longitud de $\mathfrak{a}'$.*

Intuitivamente hablando, la complejidad de $O(n \log m)$ tiempo se sigue del hecho que el ciclo en la línea 7 se ejecuta $n$ veces, mientras que el paso en la línea 10 requiere de $O(\log m)$ tiempo.

Un mejor algoritmo con $O(n \log \log n)$ tiempo puede consultarse en [8], y aunque dicho algoritmo tampoco es difícil de seguir, creemos que los pasos del Listado 2.1 ilustran de manera más intuitiva cómo obtener una subsecuencia creciente más larga.

Sean $e_1, e_2, \ldots, e_k$ los $k$ elementos distintos en la secuencia $\mathfrak{a}$, tal que $e_i < e_j$ si $i < j$. Es importante observar que si $k$ es constante con respecto a la longitud de $\mathfrak{a}$, entonces el paso en la línea 10 del listado anterior puede hacerse en tiempo constante, ésto al hacer uso de $k - 1$ apuntadores que indiquen la columna con menor índice en la que aparece el elemento $e_i$ en el tope de la pila correspondiente, con $2 \leq i \leq k$. Por tanto, en este caso, la complejidad del algoritmo sería de $O(n)$ tiempo.

### 2.2.2. Nuestro algoritmo

Después de recordar el concepto de la subsecuencia creciente más larga en la sección anterior, veamos cómo ésto nos ayudará a solucionar nuestro problema.

Sea $S = \{p_1, p_2, \ldots, p_n\}$ un conjunto de puntos 3-coloreado, de tal manera que los puntos están ordenados por su coordenada $x$ y no hay dos de ellos con la misma abcisa. Pensemos en este orden como una proyección de los puntos en $S$ sobre el eje $x$. Dada esta proyección, junto con el orden de color impuesto (rav), podemos obtener una secuencia de acuerdo a la siguiente asignación:

$$sec(p_i) = \begin{cases} 1 & \text{si } p_i \text{ es rojo,} \\ 2 & \text{si } p_i \text{ es azul, y} \\ 3 & \text{si } p_i \text{ es verde} \end{cases}$$

Tomemos como ejemplo al conjunto de puntos mostrado en la Figura 2.2. Ahora, nuestro conjunto de puntos está representado por una secuencia de enteros $\alpha$, y trabajaremos con ella para proponer una solución a nuestro problema.

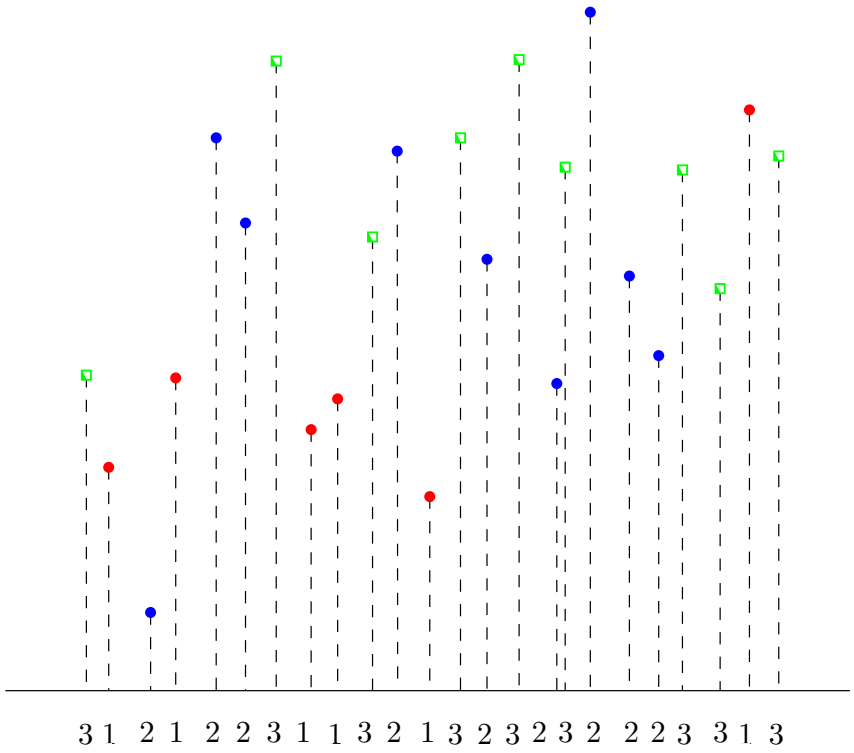


Figura 2.2: Representación de un conjunto 3-coloreado por medio de una secuencia.

Sea $\alpha'$ una subsecuencia creciente más larga de la secuencia $\alpha$. Claramente, esta subsecuencia está formada por tres bloques de elementos con valores 1, 2 y 3 respectivamente. Inducimos la posición de la recta $\ell_1$, paralela al eje $y$, como la recta que pasa por el punto cuyo elemento asociado en $\alpha'$ es el último del primer bloque. Análogamente, posicionamos $\ell_2$ fijándonos en el último punto del bloque 2; ver Figura 2.3.

Dicha posición de las rectas permite establecer un número de puntos mal clasificados en nuestro conjunto y supongamos que este número no es el mínimo. No es difícil darse cuenta que si pudiéramos clasificar de manera correcta un punto más, es decir disminuir al menos en uno el número de puntos mal clasificados, entonces $\alpha$ contendría una subsecuencia creciente de mayor longitud a la de $\alpha'$, lo cual nos lleva a una contradicción. Por tanto, podemos establecer el siguiente resultado:

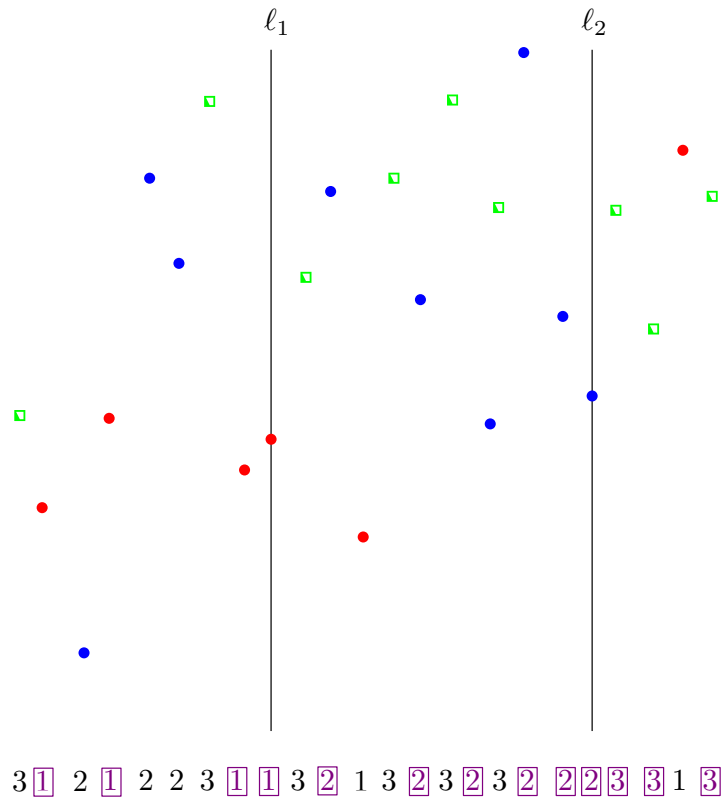


Figura 2.3: La subsecuencia creciente más larga determina la posición de $\ell_1$ y $\ell_2$.

**Teorema 2.2.** *Dado un conjunto 3-coloreado de n puntos en el plano en posición general, y un orden de los tres colores, podemos determinar en $O(n)$ tiempo las 3 bandas paralelas al eje y que separan a los puntos en $S$, minimizando el número de puntos mal clasificados de acuerdo al orden dado.*

## 2.3. Actualización dinámica de la subsecuencia creciente más larga

En esta sección describimos la herramienta base que nos ayudará a resolver el problema de separación débil ordenada con tres bandas en $O(n^2 \log n)$ tiempo. En la sección anterior mostramos, haciendo uso de la subsecuencia creciente más larga, cómo podemos resolver nuestro problema con una orientación fija. Nuestra idea para resolver el problema general es reducirlo al cálculo dinámico de la subsecuencia creciente más larga.

Recordemos que para nuestro problema, dada una orientación, podemos inducir una secuencia $\mathfrak{a} = (a_1, a_2, \ldots, a_n)$, la cual usa tres elementos distintos y representa al conjunto de puntos $S$. Lo que haremos ahora es describir la construcción de un árbol binario balanceado que nos permite representar a la secuencia $\mathfrak{a}$ junto con una de sus subsecuencias crecientes más largas. Sin pérdida

de generalidad podemos asumir que $n$ es potencia de dos, ya que en caso contrario podemos agregar al final de la secuencia elementos 0 para cumplir con dicha condición.

Cabe mencionar que la técnica usada a continuación se describe de forma general en [60]. Nuestra construcción es una adaptación particular del árbol que ahí se establece y por tanto, las complejidades en tiempo se siguen como consecuencia.

Las hojas del árbol que construimos representan a cada uno de los elementos de la secuencia $\alpha$, de tal manera que la $k$-ésima hoja del árbol, de izquierda a derecha, representa $a_k$. Cada nodo interno $v$ representa a la subsecuencia de elementos consecutivos formada por las hojas que son descendientes de $v$. Ahora, en cada uno de los nodos $v$ del árbol, guardaremos algunos datos que nos serán de utilidad:

- $I(v)$: La subsecuencia formada por todas las hojas que son descendientes de $v$.
- $(I_1(v), I_3(v))$: La pareja ordenada que representa el número de elementos 1 y 3 contenidos en $I(v)$.
- $L(v)$: La longitud de la subsecuencia creciente más larga contenida en $I(v)$.
- $(U(v), D(v), T(v))$: La tripleta ordenada que representa el número de elementos 1, 2 y 3 en la subsecuencia creciente más larga contenida en $I(v)$.
- $(L_{12}(v), L_{23}(v))$: Las longitudes de las subsecuencias crecientes más largas contenidas en $I(v)$, formadas solamente por elementos 1-2 y 2-3, respectivamente.

Sea $u$ un nodo interno en el árbol y sean $v$ y $w$ los hijos izquierdo y derecho de $u$, respectivamente. Podemos darnos cuenta que si ya tenemos los valores $I(v), I(w), I_1(v), I_1(w), I_3(v)$ e $I_3(w)$, entonces podemos obtener en tiempo constante los valores de $I(u), I_1(u)$ e $I_3(u)$. Observemos que el conjunto de elementos 2 que forman parte de la subsecuencia creciente más larga en $I(v)$ cumple con uno de los tres casos siguientes: dicho conjunto está totalmente contenido en el subárbol izquierdo o traslapa simultaneamente el subárbol izquierdo y el derecho o está totalmente contenido en el subárbol derecho; estos casos se ilustran en la Figura 2.4.

Supongamos que ya tenemos todos los datos de los nodos $v$ y $w$. Entonces, considerando los tres casos mencionados anteriormente, la longitud de la secuencia creciente más larga en el nodo $u$ está dada por:

$$L(u) = \text{máx}\,\{L(v) + I_3(w),\ L_{12}(v) + L_{23}(w),\ I_1(v) + L(w)\},$$

por lo que también el valor de $L(u)$ puede calcularse en tiempo constante. Los valores de $L_{12}(\cdot)$ y $L_{23}(\cdot)$, para cualquier nodo interno, pueden obtenerse de manera análoga, restringiendo la subsecuencia creciente más larga a dos elementos distintos. Claramente, para una hoja $a_i$ en el árbol, todos sus datos pueden calcularse en tiempo constante, lo cual implica que si llevamos a cabo un recorrido transversal de los nodos del árbol, de abajo hacia arriba, podemos calcular en $O(n)$ tiempo la información de todos los nodos. Por lo tanto, la subsecuencia creciente más larga de la secuencia $\alpha$ puede ser extraída del nodo raíz del árbol.

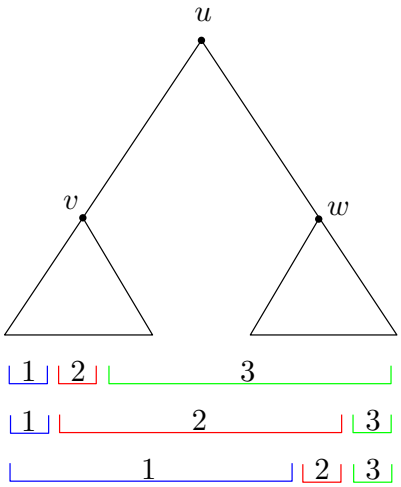


Figura 2.4: Los tres casos posibles para la subsecuencia creciente más larga en el nodo $u$.

**Observación 1.** Si el valor de $I(a_i)$ cambia, entonces en $O(\log n)$ tiempo podemos actualizar la longitud de la subsecuencia creciente más larga de $\mathfrak{a}$, ésto al recorrer la trayectoria en el árbol que va desde la hoja $a_i$ hasta el nodo raíz.

Particularmente, la propiedad del árbol mencionada en la observación anterior, será usada ampliamente por nuestra solución para el problema principal de este capítulo, que es justamente lo que trataremos en la siguiente sección.

## 2.4. Bandas sin orientación fija

Trabajemos ahora con el caso general del problema de separación débil ordenada con tres bandas, en el que requerimos encontrar la orientación y separación del par de rectas paralelas que nos garantizen el mínimo número de puntos mal clasificados en un conjunto 3-coloreado.

Hasta este punto, hemos descrito cómo solucionar el problema para una orientación fija, y ahora usaremos ésto como base para encontrar una solución al cambiar de orientación. La idea a seguir es mantener actualizada dinámicamente la subsecuencia creciente más larga al ir considerando orientaciones distintas.

Recordemos que nuestro conjunto $S$ está en posición general y que no hay dos puntos con la misma abcisa. Supongamos también, sin pérdida de generalidad, que cualesquiera dos segmentos de recta entre puntos de $S$ no son paralelos. Dado que $S$ es finito, podemos encontrar un círculo $C$ que contenga a todos los puntos de $S$ en su interior. Sea $\ell$ una recta paralela al eje $x$, tal que $\ell$ es tangente a $C$, y digamos que $\ell$ tiene una orientación hacia la derecha. Si proyectamos los puntos de $S$ sobre $\ell$, y consideramos la orientación de la recta, entonces obtenemos la secuencia $\mathfrak{a} = \{p_1, p_2, \ldots, p_n\}$ que contiene a los puntos ordenados por abcisa de manera creciente. Al rotar $\ell$ en contra de las manecillas del reloj y manteniéndola tangente a $C$, se dará un primer momento

en el que dos puntos $p_i$ y $p_{i+1}$ se proyecten en el mismo punto sobre $\ell$. Inmediatamente después de dicho momento, $p_{i+1}$ se encontrará antes que $p_i$ en la secuencia $\mathfrak{a}$; ver Figura 2.5a.

Si continuamos girando la recta $\ell$, los elementos de $\mathfrak{a}$ irán permutando, de hecho, un cambio siempre ocurre entre dos elementos consecutivos de $\mathfrak{a}$. Cuando la recta haya girado 180° en total, $\mathfrak{a}$ estará formada por los puntos de $S$ ordenados decrecientemente de acuerdo a su abcisa, es decir, el orden de los puntos se invierte totalmente; Figura 2.5b. Notemos que dos puntos durante este recorrido de $\ell$ intercambian su posición una sola vez y además, se consideran todas las posibles pendientes entre cualesquiera dos puntos.

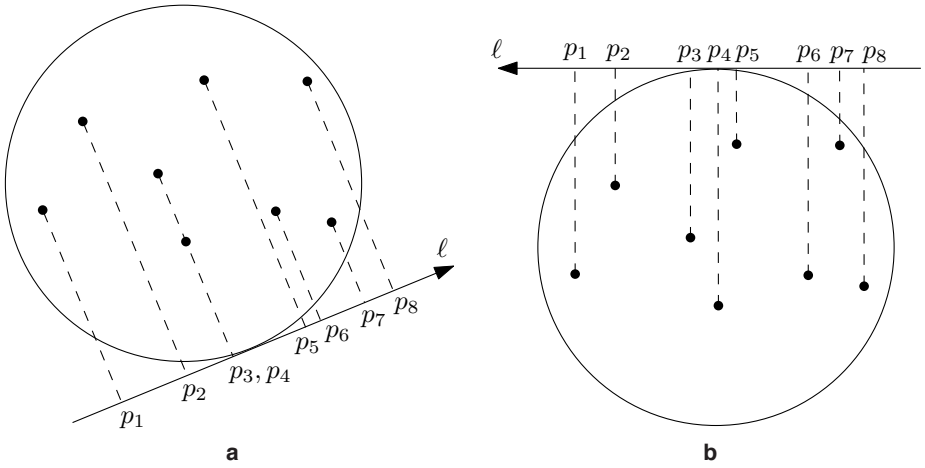


Figura 2.5: (a) Primer instante en el que dos puntos se alinean en la proyección sobre $\ell$. (b) Después de 180°, los puntos se proyectan en orden decreciente.

El conjunto de las distintas *permutaciones* o configuraciones de $\mathfrak{a}$, durante el recorrido de $\ell$, se conoce como el medio periodo de una *secuencia circular* [40]. Observemos que el encontrar dicha secuencia circular, equivale a efectuar un barrido con una recta vertical en el arreglo de rectas dual asociado a $S$, donde el intercambio de dos puntos se interpreta como la intersección de sus rectas duales; ambos procesos pueden hacerse en $O(n^2)$ tiempo.

Ahora, retomando nuevamente nuestra asignación de enteros de acuerdo al color de cada punto, 1 = rojo, 2 = azul y 3 = verde, podemos obtener las distintas permutaciones bajo esta asignación al obtener la secuencia circular asociada a nuestro conjunto de puntos. El número de permutaciones distintas es del $O(n^2)$, ya que se consideran todas las pendientes inducidas por cada par de puntos.

Con base en lo anterior, podemos considerar a la secuencia circular como el espacio discreto de búsqueda para una solución de nuestro problema de separación débil ordenado, ya que dentro del intervalo de pendientes inducido por dos permutaciones consecutivas, la proyección de los puntos de $S$ sobre $\ell$ es la misma, y por ende también su subsecuencia creciente más larga.

Por tanto, si $\mathfrak{a}'_i$ representa la subsecuencia creciente más larga de la permutación $\mathfrak{a}_i$, entonces podemos obtener en $O(\log n)$ tiempo la subsecuencia creciente más larga de la permutación $\mathfrak{a}_{i+1}$,

ésto al hacer dos actualizaciones en la estructura de datos que describimos en la sección anterior.

Podemos establecer entonces el siguiente resultado:

**Teorema 2.3.** *Dado un conjunto* 3*-coloreado de n puntos en el plano en posición general, y un orden de los tres colores, podemos determinar en* $O(n^2 \log n)$ *tiempo la orientación y posición de dos rectas paralelas que separan a los puntos en S y minimizan el número de puntos mal clasificados de acuerdo al orden dado.*

## 2.5. Más de tres colores

Nuestros problemas en las secciones anteriores involucraban únicamente tres colores, pero ¿qué podemos decir cuando el número de colores $k$ es mayor a tres?

Observemos que el espacio de búsqueda que proponemos para nuestro problema con tres bandas no cambia a causa del aumento en el número de colores, por lo que podemos usar el mismo algoritmo de $O(n^2)$ tiempo para obtener las distintas permutaciones. Además, es posible generalizar de manera directa la técnica, al hacer uso del árbol binario balanceado, para mantener dinámicamente la subsecuencia creciente más larga conforme obtenemos una nueva permutación.

Para el caso de $k = 3$, hay que notar que en cada nodo interno de la estructura de datos que proponemos, se hace uso de información que involucra a los tres colores a la vez, pero también la que sólo involucra dos o un sólo color. Entonces, cuando el número de colores aumenta, necesitaremos usar una cantidad mayor de información para poder mantener lo que nos interesa: la longitud de la subsecuencia creciente más larga. En realidad, la información que se requiere para cualquier valor de $k$, en cada uno de los nodos internos $u$ de nuestro árbol, puede verse como una matriz triangular de dimensión $k \times k$, la cual tiene la siguiente forma:

$$\begin{pmatrix} I_1(u) & I_2(u) & I_3(u) & \cdots & I_k(u) \\ & L_{12}(u) & L_{23}(u) & \cdots & L_{k-1k}(u) \\ & & L_{123}(u) & \cdots & L_{k-2k-1k}(u) \\ & & & & \vdots \\ & & & & L(u) \end{pmatrix}$$

Cada valor en el primer renglón de la matriz guarda la longitud de la subsecuencia creciente más larga que considera un sólo elemento, es decir, mantiene el número de apariciones de un elemento en la secuencia que representa el nodo $u$. En general, el $i$-ésimo renglón mantiene las longitudes de las subsecuencias crecientes más largas que hacen uso de $i$ elementos consecutivos en el orden de colores dado. Además de esta matriz, necesitamos mantener también el valor para $I(u)$, la subsecuencia que representa el nodo $u$, y la $k$-tupla que contiene en su $j$-ésima entrada el número de elementos $j$ en la subsecuencia creciente más larga del nodo $u$.

Recordemos nuevamente que en [60] se describe esta estructura de datos en forma general, en la cual toma $O(k)$ tiempo el hacer las modificaciones necesarias en cada nodo, ya que una

entrada por cada uno de los $k$ renglones de la matriz puede necesitar una modificación. Por lo tanto, para el caso general del problema de separación débil ordenada con bandas, tenemos el siguiente resultado:

**Teorema 2.4.** *Dado un conjunto $k$-coloreado de $n$ puntos en el plano en posición general, y un orden de los $k$ colores, podemos determinar en $O(n^2k\log n)$ tiempo, usando $O(nk^2)$ espacio, la orientación y posición de $k-1$ rectas paralelas que permiten separar a los puntos en $S$ minimizando el número de puntos mal clasificados de acuerdo al orden dado.*

## 2.6. Otro enfoque

En esta sección describiremos un paradigma distinto para mantener actualizada dinámicamente la subsecuencia creciente más larga, ésto al ir generando las distintas permutaciones en la secuencia circular correspondiente al conjunto $S$. La diferencia principal con respecto al enfoque mostrado en la Sección 2.4, es que aquí podemos mejorar ligeramente la complejidad para obtener una solución al problema de separación débil ordenada con bandas; no obstante hay varios casos a considerar.

Definamos primeramente algunos conceptos que usaremos para nuestra descripción. Sea $\mathfrak{a} = (a_1, a_2, \ldots, a_n)$ una permutación de longitud $n$ en la que aparecen $k$ enteros distintos y sea $\mathfrak{a}'$ una subsecuencia creciente más larga de $\mathfrak{a}$. Decimos que $\mathfrak{a}'$ *induce una partición* de $\mathfrak{a}$ en $k$ bloques de la siguiente manera. Sea $j$ la posición en $\mathfrak{a}$ del primer elemento $i$ en $\mathfrak{a}'$, con $1 < i \leq k$ y $1 \leq j \leq n$. El bloque $\mathfrak{a}_{i-1}$ termina en la posición $j-1$, mientras que el bloque $\mathfrak{a}_i$ comienza en la posición $j$.

Sea $\mathfrak{a}_i$ el $i$-ésimo bloque de $\mathfrak{a}$ inducido por $\mathfrak{a}'$, $1 \leq i \leq k$. Supongamos que $\mathfrak{a}_i$ comienza en la posición $j$ y termina en la posición $j+m-1$, con $1 \leq m \leq n$. A los elementos que ocupan estas dos posiciones los denominamos como *elementos extremo*, mientras que diremos que $s$ es un *elemento interno* de $\mathfrak{a}_i$ si es que $s$ no está en la posición $j$ ni en la posición $j+m-1$. Así mismo, decimos que $s$ es un *elemento propio* del bloque $\mathfrak{a}_i$ si es que $s = i$. Observemos entonces que un bloque inducido $\mathfrak{a}_i$, con $2 \leq i \leq k$, inicia siempre con un elemento propio del bloque.

Para fines de este nuevo enfoque haremos uso de un preprocesamiento, el cual nos ayuda a organizar cierta información de una permutación $\mathfrak{a}$ que será de utilidad más adelante. Dada $\mathfrak{a}$, construimos un árbol binario de búsqueda AVL por cada uno de los enteros distintos $i$ que $\mathfrak{a}$ posee, con $1 \leq i \leq k$. La llave de cada nodo para el $i$-ésimo árbol es la posición que guarda un elemento $i$ en $\mathfrak{a}$. Se sabe que un árbol de este tipo, representando a una permutación de $m$ elementos distintos, puede construirse en $O(m\log m)$ tiempo [35]. Por tanto, este preprocesamiento toma $O(kn\log n)$ tiempo y usa $O(n)$ espacio.

El siguiente resultado muestra cuándo podemos asegurar que dos permutaciones consecutivas en la secuencia circular, asociada a nuestro conjunto de puntos, tienen la misma subsecuencia creciente más larga.

**Lema 1.** *Sea $\mathfrak{a}$ una permutación de longitud $n$ con $k$ elementos distintos, y sea $\mathfrak{a}'$ una subsecuencia creciente más larga de $\mathfrak{a}$. Sean $a_j$ y $a_{j+1}$ dos elementos consecutivos e internos de un bloque $\mathfrak{a}_i$.*

*Si* $\mathfrak{b}$ *representa la permutación que resulta de intercambiar de posición los elementos* $a_j$ *y* $a_{j+1}$*, entonces* $\mathfrak{a}'$ *es también una subsecuencia creciente más larga de* $\mathfrak{b}$.

*Demostración*. 3 casos son posibles:

- $a_j$ y $a_{j+1}$ son elementos propios. Esto significa que ambos elementos pertenecen a $\mathfrak{a}'$ y por tanto, la subsecuencia creciente más larga sigue siendo la misma para $\mathfrak{b}$.
- Sólo uno de los dos elementos es propio. El elemento propio, después del intercambio, sigue perteneciendo a $\mathfrak{a}'$, mientras que el otro elemento continúa siendo interno y no propio, por lo que la longitud de la subsecuencia creciente más larga no aumenta. Entonces, $\mathfrak{a}$ sigue siendo válida para $\mathfrak{b}$.
- Ambos elementos no son propios del bloque. Esto quiere decir que ambos elementos no están en $\mathfrak{a}'$, además después del intercambio, ambos elementos conservan sus características de internos y no propios. Ésto no permite incrementar la longitud de la subsecuencia creciente más larga en $\mathfrak{b}$ y por tanto, se sigue el resultado.

■

Ahora, podemos establecer una relación entre dos bloques consecutivos de una permutación:

**Lema 2.** *Sean* $\mathfrak{a}_l$ *y* $\mathfrak{a}_m$ *dos bloques consecutivos de* $\mathfrak{a}$ *inducidos por su subsecuencia creciente más larga* $\mathfrak{a}'$*. En estos bloques se cumple, respectivamente, lo siguiente:*

- *Sean* $l_1$ *y* $l_2$ *los dos últimos elementos propios del bloque* $\mathfrak{a}_l$*. Entonces, existe a lo más una ocurrencia de un elemento m entre* $l_1$ *y* $l_2$.
- *Sean* $m_1$ *y* $m_2$ *los dos primeros elementos propios del bloque* $\mathfrak{a}_m$*. Entonces, existe a lo más una ocurrencia de un elemento l entre* $m_1$ *y* $m_2$.

*Demostración*. Veamos el primer caso. Supongamos que hay más de un elemento $m$ entre $l_1$ y $l_2$. En particular, $l_2$ es un elemento propio del bloque $\mathfrak{a}_l$, por lo que éste pertenece a $\mathfrak{a}'$. Si eliminamos $l_2$ de $\mathfrak{a}'$, entonces su longitud disminuiría en una unidad, pero si además agregaramos las ocurrencias de los elementos $m$ que se encuentran entra las posiciones de $l_1$ y $l_2$, entonces la longitud de esta nueva subsecuencia creciente aumentaría en al menos dos unidades, por lo que estaríamos generando una subsecuencia creciente de longitud mayor a la de $\mathfrak{a}'$, lo cual es una contradicción.

El otro caso se demuestra de manera análoga. ■

Con estos dos resultados en mente, los únicos casos en los que nos interesa fijarnos son aquellos en los que uno o ambos elementos que permutan son extremos de bloque. Así, sean $l$ y $r$ dos elementos adyacentes en la permutación $\mathfrak{a}$, que al intercambiar de posición, originan la permutación $\mathfrak{b}$. Hay que tener en mente que cada vez que dos elementos permutan, necesitamos actualizar

a lo más dos de los árboles AVL que construimos en el preprocesamiento, cada uno en $O(\log n)$ tiempo.

La subsecuencia creciente más larga de $\mathfrak{a}$ se ve afectada de acuerdo a alguno de los siguientes casos:

**Caso 1.** $l$ y $r$ son elementos propios y extremos.

Si se da alguno de los casos descritos en el Lema 2, digamos el primero, entonces podemos generar una subsecuencia creciente más larga para $\mathfrak{b}$ de la siguiente manera. Eliminamos $l$ de $\mathfrak{a}'$ y en su lugar agregamos la aparición del elemento $r$ que aparece antes del elemento que acabamos de omitir. Este elemento $r$ se encuentra en el intervalo comprendido entre las posiciones de los dos últimos elementos propios del bloque $\mathfrak{a}_l$, y podemos hallar su posición en $O(\log n)$ tiempo al hacer uso del correspondiente árbol AVL.

Si dicho elemento $r$ existe, la longitud de esta nueva subsecuencia creciente más larga $\mathfrak{b}'$ es igual que la de $\mathfrak{a}'$. La inclusión de un nuevo elemento $r$ ocasiona que los bloques inducidos por $\mathfrak{b}'$ en $\mathfrak{b}$ sean distintos a los que induce $\mathfrak{a}'$ en $\mathfrak{a}$. Para actualizarlos, lo único que tenemos que hacer es trasladar el inicio del bloque $\mathfrak{a}_r$ hasta la posición del elemento $r$ que acabamos de introducir; ver Figura 2.6.

$$\mathfrak{a} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ \underline{1}\ 2\ \overset{l}{\underline{1}}\ \overset{r}{\underline{2}}\ 3\ \underline{2}\ 3\ \underline{2}\ 1\ \underline{3})$$

$$\mathfrak{b} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ \underline{1}\ \underline{2\ 2}\ 1\ 3\ \underline{2}\ 3\ \underline{2}\ 1\ \underline{3})$$

Figura 2.6: Dos elementos propios permutan y la longitud de la subsecuencia creciente más larga se mantiene.

En dado caso que no se presente una instancia descrita en el Lema 2, entonces la longitud de $\mathfrak{b}'$ sería una unidad menor que la de $\mathfrak{a}'$, pues sólo podríamos mantener como propio a uno de los elementos que permutan, mientras que el otro tendría que eliminarse de $\mathfrak{a}'$. Sin pérdida de generalidad, supongamos que el elemento que conservamos es el menor; ver Figura 2.7. La nueva posición del bloque $S_l$ puede calcularse también en $O(\log n)$ tiempo.

$$\mathfrak{a} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ 3\ \underline{1\ 1}\ \overset{l}{\underline{2}}\ \overset{r}{3}\ \underline{2}\ 3\ \underline{2}\ 1\ \underline{3})$$

$$\mathfrak{b} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ 3\ \underline{1}\ 2\ \underline{1}\ 3\ \underline{2}\ 3\ \underline{2}\ 1\ \underline{3})$$

Figura 2.7: Dos elementos propios permutan, pero la longitud de la subsecuencia creciente más larga disminuye.

**Caso 2.** $l$ y $r$ son elementos extremos, pero sólo $r$ es un elemento propio de bloque.

Supongamos que $\mathfrak{a}_m$ es el siguiente bloque a la derecha de $\mathfrak{a}_r$. Si $l \neq m$, entonces $\mathfrak{a}'$ sigue siendo una subsecuencia creciente más larga para $\mathfrak{b}$ (Figura 2.8) y el nuevo inicio del bloque $\mathfrak{a}_r$ está una posición a la izquierda, lo cual puede actualizarse claramente en tiempo constante.

$$\mathfrak{a} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ \underline{1\ 1}\ \overset{l}{4}\ \overset{r}{\underline{2}}\ 3\ \underline{2}\ \underline{3}\ 2\ 1\ \underline{4})$$

$$\mathfrak{b} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ \underline{1}\ \underline{1}\ \underline{2}\ 4\ 3\ \underline{2}\ \underline{3}\ 2\ 1\ \underline{4})$$

Figura 2.8: Sólo un elemento de los que permuta es propio y $\mathfrak{a}'$ es válida para $\mathfrak{a}$ y $\mathfrak{b}$.

Veamos qué pasa para cuando $l = m$. Si el número de elementos $m$ en el bloque $\mathfrak{a}_r$ es menor al número de sus elementos propios menos uno, entonces $\mathfrak{a}'$ continúa siendo válida para $\mathfrak{b}$. En caso contrario, es decir, cuando el número de elementos propios de $\mathfrak{a}_r$ menos uno es igual al número de elementos $m$, la longitud de una subsecuencia creciente más larga para $\mathfrak{b}$ es mayor en una unidad a la longitud de $\mathfrak{a}'$. Esta subsecuencia creciente más larga para $\mathfrak{b}$ puede construirse con base en $\mathfrak{a}'$, ésto al mover el inicio del bloque $\mathfrak{a}_m$ justo a la posición que tenía $r$ antes de la permutación. Para cualesquiera dos bloques adyacentes, el saber la cantidad de elementos $m$ que hay en el bloque $\mathfrak{a}_r$ puede obtenerse al momento de construir la permutación $\mathfrak{a}$ y actualizarse en $O(1)$ tiempo al considerar el desplazamiento ya descrito; ver Figura 2.9.

$$\mathfrak{a} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ 3\ \underline{1}\ 2\ \underline{1}\ \overset{l}{3}\ \overset{r}{\underline{2}}\ 3\ \underline{2}\ 1\ \underline{3})$$

$$\mathfrak{b} = (2\ 3\ 2\ \underline{1\ 1}\ 3\ 3\ 2\ \underline{1\ 1}\ 2\ 3\ \underline{1}\ 2\ \underline{1}\ \underline{2}\ \underline{3\ 3}\ 2\ 1\ \underline{3})$$

Figura 2.9: En este segundo caso, puede suceder que la subsecuencia creciente más larga aumente en una unidad.

**Caso 3.** Sólo uno de los elementos es extremo del bloque.

Un análisis combinado de los Casos 1 y 2 permite resolver esta situación, por lo que se ha optado por omitir el análisis para este caso.

Observemos que en cualquiera de los casos anteriores, cuando ocurre una permutación, podría originarse un nuevo bloque. Ésto puede ser controlado fácilmente, pero hay que tenerlo en cuenta. Con base en este análisis de casos es que podemos establecer el último de los resultados de este capítulo.

**Teorema 2.5.** *Dado un conjunto $k$-coloreado de $n$ puntos en el plano en posición general, y un orden de los $k$ colores, podemos determinar en $O(n^2 \log n)$ tiempo, usando $O(n)$ espacio, la orientación y posición de $k-1$ rectas paralelas que permiten separar a los puntos en $S$ minimizando el número de puntos mal clasificados de acuerdo al orden dado.*

## 2.7. Comentarios finales

Una versión distinta al problema principal que se presenta en este capítulo, y que surge de manera inmediata, es la siguiente: Determinar la orientación y posición de $k-1$ rectas paralelas en el plano, las cuales permitan separar a los puntos de $S$ al minimizar el número de puntos mal clasificados, ésto independientemente de un orden de colores.

Podríamos dar una solución a este nuevo problema al usar el algoritmo presentado anteriormente. Sin embargo habría que considerar, en principio, los $k!$ ordenes de colores posibles en el conjunto $k$-coloreado, dando lugar a un algoritmo con complejidad exponencial. Entonces, una línea de trabajo pendiente sería el estudiar esta nueva versión, tanto para determinar su complejidad como para dar un algoritmo.

Por supuesto, es también de interés el poder mejorar la complejidad de nuestro algoritmo para la versión ordenada y habría, por tanto, que considerar enfoques distintos a los ya presentados.

# Cubierta para una clase con círculos

# 3

## 3.1. Introducción

En áreas como el *Aprendizaje automatizado*, la *Minería de datos* y el *Reconocimiento de patrones*, se requiere con frecuencia el seleccionar un conjunto pequeño de ejemplares que representen una colección de datos [10, 18, 21, 55]. Con el fin de trabajar con dichos datos, éstos pueden transformarse a conjuntos de puntos. Así por ejemplo, si $A$ y $C$ representan dos conjuntos ajenos de puntos en el plano, uno puede preguntarse por el polígono simple $P$ con el menor número de vértices, de tal manera que en su interior estén todos los puntos de $A$ y ninguno de $C$. Se conoce que este problema de clasificación es NP-duro [30], y en [58] se proporciona un algoritmo de aproximación logarítmica para dicho problema que corre en tiempo polinomial.

Otro problema relacionado, con el cual trabajaremos, se define formalmente a continuación. Sean $B$ y $R$ dos conjuntos disjuntos y finitos de puntos en el plano, tal que $B = \{b_1, \ldots, b_m\}$ y $R = \{r_1, \ldots, r_n\}$. A lo largo de este capítulo, nos referiremos a $B$ como el conjunto (o clase) de puntos *azules*, y diremos que $R$ es el conjunto (clase) de puntos *rojos*. El *problema de la cubierta para una clase* en particular, digamos $B$, consiste en encontrar una colección de cardinalidad mínima de círculos abiertos $C = \{C_1, \ldots, C_k\}$, denominada *cubierta*, tal que las siguientes dos condiciones se cumplan:

- todo punto azul se encuentra en el interior de algún círculo $C_i$: $B \subset \bigcup_{i=1}^{k} C_i$ , y
- ningún $C_i$ contiene un punto rojo en su interior: $R \cap (\bigcup_{i=1}^{k} C_i) = \emptyset$ ; ver Figura 3.1.

No está de más observar que este problema es asimétrico en el sentido que no es lo mismo establecer una cubierta para la clase azul que para la roja.

Una interpretación de este problema en el área de la *Ubicación de servicios* es la siguiente. Supongamos que una cadena comercial tiene que decidir la ubicación de un conjunto de servicios en una región, ésto con el objetivo de cubrir una demanda específica, representada por puntos azules, sin cubrir otra, denotada por puntos rojos. Por lo tanto, el objetivo es ubicar servicios, de

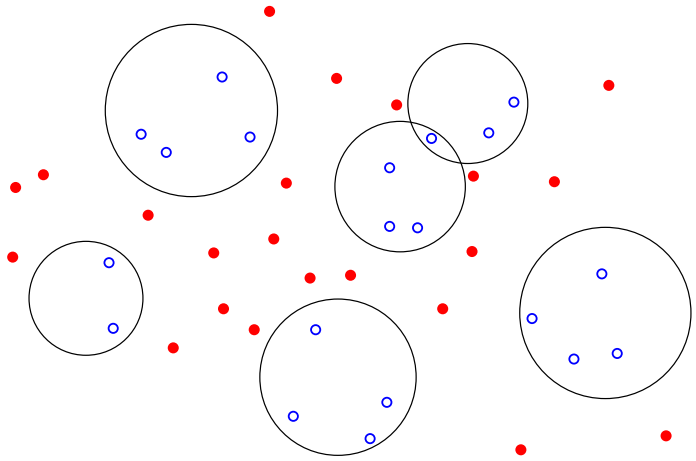

Figura 3.1: Cubierta para los puntos azules con círculos.

tal manera que el cliente azul más lejano a una de estas ubicaciones se encuentre más cerca que cualquier cliente rojo. Como se puede apreciar, lo que se desea hacer es separar los puntos azules de los rojos por medio de un criterio geométrico.

Existe diversas variantes de este problema, dependiendo de las condiciones impuestas sobre los radios y los centros de los círculos que conformarían una cubierta:

- Una cubierta *restringida* para una clase requiere que los centros de los círculos se ubiquen en puntos $b_i \in B$.
- Una cubierta para una clase se denomina *heterogenea* si no se requiere que todos los radios de los círculos sean iguales.

La versión restringida y heterogenea del problema de la cubierta para una clase ha sido la más estudiada y los primeros resultados se encuentran en [10]. Particularmente, los autores de dicho trabajo demuestran que esta versión del problema es NP-dura. En [55] se plantea un algoritmo de aproximación a esta variante utilizando un enfoque de digráficas.

En este capítulo investigamos la variante no-restringida y heterogenea del problema de la cubierta para una clase, a la cual llamaremos a partir de ahora el *problema de la cubierta para una clase con círculos arbitrarios* o el problema *CCA*. En particular, demostramos más adelante que el problema CCA es también NP-duro y además comentamos algunos resultados que pueden usarse para hablar sobre algoritmos de aproximación.

## 3.2. El problema CCA es NP-duro

Para poder demostrar que el problema CCA es NP-duro, hacemos uso de una reducción del problema 3-SAT plano. Por tal motivo, empezamos esta sección dando algunos conceptos referen-

tes a dicho problema.

Una instancia del problema 3-SAT ([38]) es una fórmula $\psi$ en forma normal conjuntiva, la cual usa $r$ variables booleanas, a las que llamaremos $x_1, \ldots, x_r$, en $s$ cláusulas, $C_1, \ldots, C_s$. Cada una de las cláusulas $C_i$ consta de tres literales, $c_{i1}, c_{i2}, c_{i3}$, donde $c_{ij}$ es $x_k$ o $\overline{x}_k$, $1 \leq i \leq s$, $1 \leq j \leq 3$, $1 \leq k \leq r$.

$$\psi = \bigwedge_{i=1}^{m} (c_{i1} \vee c_{i2} \vee c_{i3})$$

El problema 3-SAT consiste en decidir si existe una asignación de valores de verdad a las variables de $\psi$ que hagan verdadera la fórmula. Ahora, una instancia del problema 3-SAT es plana si la gráfica que representa a la fórmula $\psi$, $G(\psi) = (V, E)$ es plana. Dada la fórmula $\psi$, los vértices y las aristas de $G(\psi)$ se definen de la siguiente manera:

$$V = \{x_1, \ldots, x_r\} \cup \{C_1, \ldots, C_s\}$$
$$E = \{(x_i, C_j) \mid x_i \in C_j \ \ o \ \ \overline{x}_i \in C_j\}$$

Un ejemplo de la gráfica asociada a una fórmula $\psi$ se muestra en la Figura 3.2.

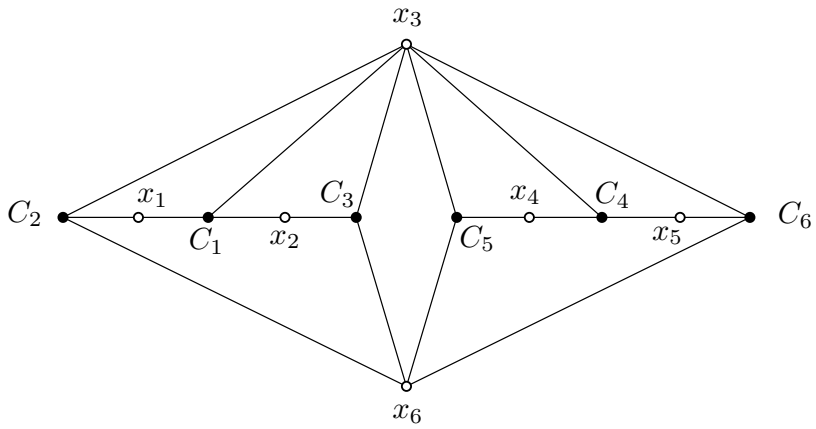


Figura 3.2: Gráfica correspondiente a la fórmula $\psi = (\overline{x}_3 \wedge x_1 \wedge x_2) \vee (\overline{x}_3 \wedge \overline{x}_1 \wedge x_6) \vee (\overline{x}_3 \wedge \overline{x}_2 \wedge x_6) \vee (\overline{x}_3 \wedge x_4 \wedge x_5) \vee (\overline{x}_3 \wedge \overline{x}_4 \wedge \overline{x}_6) \vee (\overline{x}_3 \wedge \overline{x}_5 \wedge \overline{x}_6)$.

La demostración de que el problema 3-SAT plano es NP-duro puede consultarse en [54]. Dado ésto y basándonos en una de las pruebas dadas en [2], procederemos entonces a demostrar el siguiente resultado:

**Teorema 3.1.** *El problema CCA es NP-duro.*

*Demostración.* Decimos que un polígono $P$ con vértices azules $V(P) = \{v_0, \ldots, v_{k-1}\}$ es un *polígono variable* si éste satisface las siguientes condiciones:

1. $P$ tiene un número par de vértices,
2. Las aristas de $P$ tienen longitud constante, $c_P$, y
3. La distancia entre cualquier par de vértices no adyacentes de $P$ es al menos $c_P + \epsilon$.

Dado un polígono variable $P$, definimos $D_P = \{D_0, \ldots, D_{k-1}\}$ como un conjunto de círculos, tal que $D_i$ tiene radio $\frac{c_P}{2} + \epsilon$ y cuyo centro es el punto medio de la arista de $P$ que une los puntos $v_i$ y $v_{i+1}$, $i = 0, \ldots, k-1$; la suma se hace módulo $k$ y $\epsilon$ es lo suficientemente pequeña. Podemos asociarle a $P$ un conjunto $R_P$ de $O(k)$ puntos rojos, cuya posición está lo suficientemente cerca a la unión de los $k$ círculos de $D_P$, de tal manera que cualquier otro círculo que contenga dos vértices no adyacentes de $P$, contenga al menos uno de estos puntos rojos.

Observemos que el conjunto de todos los círculos $D_i$, con $i$ par (impar respectivamente), forman una solución para el problema CCA que tiene como conjunto de puntos $V(P) \cup R_P$. Estos dos subconjuntos de círculos son las únicas soluciones y ambas hacen uso de $k$ elementos. Ahora, asignemos a $D_i$ el color $F$ si $i$ es par, en caso contrario, lo coloreamos de color $T$.

Dada una instancia $\psi$ del problema 3-SAT plano, con $r$ variables y $s$ cláusulas, obtenemos primeramente un encaje plano geométrico, $\mathcal{G}$, de la gráfica $G(\psi)$. Si un vértice de $\mathcal{G}$ representa una cláusula de $\psi$, lo coloreamos de azul.

Ahora, consideremos la estrella $S_i$, formada por un vértice de $\mathcal{G}$ que representa una variable $x_i$ de $\psi$, junto con las aristas de $\mathcal{G}$ que lo unen a los puntos que representan las cláusulas de $\psi$. Es posible reemplazar $S_i$ por un polígono variable $P(i)$, el cual usa un número par de vértices, de manera que:

- Cada $P(i)$ tiene un número polinomial (sobre $r + s$ ) de vértices, al cual denotamos como $n_i$,
- El conjunto de polígonos $P(i)$, generados por las variables $x_i$ de $\psi$, no se intersecan, $1 \leq i \leq r$,
- Ningún círculo $D$, que contenga dos vértices azules de $P_i$, contiene en su interior un vértice de otro polígono variable $P(j)$, $i \neq j$, y
- Un vértice de $\mathcal{G}$, que representa una cláusula $C_j$ de $\psi$ en la cual aparece la literal $x_i$ ($\overline{x_i}$ respect.), pertenece exactamente a la frontera de un círculo en $D_{P(i)}$, y dicho círculo tiene color $T$ ($F$ respectivamente).

Definimos como $B = \{\{C_1, \ldots, C_s\} \cup V(P(1)) \cup \ldots \cup V(P(r))\}$ y a $R = \{R_{P(1)} \cup \ldots \cup R_{P(r)}\}$ (Figura 3.3).

Supongamos que existe una asignación de valores de verdad para las variables de la fórmula $\psi$, de tal manera que al evaluar $\psi$ con estos valores, la fórmula se satisface. Sin pérdida de generalidad, supongamos que $x_1, \ldots, x_\ell$ son verdaderas, mientras que $x_{\ell+1}, \ldots, x_r$ son falsas. Entonces, la unión del subconjunto de círculos de $C_{P(i)}$, coloreados de color $T$, junto con el subconjunto de círculos

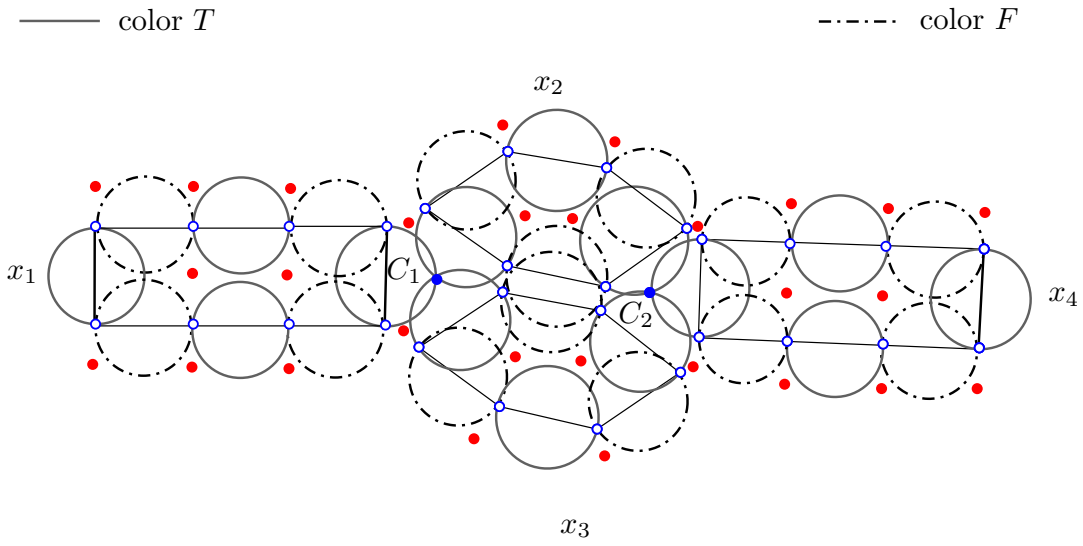


Figura 3.3: Instancia del problema CCA generada a partir de la fórmula $\psi = (x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_2 \vee \overline{x}_3 \vee x_4)$.

de $C_{P(j)}$, coloreados de color $F$, $1 \leq i \leq \ell < j \leq r$, forman una solución al problema CCA para el conjunto $B \cup R$, la cual contiene exactamente $\sum_{i=1}^{r} \frac{n_i}{2}$ círculos.

La solución propuesta es óptima, además si para $\psi$ no existe una asignación de valores de verdad que la satisfagan, entonces cualquier solución para el problema CCA para $B \cup R$ necesitaría más de $\sum_{i=1}^{r} \frac{n_i}{2}$ elementos. Por lo tanto, el resultado queda demostrado.

■

## 3.3. Algoritmos de aproximación

En esta sección mencionaremos algunos resultados de aproximación respecto al problema CCA, ésto como complemento a la sección anterior en donde se mostró que dicho problema es NP-duro.

Sea $S$ un conjunto de puntos en posición general en el plano, tal que no existen 4 puntos cocirculares. Una *triangulación* de $S$ es un conjunto de triángulos $\mathcal{T} = \{T_1, \ldots, T_t\}$ con interiores disjuntos, tal que los vértices de cualquier triángulo en $\mathcal{T}$ están en $S$, $T_1 \cup \ldots \cup T_t$ es el cierre convexo de $S$ y ningún punto en $S$ está en el interior de algún triángulo de $\mathcal{T}$.

La *triangulación Delaunay* de $S$, $\mathcal{DT}(S) = \{T_1, \ldots, T_t\}$, es una triangulación de $S$, tal que el círculo $D_i$ definido por los vértices de cualquier triángulo $T_i$ en $\mathcal{DT}(S)$ no contiene puntos de $S$ en su interior. A este tipo de círculos los llamaremos *círculos de Delaunay* de $S$. Ahora, las aristas de los triángulos de $\mathcal{DT}(S)$ son llamadas *aristas de Delaunay*. Se sabe que si un círculo $D$ contiene a dos puntos $p, q \in S$ en su frontrera, y en su interior no contiene puntos de $S$, entonces el segmento

de recta que une a $p$ con $q$ es una arista de Delaunay de $S$. A cualquier círculo $D$ como éste lo llamaremos *círculo semi-Delaunay*.

Sea $B \cup R$ un conjunto bicromático de puntos, tal que $|B| = m$ y $|R| = n$. En lo que resta de esta sección, asumiremos que $B \cup R$ está en posición general y además, que no hay cuatro puntos cocirculares. Esta suposición nos asegura que $\mathcal{DT}(R)$ está bien definida. Podemos hacer entonces la siguiente observación:

**Observación 2.** Sea $C = \{C_1, C_2, \ldots, C_k\}$ un conjunto de círculos que forma una solución al problema CCA para $B \cup R$. Es posible sustituir cada círculo $C_i$ por otro círculo $C_i'$, tal que todo punto azul en el interior de $C_i$ también está en el interior de $C_i'$, además $C_i'$ contiene al menos dos puntos rojos en su frontera, es decir, $C_i$ puede sustituirse por un círculo semi-Delaunay.

Definimos como $C^*$ al conjunto de todos los círculos de Delaunay de $R$ unión los círculos semi-Delaunay de $R$ que tienen un punto azul en la frontera (Figura 3.4). Entonces, cualquier solución al problema CCA es un subconjunto de $C^*$. Notemos que $|C^*| = O(mn)$.

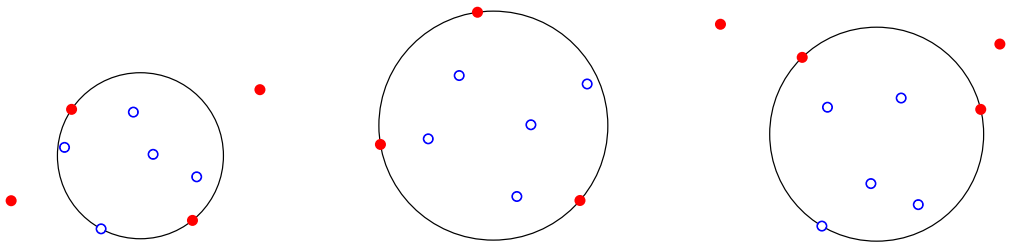

Figura 3.4: Diferentes tipos de círculos en $C^*$.

Cabe mencionar que en los resultados mencionados a continuación, se hace uso de las mismas técnicas que en [7], artículo en el que también se plantea un problema de cubierta para una clase, aunque a través del mínimo número de rectángulos abiertos con lados paralelos a los ejes coordenados. Por tal motivo, nuestros resultados de aproximación obtenidos son similares.

Sea $X$ un conjunto finito de objetos y sea $\mathcal{R}$ una colección finita de subconjuntos de $X$; diremos que la pareja $(X, \mathcal{R})$ forma un *sistema finito de conjuntos*.

Definimos $\mathcal{R}_x$ como el conjunto de todos los subconjuntos en $\mathcal{R}$ que contienen al elemento $x \in X$. Entonces, dado un sistema (primal) finito de conjuntos $(X, \mathcal{R})$, se puede asociar un sistema *dual* de conjuntos [9], $(\mathcal{R}, X^*)$, donde $X^* = \{\mathcal{R}_x \mid x \in X\}$.

Dado el sistema de conjuntos $(X, \mathcal{R})$, el *problema set cover*[1] [38] busca el subconjunto $D \subset \mathcal{R}$ de cardinalidad mínima tal que todo elemento de $X$ esté en algún elemento de $D$, es decir, $D$ *cubre* a $X$. El problema dual de set cover es el *problema del hitting set*[1], en el cual se requiere de un conjunto $P \subseteq X$ de cardinalidad mínima tal que la intersección de $P$ con cada elemento de $\mathcal{R}$ sea distinta del vacío. Observemos entonces que el problema del set cover en el sistema primal se convierte en el problema del hitting set en el sistema dual y viceversa.

[1] Ya que no existe una traducción estándar para este término, decidimos conservar la escritura en inglés.

El problema del set cover es NP-duro y el mejor factor de aproximación que se conoce, dado por un algoritmo polinomial, es de $\ln|X|+1$ [29, 38]. Dicho algoritmo sigue un paradigma glotón, es decir, mientras haya elementos en $X$ que no estén cubiertos, se agrega a la solución el conjunto en $\mathcal{R}$ que contenga el máximo número de elementos de $X$ no cubiertos.

Ahora bien, el problema CCA es una instancia del problema set cover en el sistema $(B, C^*)$. Por tanto, el algoritmo glotón que mencionamos nos proporciona el mismo factor logarítmico de aproximación para el problema CCA:

**Proposición 3.** *Un algoritmo con paradigma "glotón" proporciona una solución con aproximación de $O(\log m)$ para el problema CCA.*

Brönnimann y Goodrich [9] proporcionaron un método general para encontrar una solución aproximada para el problema hitting set en sistemas de conjuntos. Este método se aplica a espacios de conjuntos con *dimensión VC* finita [9, 44, 69] y se basa en encontrar, como candidatos a hitting set, conjuntos de cardinalidad pequeña llamados *redes-$\varepsilon$*.

En términos de nuestro problema, para un sistema primal de conjuntos, una red-$\varepsilon$, $0 \leq \varepsilon \leq 1$, es un subconjunto $B' \subseteq B$ tal que cualquier círculo en $C^*$ que contiene $\varepsilon|B|$ puntos, cubre un elemento de $B'$. Si hablamos de un sistema dual, una red-$\varepsilon$ es un subconjunto $C \subseteq C^*$ que cubre todo punto de $B$, y tal que un punto $p \in B$ es cubierto por por al menos $\varepsilon|C^*|$ círculos de $C^*$.

La dimensión VC de un sistema $(X, \mathcal{R})$ se define como la cardinalidad máxima de un subconjunto $Y \subseteq X$ tal que cualquier subconjunto de $Y$ es la intersección de $Y$ con algún elemento de $\mathcal{R}$. Si la dimensión VC de un sistema es $d$, entonces la dimensión VC del sistema dual asociado es a lo más $2^{d+1}$ [9, 69]. Observemos que la dimensión VC de nuestro sistema $(B, C^*)$ es a lo más tres, ya que en cualquier subconjunto $W \subseteq B$ con al menos cuatro puntos, existe un subconjunto $W' \subset W$ que no puede separarse de $W \setminus W'$ con un círculo en $C^*$; ver Figura 3.5.


Figura 3.5: Los conjuntos $\{\alpha, \beta, \gamma\}$ y $\{\alpha, \beta\}$, respectivamente, no pueden separarse de su complemento por ningún círculo en $C^*$.

Dicho método de Brönnimann y Goodrich para sistemas con dimensión VC finita, reporta un hitting set de tamaño a lo más un factor de $O(\log c)$ con respecto del tamaño óptimo $c$, y este hitting set induce una cubierta de conjuntos para el problema CCA con el mismo tamaño, es decir,

**Proposición 4.** *Para el problema CCA existe un algoritmo de aproximación de $O(\log c)$, donde $c$ es el tamaño de una cubierta óptima.*

El resultado de Brönnimann y Goodrich [9] se basa en el hecho de que, para cada sistema de conjuntos con dimensión VC finita $d$, existe una red-$\varepsilon$ de tamaño $O\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$ [44]. En general, si el sistema de conjuntos tiene dimensión VC constante, y existe una red-$\varepsilon$ de tamaño $O\left(\frac{1}{\varepsilon}\phi\left(\frac{1}{\varepsilon}\right)\right)$, su método encuentra un hitting set de tamaño $O(\phi(c)\,c)$, donde $c$ es el tamaño de una solución óptima.

Por otro lado, Matoušek et al. [56] probaron la existencia de redes-$\varepsilon$ de tamaño $O(\frac{1}{\varepsilon})$ para sistemas $(X,\mathcal{R})$ donde $\mathcal{R}$ son subconjuntos de puntos o discos. Debido a ésto, y sumando la técnica de Brönnimann y Goodrich, podemos establecer lo siguiente:

**Proposición 5.** *Existe un algoritmo de aproximación de $O(1)$ para el problema CCA.*

Hay que resaltar que fuera del algoritmo de aproximación glotón para el problema CCA, los demás resultados son del tipo teórico existencial, por lo que los pasos de los algoritmos que permitirían obtener las aproximaciones mencionadas no están totalmente definidos.

## 3.4. Comentarios finales

Claramente, el problema de establecer un algoritmo que proporcione una solución como la especificada en la Proposición 5 permanece abierto, aunque probablemente éste sea un problema bastante desafiante.

Hay que mencionar que la demostración de NP-dureza que presentamos en este capítulo, para la versión no-restringida y heterogenea del problema de la cubierta para una clase, bien podría adaptarse para cuando requerimos que los círculos que conformen la cubierta sean todos del mismo radio. No obstante hay que revisar con cuidado algunos detalles respecto a la realizabilidad del conjunto bicoloreado propuesto con base a los polígonos que representan las variables involucradas en una fórmula dada.

Otra línea de investigación por explorar sería el considerar patrones geométricos diferentes al círculo, para así obtener resultados como los presentados aquí y en [7], donde se consideran rectángulos con lados paralelos a los ejes coordenados.

# Islas óptimas 4

## 4.1. Introducción

Desde hace dos décadas aproximadamente, se han estudiado cada vez más problemas geométricos y algorítmicos en conjuntos bicromáticos de puntos en el plano. Uno de los primeros problemas que se abordaron refieren a la existencia de trayectorias simples y alternantes de color, cuyos vértices son precisamente puntos de un conjunto bicoloreado [3]. Otro problema, por ejemplo, se estudia en [68], donde se requiere encontrar dos árboles generadores monocromáticos, uno por cada color en el conjunto de puntos, de tal manera que las intersecciones entre ellos sean pocas.

Denotemos como $S$ a un conjunto de puntos en el plano, los cuales están en posición general. Es de conocimiento general que el *cierre convexo* de $S$, $CH(S)$, es el conjunto convexo más pequeño en el plano que contiene a $S$. A lo largo de este capítulo, los elementos de $S$ estarán nuevamente divididos en dos clases, cada una identificada por un color: puntos rojos y puntos azules.

Un concepto que se usará con frecuencia en este capítulo es el siguiente. Sea $I$ un subconjunto de puntos de $S$. Diremos que $I$ forma una *isla* de $S$ si $CH(I) \cap S = I$ (Figura 4.1). Así también, diremos que una isla de $S$ es monocromática si todos sus elementos tienen asignado el mismo color. Dado ésto, una isla monocromática de $S$ podrá ser llamada roja o azul, dependiendo del color de sus elementos.

Muchos de los problemas en conjuntos bicromátcos de puntos pueden describirse como problemas de partición. En dichos problemas, se requiere dividir al conjunto $S$ en conjuntos de islas disjuntas con propiedades particulares. Por ejemplo, supongamos que un conjunto de puntos $S$ contiene $kn$ puntos rojos y $n$ puntos azules. Se desea dividir $S$ en un conjunto de $n$ islas disjuntas, cada una conteniendo $k$ puntos rojos y sólo un azul, y cuya unión sea précisamente $S$. Una recopilación importante de este tipo de problemas puede consultarse en [52] .

El problema que estudiaremos en este capítulo puede describirse de la siguiente manera: encontrar la *isla monocromática más grande* que posee un conjunto bicromático de puntos $S$, esto es, encontrar la isla monocromática de mayor cardinalidad; ver Figura 4.2. A lo largo de este capítulo

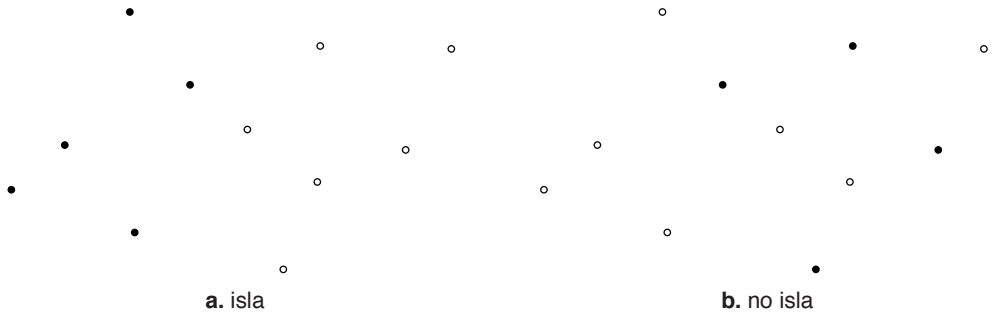


Figura 4.1: Los puntos negros en (a) forman una isla en el conjunto, mientras que en (b) no es así.

nos referiremos a este problema como el *problema IMG*, para el cual se logró establecer un algoritmo que en $O(n^3)$ tiempo, usando $O(n^2)$ espacio, nos permite dar solución a dicho problema. Este algoritmo representa una mejora a un resultado previo [31], cuyo algoritmo propuesto toma $O(n^3 \log n)$ tiempo para resolver la misma problemática.

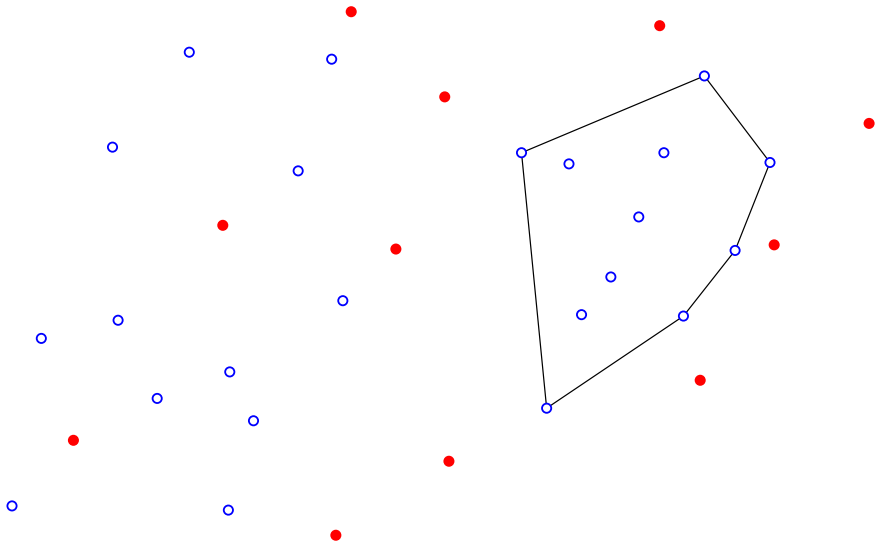

Figura 4.2: Isla azul de mayor cardinalidad en el conjunto de puntos bicromático.

Más adelante también veremos que, haciendo pequeñas modificaciones al algoritmo principal propuesto en este capítulo, se pueden resolver versiones del problema *IMG* en las que se involucran puntos con pesos (usualmente valores enteros). En estas versiones, el objetivo es encontrar islas de $S$ con máximo peso. Observemos que cuando las etiquetas o pesos asociados a los puntos de $S$ se escogen cuidadosamente, podemos resolver problemas que aparentemente no tienen relación alguna. Por ejemplo, encontrar una isla de $S$ con máxima *discrepancia*, esto es, una isla en la que el valor absoluto de la diferencia entre el número de puntos rojos y azules se maximiza [23], se puede obtener al resolver dos instancias del problema de máximo peso: Primero, asignamos peso 1 ($-1$) a todo punto azul (rojo) y encontramos una isla con máximo peso. Segundo, asignamos

peso $-1$ (1) a todo punto azul (rojo) de $S$ y encontramos una isla con máximo peso. La solución con máximo peso entre los dos problemas es en realidad la isla de $S$ con máxima discrepancia.

La motivación para estudiar este problema proviene de aplicaciones en áreas como Minería de Datos, Agrupamiento Estadístico (Statistical Clustering), Reconocimiento de Patrones o Compresión de Datos. Así por ejemplo, en Minería de Datos y problemas de clasificación, un método natural para el análisis de la información es el seleccionar prototipos que representen diferentes clases de datos. Una técnica común para hacer dicha selección es el análisis de cúmulos sobre los datos [24], los cuales pueden obtenerse al usar figuras geométricas simples, como en [25], donde se consideran círculos y rectángulos con lados paralelos a los ejes. En el trabajo descrito en este capítulo, particularmente, se hace uso de polígonos convexos. En el área de Reconocimiento de Patrones, los cierres convexos se han considerado para medir la separación entre clases [53]. De hecho, se ha estudiado también la relación entre los cierres convexos y otros objetos, como las máquinas de vectores soporte (SVMs) [5].

## 4.2. La isla azul más grande

En esta sección, describiremos un algoritmo de $O(n^3)$ tiempo que nos permite encontrar, en un conjunto bicromático de puntos $S$, la isla azul más grande. Al ejecutar una segunda vez el algoritmo, pero considerando el color rojo, se podrá determinar la solución del problema *IMG*. Entonces, sin pérdida de generalidad de ahora en adelante, supondremos que la isla monocromática más grande en $S$ es de color azul. Cabe mencionar que nuestro método se basa en el algoritmo propuesto por Fischer en [31] y cuya complejidad de tiempo es del $O(n^3 \log n)$.

### 4.2.1. Muchas soluciones

Antes de comenzar con el algoritmo, observemos que existen configuraciones de puntos rojos y azules que poseen un número exponencial de soluciones al problema *IMG*. Para ejemplificarlo, tomemos la siguiente construcción: sea $P_k$ un polígono regular de $k$ lados y cuyos vértices son $v_1, v_2, \ldots, v_k$. Para cada vértice $v_i$ de $P_k$, coloquemos un conjunto $S_i$ de $2k$ puntos sobre una curva convexa $C$, $k$ rojos y $k$ azules, de tal manera que puntos adyacentes sean de distinto color, como se muestra en la Figura 4.3.

Observemos que cualquier isla azul más grande en el conjunto de puntos $S = S_1 \cup \ldots \cup S_k$, tiene exactamente $k$ elementos. Más aún, estas islas pueden obtenerse ya sea al seleccionar un punto azul de cada subconjunto $S_i$, o al seleccionar todos los puntos azules de algún $S_i$. Haciendo el conteo correspondiente, podemos ver que existen $k^k + k$ islas azules con dichas características.

### 4.2.2. Algoritmo

Pasemos ahora a definir algunos conceptos y terminología que nos serán útiles al describir el algoritmo. Asumiremos sin pérdida de generalidad, de ahora en adelante, que cualesquiera dos

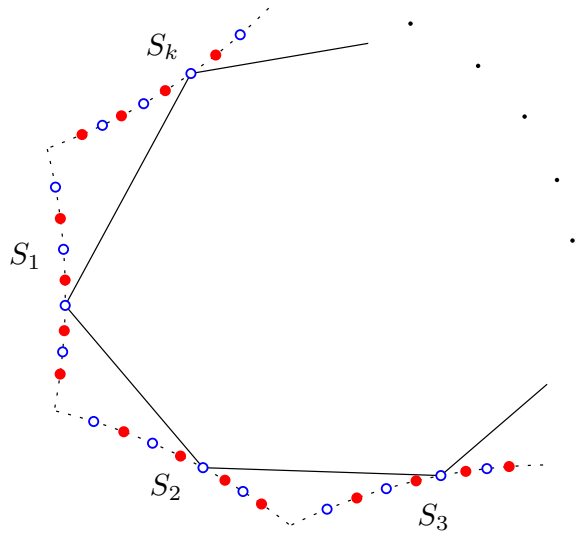


Figura 4.3: Una instancia del problema *IMG* que consiste de $k$ grupos de $2k$ puntos cada uno. La instancia tiene $k^k + k$ soluciones y los puntos unidos por segmentos de recta forman una de ellas.

puntos en $S$ tienen distinta coordenada $y$. Denotemos como $pq$ al segmento de recta que une a los puntos $p$ y $q$. Ahora, dado un segmento de recta $pq$ y un punto $r$ fuera del segmento, $\triangle(r, pq)$ denotará al triángulo cuyos vértices son los puntos $p$, $q$ y $r$. Así también, dado un conjunto de puntos $X$, $Az(X)$ representa el número de puntos azules en $X$. Entonces, $Az(\triangle(r, pq))$ denota el número de puntos en $\triangle(r, pq)$.

Sea $p$ un punto y $e = qr$, $e' = rs$ dos segmentos, cuyos extremos son puntos azules. Decimos que $e$ y $e'$ son *p-compatibles* si se dan las siguientes tres condiciones (Figura 4.4):

- $\triangle(p, e)$ y $\triangle(p, e')$ no contienen puntos rojos en su interior,
- $\triangle(p, e)$ y $\triangle(p, e')$ tienen interiores disjuntos, y
- $\triangle(p, e) \cup \triangle(p, e')$ forman un polígono convexo

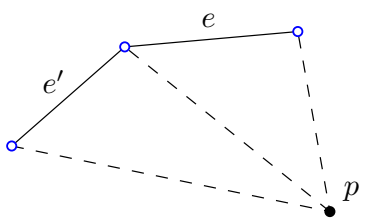


Figura 4.4: Dos segmentos de recta *p-compatibles*.

Sea $\mathcal{P}$ un polígono convexo con vértices en $S$ , y sea $p$ el vértice de $\mathcal{P}$ con ordenada más grande. Diremos, indistintamente, que $\mathcal{P}$ tiene *ancla* en $p$ o que $\mathcal{P}$ está anclado en $p$. La idea base para el

algoritmo será el encontrar, para cada punto azul $p \in S$, la isla azul más grande $\mathcal{B}$ de $S$ tal que el polígono convexo determinado por su cierre convexo esté anclado en $p$, es decir, buscaremos la *isla azul más grande en $S$ anclada en $p$*.

Supongamos que $\mathcal{B}$ es una isla azul anclada en un punto $p$. Asumamos también que los vértices de $CH(\mathcal{B})$ están etiquetados como $p, p_{\sigma_1}, \ldots, p_{\sigma_k}$, ésto en el sentido contrario a las manecillas del reloj a lo largo de la frontera de $CH(\mathcal{B})$. Dado lo anterior, diremos que $\mathcal{B}$ termina en el segmento $p_{\sigma_{k-1}} p_{\sigma_k}$.

Denotemos como $h_p$ a la recta horizontal que pasa a través de $p$, y sean $p_i$ y $p_j$ dos puntos azules de $S$ que están por debajo de $h_p$. Observemos que si $\triangle(p, p_i p_j)$ contiene puntos rojos de $S$ en su interior, entonces el segmento $p_i p_j$ no será nunca una arista de una isla azul anclada en $p$. Por tanto, estaremos interesados en segmentos $p_i p_j$ tales que $\triangle(p, p_i p_j)$ no contiene puntos rojos. Asociemos un peso $w(p_i p_j)$ a una arista $p_i p_j$ como sigue: $w(p_i p_j)$ es igual al valor $Az(\mathcal{B})$, donde $\mathcal{B}$ es la isla azul más grande anclada en $p$ que termina en la arista $p_i p_j$.

Entonces, encontrar la isla azul más grande en $S$, anclada en $p$, se reduce a encontrar la arista $p_i p_j$ con máximo peso. Para lograr ésto, haremos uso de la técnica de programación dinámica y la propiedad que nos permitirá implementar este paradigma se encuentra en la siguiente observación:

**Observación 3.** Sea $\mathcal{B}$ una isla azul anclada en $p$; sean $p, p_{\sigma_1}, \ldots, p_{\sigma_k}$ los vértices de la frontera de $CH(\mathcal{B})$, etiquetados en orden contrario a las manecillas del reloj. Denotamos como $\mathcal{B}^{(i)}$, $1 \leq i \leq k$, a la isla cuyos vértices de la frontera de $CH(\mathcal{B}^{(i)})$ son $p, p_{\sigma_1}, \ldots, p_{\sigma_i}$. Podemos ver que $Az(\mathcal{B})$ satisface lo siguiente:

$$Az(\mathcal{B}^{(i)}) = \begin{cases} 2 & \text{si } i = 1 \\ Az(\mathcal{B}^{(i-1)}) + Az(\triangle(p, p_{\sigma_{i-1}} p_{\sigma_i})) - 2 & \text{si } 1 < i \leq k \end{cases}$$

Notemos entonces que la arista $p_{\sigma_i} p_{\sigma_{i+1}}$ y la arista $p_{\sigma_{i+1}} p_{\sigma_{i+2}}$, ambas en $\mathcal{B}$, son compatibles, $i = 1, \ldots, k-2$.

La observación anterior nos permitirá encontrar la isla monocromática más grande, anclada en $p$, de la siguiente manera: haremos un barrido radial de los puntos azules que se encuentran por debajo de la recta $h_p$, ésto en sentido contrario a las manecillas del reloj alrededor de $p$, e iremos uniendo conjuntos de aristas $p$-compatibles, es decir, conjuntos de triángulos con vértices azules, donde uno de ellos es $p$, tal que sus interiores son disjuntos, no contienen puntos rojos, y su unión forma un polígono convexo anclado en $p$. Ver Figura 4.5.

La diferencia principal entre el algoritmo introducido en este capítulo y el presentado en [31], yace precisamente en la forma en que se decide cómo unir las aristas compatibles. El uso de la estructura de datos *prefix-maximum* en [31], no permite evitar hacer una búsqueda binaria para seleccionar la mejor solución en cada paso.

En la siguiente sección mostraremos cómo calcular la isla azul más grande anclada en $p$, en $O(n^2)$ tiempo y espacio, de tal manera que ésto nos permita establecer un algoritmo de $O(n^3)$ tiempo para resolver el problema *IMG* en $S$.

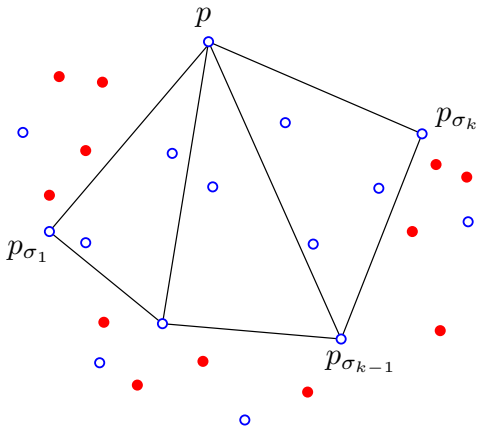


Figura 4.5: Unión de aristas compatibles para hallar la isla azul más grande anclada en el punto $p$.

### 4.2.3. Preprocesamiento

Comencemos esta sección presentando un resultado de [27], el cual nos ayudará más adelante a determinar las aristas $p$-compatibles:

**Teorema 4.1.** *Sea $P$ un conjunto de $n$ puntos en posición general en el plano. Es posible hacer un proceso previo sobre $P$ que toma $O(n^2)$ tiempo y espacio, de tal manera que para cualquier triángulo $T$ con vértices en $P$, se pueda determinar en tiempo constante el número de puntos de $P$ en $T$.*

La idea detrás de este resultado es calcular, para cualquier par de puntos $p, q \in P$, el número de puntos que yacen en la banda vertical por debajo del segmento de recta $pq$, denotado como $s[p, q]$. Así, el número de puntos que contiene un triángulo $\triangle pqr$, cuyo punto más a la izquierda es $p$ y cuyo punto más a la derecha es $r$, puede calcularse simplemente como el valor absoluto de $s[p, q] + s[q, r] - s[p, r]$ (Ver Figura 4.6).


Figura 4.6: Número de puntos en un triángulo $\triangle pqr$.

Veámos cómo calcular todos los valores posibles de $s[*, *]$. Sin pérdida de generalidad, tratemos a todos los segmentos de izquierda a derecha, de acuerdo a su extremo derecho; los segmentos

de recta con el mismo extremo derecho los ordenaremos en sentido de las manecillas del reloj. Se puede plantear entonces el algoritmo del Listado 4.1.

Listado 4.1: Número de puntos por debajo de cada segmento de recta en $P$.

```
1  Inicializar todos los elementos [*,*] a cero.
2  Ordenar los puntos de P por abcisa, de izquierda a derecha. Sea
       p_1,...,p_n dicho orden.
3  Por cada punto p_i ∈ P, ordenar todos los puntos a la izquierda de p_i,
       en sentido de las manecillas del reloj alrededor de p_i. Sean
       p_1^i, p_2^i, ..., p_{i-1}^i las secuencias obtenidas.
4  Para i = 2 hasta n
5      Para j = 2 hasta i - 1
6          Si p_j^i está a la izq. de p_{j-1}^i, entonces
7              s[p_j^i, p_i] = s[p_{j-1}^i, p_i] + s[p_j^i, p_{j-1}^i] + 1
8          Si p_j^i está a la der. de p_{j-1}^i, entonces
9              s[p_j^i, p_i] = s[p_{j-1}^i, p_i] - s[p_j^i, p_{j-1}^i]
```

Ya que los puntos $p^i_j$ se encuentran en orden radial con respecto a $p_i$ y el punto $p^i_j$ es el sucesor de $p^i_{j-1}$ en dicho orden, el triángulo $\triangle(p_i, p^i_{j-1}p^i_j)$ debe ser vacío. Por tanto, $s[p^i_j, p_i]$ se puede calcular como la suma (línea 7) o la diferencia (línea 9) de $s[p^i_{j-1}, p_i]$ y $s[p^i_j, p^i_{j-1}]$. En el primer caso (línea 7), el 1 adicional que aparece en la suma, corresponde al punto $p^i_{j-1}$. Más aún, observe que el cálculo de algún elemento $s[*,*]$ sólo hace uso de valores previamente calculados. Más detalles pueden consultarse [27].

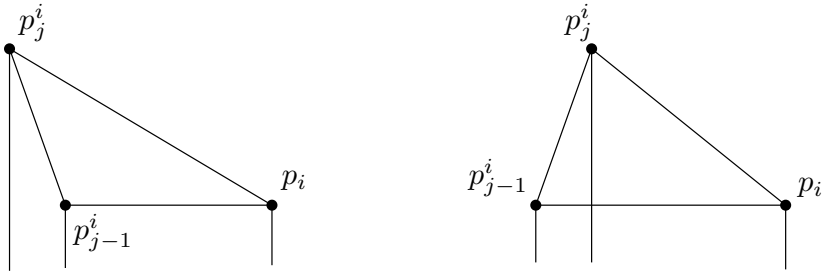


Figura 4.7: Casos presentados en el Listado 4.1, líneas 7 y 9 respectivamente.

Observemos que es posible hacer modificaciones sencillas al algoritmo presentado en el Listado 4.1, de manera que podamos resolver los siguientes problemas:

- Para cada triángulo $T$ con vértices en $P$, encontrar en tiempo constante el número de puntos rojos y azules contenidos en $T$.

- Si los elementos de $P$ tienen pesos asociados, calcular en tiempo constante la suma de los pesos de los elementos de $P$ contenidos en algún $T$.

### 4.2.4. Cálculo del peso de una arista $p_i p_j$

Sea $S_p$ el conjunto de puntos azules en $S$ que se encuentran por debajo de la recta $h_p$, la cual pasa por $p \in S$ y es paralela al eje $x$. Supongamos que los elementos de $S_p$ tienen etiquetas $p_1, \ldots, p_k$, ésto con base en el orden radial alrededor de $p$. Observemos que, usando el resultado del Teorema 4.1, podemos descartar, en tiempo constante por elemento, todas las aristas $p_i p_j$ tales que $\triangle(p, p_i p_j)$ contiene al menos un punto rojo (Figura 4.8).

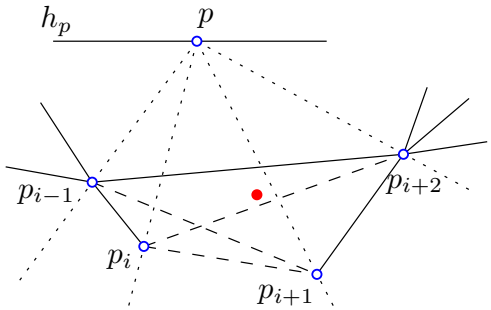


Figura 4.8: La arista $p_i p_{i+2}$ es descartada, dado que $\triangle(p, p_i p_{i+2})$ contiene un punto rojo.

Antes de describir cómo son procesadas el resto de las aristas, hagamos algunas consideraciones, las cuales pretenden hacer más transparentes las ideas planteadas. Si $i < j$, orientaremos la arista $p_i p_j$ como $p_i \to p_j$, con lo cual obtenemos una digráfica acíclica, $G_p$, cuyo conjunto de vértices son los puntos en $S_p$, $1 \leq i < j \leq k$. Por cada $1 < i \leq k$, re-etiquetemos el conjunto de aristas de entrada y salida de $p_i$, ésto como $A_i = \{a_{i,1}, \ldots, a_{i,q}\}$ y $B_i = \{b_{i,1}, \ldots, b_{i,r}\}$ respectivamente; $A_i$ y $B_i$ están ordenadas radialmente con respecto de $p_i$, como se muestra en la Figura 4.9. Para todo punto $p \in S$, el orden correspondiente de puntos en $S_p$, $p_1, \ldots, p_k$, así como los conjuntos ordenados $A_i$ y $B_i$, $i = 1, \ldots, k$, pueden obtenerse en tiempo y espacio cuadráticos [26].

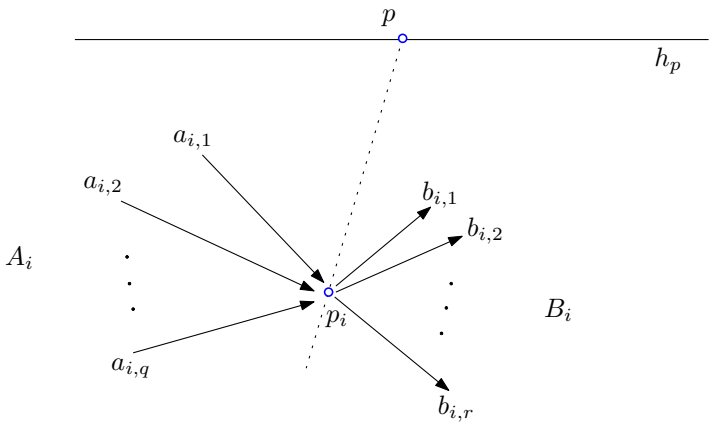


Figura 4.9: Orden de las aristas en $A_i$ y $B_i$.

Mostremos entonces cómo calcular $w(p_i p_j)$ recursivamente para todo $1 \leq i < j \leq k$. Recordemos que para calcular $w(b_{i,m})$, $1 \leq m \leq r$, debemos encontrar una arista $a_{i,s}$ que sea $p$-compatible

con $b_{i,m}$, y cuyo peso sea lo más grande posible. Para ésto, la idea será manejar las listas $A_i$ y $B_i$ de tal manera que el peso de todas las arista de salida de $p_i$, se calcule en tiempo lineal.

Asignamos a toda arista $p_i p_j$ un apuntador, el cual identificamos como prev($p_i p_j$) y cuyo valor inicial es *null*. Notemos que cada arista $p_1 p_j$ tiene asignado el peso $Az(\triangle(p, p_1 p_j))$. Supongamos ahora que todas las aristas $p_\alpha p_\beta$ ya tienen pesos asignados, $1 \leq \alpha < \beta \leq k$, $\alpha < i < k$. A continuación mostramos cómo asignar los pesos a todas las aristas $p_i p_j$, $i < j \leq k$.

Para toda $1 \leq \ell \leq q$, sea $z(\ell)$ el entero más pequeño tal que $w(a_{i,z(\ell)}) = \text{máx}\{w(a_{i,1}), \ldots, w(a_{i,\ell})\}$. Los valores $z(1), \ldots, z(q)$ pueden calcularse en $O(q)$ tiempo de la siguiente manera: $z(1) = 1$ y para $\ell = 2, \ldots, q$ podemos aplicar la siguiente fórmula,

$$z(\ell) = \begin{cases} \ell & \text{si } w(a_{i,\ell}) > w(a_{i,z(\ell-1)}) \\ z(\ell-1) & \text{si } w(a_{i,\ell}) \leq w(a_{i,z(\ell-1)}) \end{cases}$$

En este punto, es importante observar lo siguiente:

**Observación 4.** Sea $s$ el índice más grande tal que $a_{i,s}$ es $p$-compatible con $b_{i,m}$. Entonces, $a_{i,z(s)}$ es $p$-compatible con $b_{i,m}$ y tiene peso máximo sobre todas las aristas en $A_i$ que son compatibles con $b_{i,m}$.

Dado lo anterior, podemos asignar a $w(b_{i,m})$ el valor de $w(a_{i,z(s)}) + Az(\triangle(p, b_{i,m})) - 2$, mientras que prev$(b_{i,m}) = a_{i,z(s)}$.

El procedimiento general que nos permite calcular $w(\cdot)$ y $prev(\cdot)$, para toda arista $b_{i,m}$ en la lista $B_i$, es el siguiente:

Para $m = 1, \ldots, r$, encontrar el índice $s_m$ más grande tal que $a_{i,s_m}$ y $b_{i,m}$ son $p$-compatibles. Si $p_i$ no posee una arista de entrada que sea $p$-compatible con $b_{i,m}$, hacemos $s_m = 0$. Si $s_m = 0$, entonces $w(b_{i,m}) = Az(\triangle(p, b_{i,m}))$; de lo contrario $w(b_{i,m}) = w(a_{i,z(s_m)}) + Az(\triangle(p, b_{i,m})) - 2$. Dado que $s_1 \leq s_{2 \leq \ldots \leq s_r}$, se sigue que podemos calcular $w(\cdot)$ y $prev(\cdot)$ para todo elemento de $B_i$ en una sólo iteración sobre $A_i$ y $B_i$. Así, este procedimiento toma $O(n)$ tiempo, por lo que podemos establecer el resultado siguiente:

**Lema 3.** *El peso de toda arista $p_i p_j$ que yace por debajo de $h_p$, puede calcularse en $O(n^2)$ tiempo.*

Para obtener una isla azul $\mathcal{B}$ anclada en $p$, encontramos una arista $p_i p_j$ con máximo peso; para calcular el cierre convexo de $\mathcal{B}$, simplemente seguimos los apuntadores $prev(\cdot)$ recursivamente. Al repetir el procedimiento descrito previamente en todos los elementos azules de $S$ obtenemos:

**Teorema 4.2.** *Sea $S$ un conjunto bicromático de n puntos en posición general en el plano. La isla monocromática azul de mayor cardinalidad puede calcularse en $O(n^3)$ tiempo, haciendo uso de un pre-procesamiento de $O(n^2)$ tiempo y espacio.*

## 4.3. Generalizaciones

El algoritmo presentado en la sección anterior puede usarse para resolver una colección de problemas de optimización. Para ver ésto, supongamos que tenemos una función $f : \mathcal{P} \to \mathbb{R}$, donde $\mathcal{P}$ es el conjunto de polígonos convexos.

**Definición 1** ([27])**.** Decimos que una función $f : \mathcal{P} \to \mathbb{R}$ es *descomponible* si existe una función $g : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$, la cual puede calcularse en tiempo constante, tal que para cualquier polígono $P = \langle p_1, p_2, \ldots, p_k \rangle \in \mathcal{P}$ y cualquier índice $2 < i < k$,

$$f(P) = g(f(\langle p_1, \ldots, p_i \rangle), f(\langle p_1, p_i, \ldots, p_k \rangle), p_1, p_i)$$

donde $g$ puede calcularse en tiempo constante.

En términos simples, la definición anterior puede verse de la siguiente manera. Sean $P_1$ y $P_2$ dos subpolígonos de un polígono $P$, los cuales se obtienen al cortar $P$ a lo largo de una de sus diagnonales $e$, la cual une los vértices $p_1$ y $p_i$. Una función $f$ es descomponible si $f(P)$ puede calcularse en tiempo constante a partir de $f(P_1)$, $f(P_2)$ y alguna información acerca de $e$. Por ejemplo, la función $\mathcal{H}$ que cuenta el número de puntos de $S$ que están en el interior o en la frontera de un polígono convexo $P$ es descomponible, ya que $\mathcal{H}(P) = g(x, y, p_1, p_i) = x + y - 2$, donde $\mathcal{H}(P_1) = x$ y $\mathcal{H}(P_2) = y$.

Podemos enunciar el siguiente resultado:

**Teorema 4.3.** *Sea $S$ un conjunto bicromático de $n$ puntos en posición general en el plano, y sea $f$ una función monótona descomponible. El problema de calcular la isla que maximiza (minimiza) $f$ puede resolverse en $O(n^3 + G(n))$ tiempo, donde $G(n)$ es el tiempo que se requiere para calcular $f$ para el $O(n^3)$ triángulos en $S$.*

*Demostración*. El algoritmo es el mismo que el descrito en la Sección 4.2, pero con la siguiente modificación en la Observación 3:

$$f(\mathcal{B}^{(i)}) = \begin{cases} f(pp_{\sigma_i}) & \text{si } i = 1 \\ f(\mathcal{B}^{(i-1)}) + f(\triangle(p, p_{\sigma_{i-1}} p_{\sigma_i})) - f(pp_{\sigma_i}) & \text{si } 1 < i \leq k \end{cases}$$

Observemos que es esencial para que el algoritmo sea correcto el que $f$ sea una función monótona descomponible. La complejidad en tiempo se sigue naturalmente.

■

No es difícil ver que funciones como el área, el perímetro o la discrepancia son monótonas descomponibles, y que $G(n)$ puede calcularse en $O(n^3)$ tiempo para cualquiera de ellas.

Supongamos ahora que le asignamos pesos a los elementos de $S$, y para nuestros fines pensemos que dichos pesos son valores enteros. Observemos que la suma de los pesos de los puntos en el interior de un triángulo puede calcularse en tiempo constante por triángulo. Así también, la suma de los pesos es una función monótona descomponible para valores enteros. De lo anterior, la prueba del siguiente corolario se sigue.

**Corolario 1.** *Los problemas de encontrar la isla con máximo (mínimo) perímetro, área, discrepancia o peso, pueden resolverse en $O(n^3)$ tiempo.*

Concluimos esta sección mencionando algunas generalizaciones de nuestros problemas sobre conjuntos de puntos bicromáticos, ésto al considerar ahora conjuntos de puntos cuyos elementos están coloreados con $k$ colores. Consideremos que $S$ es un conjunto de puntos $k$-coloreado y sea $P$ un polígono convexo con vértices en $S$. Decimos que $P$ es un *hoyo* de $S$ si $P$ no contiene elementos de $S$ en su interior. El problema de encontrar un hoyo monocromático $P$ en $S$, ya sea con el mayor número de vértices o de máxima (mínima) área o perímetro, puede resolverse en $O(n^3)$ tiempo al hacer modificaciones simples al algoritmo presentado en la Sección 4.2: En cada punto $p \in S$, unimos únicamente aristas $p$-compatibles cuyos vértices son puntos en $S_p$ con el mismo color que el de $p$.

Un problema abierto en este escenario de $k$ colores es el siguiente: Encontrar, si existe, una isla heterocromática de $S$ con $k$ elementos, es decir, una isla cuyos $k$ puntos son todos de distinto color; claro, tomando en cuenta también alguna función de optimización. Otro problema abierto sería encontrar, si existe, una poligonal convexa y heterocromática o una poligonal monótona y heterocromática de longitud mínima.

No obstante, si restringimos que los elementos de la poligonal convexa, o de la poligonal monótona, aparezcan en orden predeterminado, entonces sí se puede decir algo al respecto. Supongamos que se desea que los colores aparezcan en orden $1, 2, \ldots, k$; la poligonal monótona y heterocromática (con orden) de menor longitud puede calcularse en $O(n \log^2 n)$ tiempo ([20]), mientras que la poligonal convexa puede calcularse en $O(n^3)$ tiempo al hacer uso de nuestro algoritmo: La idea sería ahora localizar aristas $p$-compatibles que sigan el orden de colores y que consideren la optimización deseada.

## 4.4. El problema de cubierta de clase con islas

En esta sección proponemos usar el algoritmo que calcula la isla monocromática más grande, ésto con el fin de aproximar la región que abarca una clase por medio de una colección de conjuntos convexos. Varios objetos geométricos, como rectángulos o círculos, se han usado en áreas como *Aprendizaje Automático* y *Reconocimiento de Patrones* para separar un conjunto bicromático de puntos [23, 24, 25]. No obstante, como se señala en [53], el número de cierres convexos que se necesitan para aproximar la región de una clase es menor, por ejemplo, que el número de rectángulos necesarios para aproximar la misma región. Así, el uso de los cierres convexos de un conjunto de islas puede ser más adecuado para aproximar la región de una clase, en vez de hacerlo con un conjunto de rectángulos.

Dado un conjunto $S$ de dos clases distintas, digamos puntos rojos y azules, se puede plantear un problema de clasificación, el cual se conoce comúnmente como el *problema de cubierta de clases*. En este problema se requiere encontrar un pequeño número de conjuntos que cubran (contengan) puntos de una clase sin cubrir ningún punto de la otra. El problema se introdujo en [10] y en él se busca un conjunto de cardinalidad mínima $C$ de círculos del mismo radio, de tal manera que ningún elemento de $C$ contenga punto rojo alguno y que todo punto azul en $S$ se encuentre contenido en al menos un elemento de $C$.

En este capítulo consideramos una variante del problema de la cubierta de clase, la cual denominamos como el *problema de la cubierta de clase con polígonos convexos* o *problema CCPC*. En esta versión requerimos cubrir, de igual manera, los puntos azules del conjunto pero usando el mínimo número de polígonos convexos no necesariamente disjuntos. Primero que nada, mostraremos la complejidad de este problema y posteriormente propondremos una aproximación.

Demostraremos la NP-dureza de nuestro problema al hacer una reducción a partir del problema 3-SAT plano [38], el cuál también es NP-duro. Tomamos ventaja de la construcción hecha en [2], la cual se usa para probar la NP-dureza del *problema de la partición plana y bicromática*, el cual se define de la siguiente manera: Dados $n$ puntos rojos y $m$ puntos azules en el plano, encontrar el mínimo número de triángulos, disjuntos dos a dos, tal que cada punto azul esté cubierto por algún triángulo y ningún punto rojo se encuentre en el interior de algún triángulo.

Esencialmente, los autores de [2] toman cualquier instancia del problema 3-SAT plano y la transforman en una gráfica plana encajada en el plano. A partir de dicha gráfica se construye un conjunto de puntos rojos y azules, de tal manera que una solución al problema de la partición plana y bicromática existe en este conjunto de puntos si y sólo si existe una solución a la instancia inicial del problema 3-SAT plano.

En particular, el conjunto de puntos rojos y azules que se construye en [2] tiene la siguiente propiedad: Cualquier polígono convexo que contiene sólo puntos azules en su interior, contiene a lo más tres puntos, por lo que un polígono como éstos puede remplazarse por un triángulo. Entonces, cualquier algoritmo que resuelva el problema de la cubierta de clase con polígonos convexos resolverá también el problema de la partición plana y bicromática. Observemos que, aunque la solución para el problema de la partición plana y bicromática consiste de un conjunto de triángulos ajenos dos a dos (polígonos convexos), la reducción sigue siendo válida. Por tanto, podemos establecer el siguiente resultado:

**Teorema 4.4.** *El problema de la cubierta de clases con polígonos convexos es NP-duro.*

A continuación mostraremos un algoritmo de aproximación basado en el problema $IMG$. Notemos que podemos asumir que los vértices de los polígonos son puntos del conjunto dado, es decir, los polígonos convexos son los cierres de los subconjuntos de puntos azules. Así, la idea clave para nuestra aproximación es observar que nuestro problema es una instancia del problema *geométrico de cubierta de un conjunto* (set cover) [38]. El enfoque glotón de este problema, el cual permite generar una aproximación de $O(\log n)$ con respecto a su solución óptima, puede ser aplicado al problema CCPC de la siguiente manera: Calcular recursivamente la isla azul de mayor cardinalidad de $S$, removiéndola en cada paso; repitimos lo anterior hasta que no haya más puntos

azules en $S$; ver Figura 4.10.

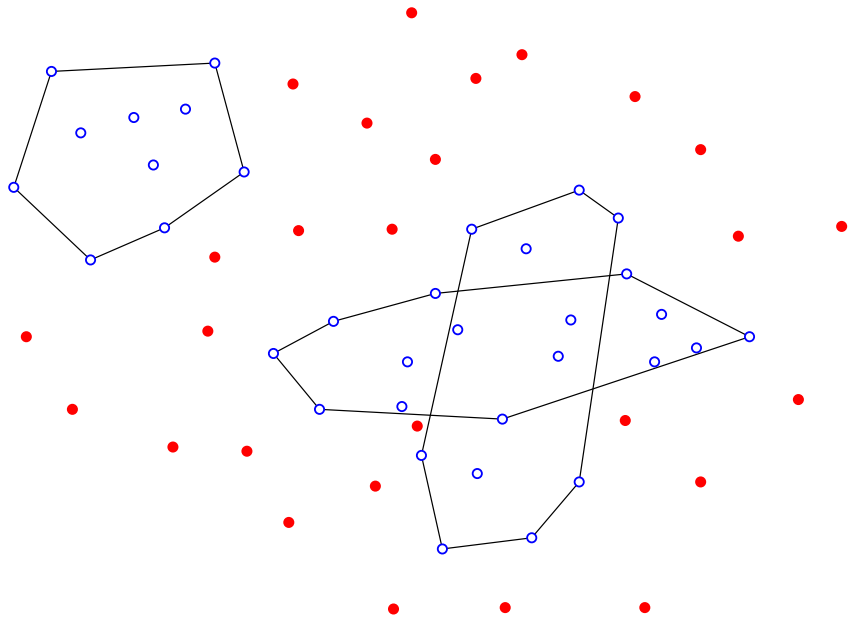

Figura 4.10: Enfoque glotón para cubrir la clase azul por medio de polígonos convexos.

Podemos establecer entonces el siguiente teorema, basados en el proceso glotón descrito hace un momento.

**Teorema 4.5.** *Existe un algoritmo de $O(n^4)$ tiempo que porduce una solución aproximada con factor $O(\log n)$ para el problema de la cubierta de clases con polígonos convexos.*

*Demostración.* Observemos que los cierres convexos de las islas azules que se obtienen del proceso glotón pueden intersecarse. De hecho, este proceso requiere $O(n)$ iteraciones en el peor de los casos, lo cual ocurre cuando todas las islas resultantes tienen cardinalidad constante. Por tanto, el algoritmo glotón requiere $O(n^4)$ tiempo en total para producir una colección de conjuntos convexos, la cual tiene cardinalidad de un factor de $O(\log n)$ con respecto de la solución óptima.

■

Comentemos como último punto que en aplicaciones dentro de áreas como la Minería de Datos y el Aprendizaje Automático, el objetivo principal es separar puntos azules de rojos. Entonces, es posible que sí se consideren como parte de una solución islas que se traslapen y por tanto se utilize el método descrito anteriormente. En otras aplicaciones, de áreas como la Visualización o la Graficación por Computadora, el cálculo de piezas disjuntas es de interés.

Un método glotón similar podría ser usado para aproximar la región de una clase al usar conjuntos convexos disjuntos. Desafortunadamente, la condición de que las piezas sean disjuntas dos a dos hace imposible el aplicar dicho procedimiento glotón para la cubierta de un conjunto,

como se establece en [51]. Más aún, podemos mostrar que un procedimiento glotón podría dar una solución con cardinalidad arbitrariamente grande comparada con la obtenida cuando se permite que las islas se intersecten. Por ejemplo, sea $S$ el conjunto con $2k$ puntos ($k$ rojos y $k$ azules) como se muestra en la Figura 4.11. Claramente, los $k$ puntos azules, con $k \geq 4$, pueden cubrirse con a lo más tres islas que se intersectan (dos islas si $k$ es par). Sin embargo, si requerimos islas azules con cierres convexos ajenos, no podemos cubrir los $k$ puntos azules con menos de $\left\lfloor \frac{k}{3} \right\rfloor + 1$ islas disjuntas.

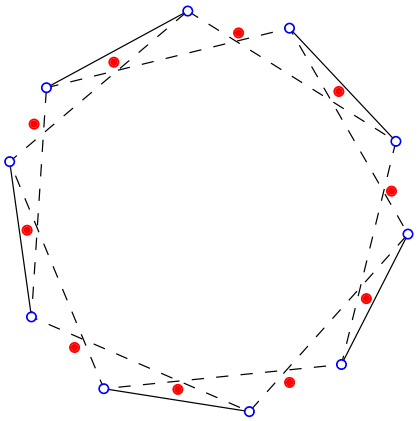

Figura 4.11: Islas disjuntas (líneas solidas) e islas no disjuntas (punteadas) para cubrir a los puntos azules.

## 4.5. Comentarios finales

Los resultados presentados en este capítulo nos permiten establecer algunos problemas abiertos. El más obvio sería reducir el tiempo de ejecución del algoritmo para nuestro problema base, el de calcular una isla monocromática maximal.

Una segunda pregunta abierta es el encontrar, si existe, una isla heterocromática en un conjunto con $k$ colores, es decir, una isla con $k$ puntos tal que todos sus elementos tengan distinto color. Hasta donde es de nuestro conocimiento al escribir esta tesis doctoral, no se conoce de complejidad alguna respecto de este problema. Para el problema de la cubierta de clase con islas, un problema práctico interesante es diseñar cuidadosamente métodos de aproximación bastante eficientes (de tiempo polinomial con bajo grado). Más aún, el encontrar heurísticas eficientes con un buen desempeño parece un problema difícil para la mayoría de los problemas de cubierta de clase y requiere de más trabajo en el futuro.

Finalmente, otra dirección interesante para el trabajo a futuro es el de poder extender nuestros resultados a espacios con dimensión más alta.

# 5 $L$-corredor $k$-cromático

## 5.1. Introducción

Los problemas en los que se trabaja con conjuntos de puntos de más de dos colores han sido de interés desde hace ya varias décadas. No obstante, el enfoque que busca extraer un subconjunto de puntos que contenga cada uno de los colores, bajo cierto criterio, es relativamente nuevo. Así por ejemplo, se ha planteado el problema de encontrar una poligonal de longitud mínima que visite todos los colores. Se sabe que este problema es $NP$-duro, ya que es una generalización del problema del agente viajero (TSP). En [20], sin embargo, se muestra que al considerar ciertas restricciones sobre la poligonal, el problema puede ser resuelto en tiempo polinomial.

Sea $S$ un conjunto de $n$ puntos en posición general en el plano. Diremos que $S$ es un conjunto *$k$-cromático*, o *$k$-coloreado*, si cada uno de sus elementos tiene asociado un único color, digamos $i$, con $1 \leq i \leq k$. Dado $p = (a, b)$ un punto en en el plano, le asociamos el siguiente conjunto $Q_p = \{(x, y) \ : \ a \leq x, \ b \leq y\}$. Dado un punto $q = (c, d)$ en el interior de $Q_p$, definimos el *$L$-corredor* $L(p, q)$ de la siguiente manera:

$$L(p,q) = \{\,(x,y) \ : \ a \leq x, \ b \leq y \leq d\,\} \cup \{\,(x,y) \ : \ a \leq x \leq c, \ b \leq y\,\}$$

A lo largo de este capítulo, llamaremos a $w_x = c - a$ el *$x$-ancho* del $L$-corredor $L(p, q)$, mientras que $w_y = d - b$ denotará su *$y$-ancho*. Cuando $w_x = w_y = w$, diremos que $L(p, q)$ tiene ancho $w$, en cuyo caso $L(p, q)$ será denotado como $L_w(p)$.

El subconjunto $C_i \subset S$ que contiene todos los puntos de color $i$ es llamado la $i$-ésima clase cromática de $S$. Diremos que un $L$-corredor $L(p, q)$ es $k$-cromático, si $L(p, q) \cap S$ contiene al menos un punto de cada $C_i$, $1 \leq i \leq k$; véase Figura 5.1.

En este capítulo estudiamos como problema principal el encontrar un $L$-corredor $k$-cromático de mínimo ancho, para el cual proporcionamos un algoritmo de $O(n^2)$ tiempo que nos permite encontrar una solución. Este algoritmo representa una mejora a un resultado previo que presentamos en [4], donde el algoritmo propuesto toma $O(n^2 \log k)$ tiempo para dar una solución.

Así también, es de nuestro interés un $L$-corredor $k$-cromático que minimize la suma de su

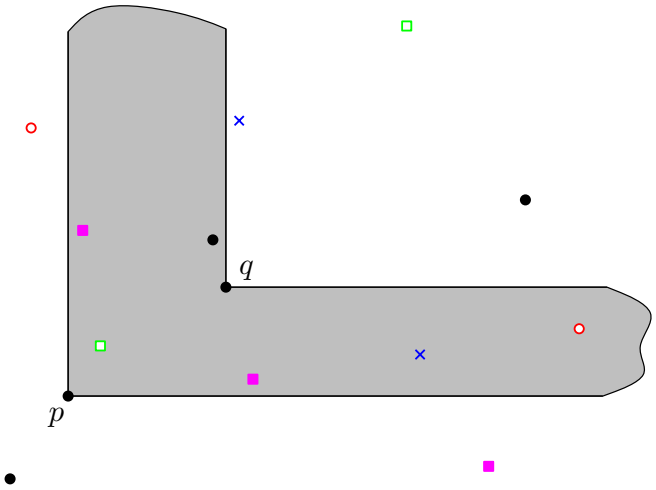


Figura 5.1: $L$-corredor $k$-cromático inducido por los puntos $p$ y $q$.

$x$- y su $y$-ancho. Para encontrar un corredor solución con estas características, establecemos un algoritmo de $O(n^2k)$ tiempo.

Otra versión que estudiaremos al final es el encontrar un $L$-corredor $k$-cromático de mínimo ancho bajo rotaciones del plano. Siendo más precisos, dado un ángulo $\theta$, con $0 \leq \theta < 2\pi$, llamamos $S_\theta$ al conjunto de puntos al que se mapea $S$ cuando rotamos el plano $\theta$ grados alrededor del orígen. Para una $\theta$ en particular, denotamos como $L_\theta(p)$ al $L$-corredor $k$-cromático de mínimo ancho en $S_\theta$, mientras que $w_\theta$ denota el ancho de $L_\theta(p)$. Planteamos un algoritmo de $O(n^3\alpha(n)\,\alpha(k)\log k)$ tiempo para encontrar un ángulo $\theta$ tal que $w_\theta$ es mínimo, donde $\alpha(\cdot)$ denota la función inversa de Ackerman.

La motivación original de los problemas bajo el paradigma de separación $k$-cromática proviene de situaciones que se presentan, principalmente, en el área de la *Ubicación de Servicios*. Supongamos que existen $k$ tipos de servicios distintos en una población, como mercados, farmacias, escuelas. Éstos pueden ser modelados como un conjunto $k$-coloreado de $n$ puntos en el plano. Un criterio, que puede ser importante, para seleccionar la ubicación de un nuevo fraccionamiento de viviendas, es que se tenga al *alcance* en el posible vecindario al menos un representante de cada uno de los servicios disponibles. Cabe hacer notar que, el uso de diferentes objetos geométricos que separan al menos $k$ puntos, uno de cada color, es lo que da sentidos distintos a que un servicio esté al alcance.

## 5.2. Mínimo ancho

En esta sección, nos dedicaremos a describir el algoritmo que en $O(n^2)$ tiempo nos permite solucionar el problema de encontrar un $L$-corredor $k$-cromático de mínimo ancho, ésto dentro de un conjunto $S$ que está $k$-coloreado.

Comencemos dando algo de notación que nos será de utilidad en el resto del capítulo. Consideremos un punto $p = (a, b)$ y el conjunto $Q_p$. La *frontera* de $Q_p$, denotada como $\partial Q_p$, es el conjunto de puntos $(x, y) \in Q_p$ tal que $x = a$ o $y = b$. Como consecuencia, la frontera de un $L$-corredor $L(p, q)$ es la unión de $\partial Q_p$ y $\partial Q_q$. Los conjuntos $\partial Q_p$ y $\partial Q_q$ serán llamados, respectivamente, la *frontera exterior* e *interior* de $L(p, q)$.

Diremos que un punto $r \in S$ en la frontera de $L(p, q)$ es *esencial*, con respecto de $L(p, q)$, si no existe otro punto de $S$ en el interior de $L(p, q)$ con el mismo color que el de $r$. En el resto de esta sección, asumiremos que el $x$- y el $y$-ancho de todo $L$-corredor considerado son iguales.

El siguiente lema caracteriza una solución para nuestro problema.

**Lema 4.** *Sea $S \subset \mathbb{R}^2$ un conjunto $k$-coloreado de puntos en el plano. Existe un $L$-corredor $k$-cromático de mínimo ancho, $L_w(p)$, que satisface las siguientes propiedades:*

*i) Cada uno de los rayos que forma parte de la frontera exterior de $L_w(p)$ contiene un punto esencial, y*

*ii) hay un punto esencial en la frontera interior de $L_w(p)$.*

*Demostración.* Sea $L_w(p)$ un $L$-corredor $k$-cromático de mínimo ancho de $S$, tal que $p = (a, b)$. Supongamos que ningún punto de $S$ yace en el rayo horizontal $r'$ de la frontera exterior de $L_w(p)$. Entonces, podemos trasladar $p$ verticalmente y hacia arriba hasta que $r'$ toque un elemento $p'$ de $S$. No es difícil ver que el $L_w(p)$ inducido sigue siendo un objeto $k$-cromático de $S$ y con ancho $w$.

Si $p'$ es un punto esencial de $L_w(p)$, entonces permanecemos ahí, ya que hemos encontrado un $L$-corredor con un punto esencial sobre $r'$. Si $p'$ no es un punto esencial, continuamos desplazando $p$ hacia arriba e ignoramos los puntos en $S$ que toquen $r'$ y que no sean esenciales de $L_w(p)$. Observemos que eventualmente, $r'$ pasará por un punto de $S$ que sea esencial. De manera similar, podemos probar que podemos trasladar $L_w(p)$ horizontalmente y hacia la derecha, hasta que el rayo vertical de su frontera exterior contenga un punto esencial con respecto de $L_w(p)$.

De manera directa podemos ver que si la frontera interior de $L_w(p)$ no contiene un punto esencial, entonces al trasladar dicha frontera hacia $p$, podemos obtener un $L$-corredor $k$-cromático con un ancho menor a $w$, lo cual implicaría una contradicción.

■

Dado un punto $p = (a, b)$ en el plano, tomemos un punto $r = (f, g)$ en $Q_p$. Definimos $d_x(r, Q_p) = f - a$ y $d_y(r, Q_p) = g - b$, a la cuales llamaremos, respectivamente, la $x$- y $y$-*distancia* del punto $r$ a la frontera de $Q_p$. Así, definimos también la *distancia de r a la frontera de* $Q_p$ como mín $\{ d_x(r, Q_p),\ d_y(r, Q_p) \}$ y la denotaremos como $d(r, Q_p)$.

Tomemos un punto en el plano $p$, tal que cada conjunto $C_i' = C_i \cap Q_p$ es distinto del vacío, con $1 \leq i \leq k$. Podemos mostrar de manera sencilla cómo encontrar en tiempo lineal el mínimo ancho $w$ tal que $L_w(p)$ es $k$-cromático. Para cada $C_i'$, encontramos el elemento $m_i \in C_i'$ que minimiza su

distancia, $d_i$, a $\partial Q_p$. Notemos que si $w = max\{d_1, d_2, \ldots, d_k\}$, $L_w(p)$ es el $L$-corredor $k$-cromático de mínimo ancho contenido en $Q_p$.

Dado lo anterior, podemos obtener facilmente un algoritmo de $O(n^3)$ tiempo para encontrar un $L$-corredor $k$-cromático en $S$. Para cada par de puntos $p_i, p_j \in S$, sea $p$ el punto de intersección de la línea horizontal que pasa a través de $p_i$ con la línea vertical que pasa por $p_j$, o la recta vertical que pasa por $p_i$ y la recta horizontal que pasa por $p_j$, de tal forma que $p_i$ y $p_j$ pertenecen a $Q_p$. Entonces, como primer paso, en tiempo lineal podemos averiguar si $Q_p$ contiene al menos un elemento de cada clase cromática en $S$, y en caso afirmativo en tiempo lineal encontramos el mínimo $w$ tal que $L_w(p)$ es $k$-cromático; en caso contrario, simplemente ignoramos el conjunto $Q_p$. Al repetir este proceso por cada par de puntos en $S$ encontraremos el $L$-corredor deseado.

Nuestra intención a continuación será el describir cómo podemos generar y procesar todos los conjuntos $Q_p$ inducidos por pares de puntos en $S$, de tal manera que en $O(n^2)$ tiempo podamos encontrar una solución a nuestro problema.

Dado $p \in \mathbb{R}^2$, retomemos nuestros conjuntos $C'_i = C_i \cap Q_p$, con $1 \leq i \leq k$. Recordemos que si para alguna $i$, $C'_i = \emptyset$, $Q_p$ no contiene un $L$-corredor $k$-cromático. Supongamos entonces que $C_i \neq \emptyset$ para toda $i$. Sea $p_{\sigma(i)}$ el punto en $C'_i$ con mínima $x$-distancia a $Q_p$, mientras que $p_{\rho(i)}$ representa al punto en $C'_i$ con mínima $y$-distancia a $Q_p$. Para cada conjunto $Q_p$ que podemos inducir, mantendremos las listas $V_p = \{p_{\sigma(1)} \ldots, p_{\sigma(k)}\}$ y $H_p = \{p_{\rho(1)}, \ldots, p_{\rho(k)}\}$. Para cada $C'_i$, si $d_x(p_{\sigma(i)}, Q_p) < d_y(p_{\rho(i)}, Q_p)$, entonces $m_i = p_{\sigma(i)}$, de lo contrario $m_i = p_{\rho(i)}$. Mantendremos también la lista $M_p = \{m_1, \ldots, m_k\}$.

### 5.2.1. Número discreto de conjuntos $Q_p$

Sea $u = (a, b)$ un punto en $S$ y $S_u$ el subconjunto de puntos de $S$ contenidos en el semiplano cerrado inducido por la linea horizontal que pasa por $u$. Supongamos que los puntos en $S_u$ están etiquetados como $p_1, p_2, \ldots, p_t$ tal que si $i < j$, entonces la coordenada $x$ de $p_i$ es mayor que la de $p_j$. Notemos que el punto $u$ es algún $p_i$, $1 \leq i \leq t$.

Barramos el plano de derecha a izquierda con una recta vertical $\ell$, y detengamonos en cada punto $p_i = (x_i, y_i)$ en $S_u$. En cada una de estas paradas, podemos determinar un conjunto $Q_p$, tal que $p = (x_i, b)$. Por supuesto, para cada uno de estos conjuntos $Q_p$, podemos inducir sus listas $H_p$, $V_p$ y $M_p$ correspondientes. Observemos que para dos puntos consecutivos $p_i$ y $p_{i+1}$, sus respectivas listas $M_p$ pueden ser drasticamente distintas; ver Figura 5.2.

De hecho, las listas $H_p$, $V_p$ y $M_p$ se ven afectadas debido a dos tipos de eventos, los cuales pueden ocurrir cada vez que nos detenemos en un nuevo punto $p_i \in S_u$, con color $j$, durante el barrido con la recta vertical $\ell$:

**Tipo I:** Cuando $\ell$ llega a $p_i$, este punto se convierte en el punto más cercano de color $j$ a la frontera del conjunto inducido $Q_p$, por lo que debe estar en $M_p$. Claramente, $p_i$ debe estar también en $V_p$ y posiblemente en $H_p$.

**Tipo II:** Supongamos que cuando $\ell$ se detiene en $p_{i-1}$, el punto de color $t \neq j$ en $V_p \cap M_p$ es $p_s$.

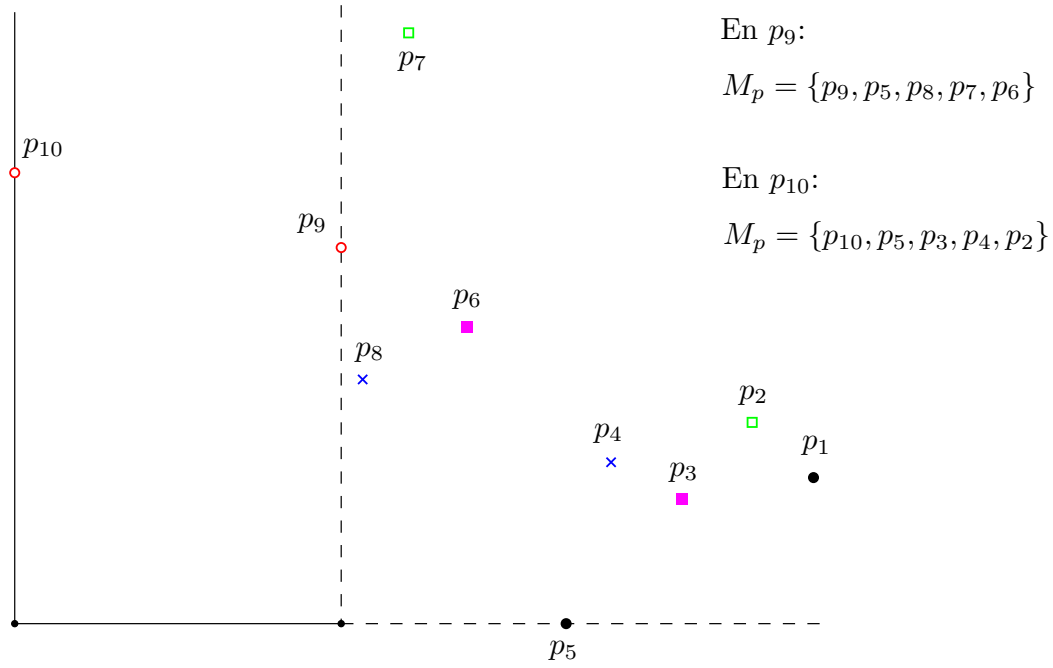


Figura 5.2: $M_p$ difiere drasticamente de $p_9$ a $p_{10}$.

Sea también $p_r$ el punto en $H_p$ con color $t$, con $r \leq s < i$, y supongamos que $d_y(p_r, Q_p) = d$. El punto $p_s$ no será parte de $V_p \cap M_p$, en la parada $p_i$, si $x_s - x_i > d$. Notemos que este tipo de evento ocurre debido a que $d_x(p_i, Q_p)$ se vuelve cada vez más grande conforme $\ell$ se desplaza a la izquierda en el barrido, mientras que $d_y(p_i, Q_p)$ permanece igual. Así, asociamos al *punto real* $p_s$ un *punto artificial* de color $t$, $p'_s = (x_s - d, y_r)$, el cual representa el momento exacto donde un evento de este tipo se sucita.

Diremos que un punto artificial $p'_i$ de color $j$ está *inactivo* si hay un punto real $p_r \neq p_i$, también de color $j$, tal que $x'_i < x_r < x_i$; en caso contrario diremos que $p'_i$ está *activo*. Entonces, a causa de estos dos tipos de eventos, si el barrido de recta considera los puntos reales y los puntos artificiales activos, podemos observar lo siguiente:

**Observación 5.** Sean $p_1, p_2, \ldots, p_t$ los puntos reales y artificiales activos en $S_u$, etiquetados en orden decreciente de acuerdo a su coordenada $x$. Entonces, las listas $H_p$, $V_p$ y $M_p$ en $p_i$ difieren a lo más en un elemento de su configuración en el punto $p_{i+1}$.

Ahora podemos demostrar el resultado siguiente:

**Lema 5.** *Dado $u = (a, b) \in S$, sea $R = \{p_1, p_2, \ldots, p_t\}$ la lista de puntos en $S_u$ ordenados decrecientemente por su coordenada $x$. Es posible construir en $O(n(\log n + k))$ tiempo lo siguiente:*

*a) La lista A de puntos artificiales asociados a los puntos reales en $S_u$, ordenados por su coordenada $x$; particularmente los que son activos.*

*b) Para cada punto real o artificial activo, sus listas $V_p$ y $H_p$, cuyos elementos están ordenados por su coordenada $x$ y su coordenada $y$, respectivamente.*

*c) Si existe, el L-corredor k-cromático de mínimo ancho, $L_w(p)$, tal que u es un punto esencial en su frontera exterior.*

*Demostración*. Mostraremos que las estructuras de los incisos a) y b) pueden obtenerse conforme iteramos sobre los elementos de $R$. Entonces, supongamos que en $p_i = (x_i, y_i) \in R$, la lista $A$ que contiene los puntos artificiales $p'_l = (x'_l, y'_l)$ asociados a los puntos $p_l$, con $l \leq i$, han sido ya calculados; los puntos en dicha lista están ordenados por coordenada $x$. $k$ de los elementos de $A$ están identificados por apuntadores $\alpha(j), 1 \leq j \leq k$, los cuales representan el último punto de color $j$ que fue insertado. Supongamos también que ya tenemos las listas $H_p$ y $V_p$ de cada punto real y cada punto activo con coordenada $x$ mayor o igual que $x_i$.

Sea $p_A = (e, f)$ el punto activo en $A$ con la coordenada $x$ más grande, tal que $e \leq x_i$. Si $x_{i+1} \leq e$, entonces podemos construir las listas $H_p$ y $V_p$, asociadas a $p_A$ sin ningún problema, como veremos un poco más adelante.

Por el contrario, si $e < x_{i+1}$, además de construir para $p_{i+1} \in C_j$ sus listas $H_p$ y $V_p$, necesitamos considerar un poco de trabajo extra. El punto $\alpha(j) = (c, d)$ se marca como inactivo si $c \leq x_{i+1}$. El punto artificial asociado a $p_{i+1}$ se calcula al consultar el punto de color $j$ en la lista $H_p$ de $p_i$, ya que éste representa el punto de dicho color con mínima $y$-distancia en el conjunto inducido $Q_p$ en $p_i$; este nuevo punto artificial se marca como activo y es insertado en $A$ en $O(\log n)$ tiempo.

Con el fin de construir las listas $H_p$y $V_p$ para un punto de color $j$, en cualquiera de los dos casos anteriores, hacemos lo siguiente. Sea $s = (g, m)$ el punto de color $j$ en la lista $H_p$ asociada a $p_i$. Debido a la Observación 5, para el caso cuando vemos el punto real $p_{i+1}$, podemos hacer una copia de la lista $H_p$ asociada a $p_i$ en $O(k)$ tiempo. Si $y_i < m$, entonces eliminamos $s$ de la copia e insertamos $p_{i+1}$ en $O(\log k)$ tiempo. Dicha lista resultante es precisamente la lista $H_p$ de $p_{i+1}$; además podemos asegurar que $p_{i+1}$ o $s$ no participan en $H_p \cap M_p$. Ahora, con respecto a la lista $V_p$ para $p_{i+1}$, una vez más podemos hacer una copia de la lista $V_p$ de $p_i$, eliminamos su punto de color $j$ e insertamos $p_{i+1}$ como el nuevo punto de dicho color que participa en $V_p \cap M_p$.

Si el punto $p_A$ es considerado, en lugar de eliminar $s$ del duplicado de la lista $H_p$ de $p_i$, sólo lo marcamos como miembro de $H_p \cap M_p$. Entonces, podemos asignar dicha lista $H_p$ a $p_A$. Para su lista $V_p$, simplemente copiamos la lista $V_p$ de $p_i$ y se la asignamos a $p_A$, asegurando que su elemento de color $j$ no participa en $V_p \cap M_p$.

Después de que hemos construido ambas listas, $H_p$ y $V_p$, nos movemos hacia adelante en $A$ para buscar el siguiente elemento activo, que será el próximo elemento $p_A$ a considerar. Notemos que una vez que un elemento en esta lista es procesado, nunca es considerado otra vez.

El tiempo total para construir todas las listas $H_p$ y $V_p$, para cada punto real o punto artificial activo, toma $O(nk)$ tiempo.

Sólo resta mostrar cómo podemos generar el $L$-corredor $k$-cromático $L_w(p)$ de mínimo ancho, tal que $u$ es un punto esencial que se encuentra en su frontera exterior. Una vez que hemos generado

las listas $H_p$ y $V_p$ para un punto, real o activo, podemos usar un *max heap* para representar cada una de estas listas. El heap que representa a $H_p$ nos proporciona el elemento en $H_p \cap M_p$ con la coordenada $y$ más grande, ésto en $O(\log k)$ tiempo. Lo mismo puede ser hecho para $V_p$, pero en este caso obtenemos el elemento en $V_p \cap M_p$ con la coordenada $x$ más grande. El máximo entre estos dos valores arrojados por los heaps, llamado *max*, se asocia también al punto en turno.

Así, al final, podemos iterar nuevamente sobre los puntos de $R$, y mantenemos el punto cuyo valor de max es el más pequeño, ya que éste garantiza que hemos encontrado el $L$-corredor con mínimo ancho. Esta nueva iteración toma $O(n)$ tiempo, por lo tanto el resultado se sigue.

■

El uso del Lema 5 en cada punto de $S$ implica inmediatamente un algoritmo de $O(n^2(\log n+k))$ tiempo para obtener una solución a nuestro problema. No obstante, un análisis más cuidadoso puede llevarse a cabo para lograr una nueva mejora. Sean $u = (u_x, u_y)$ y $v = (v_x, v_y)$ dos puntos consecutivos en $S$ de acuerdo a su coordenada $y$, y supongamos que $v$ es el punto con la ordenada más grande. Al aplicar el Lema 5 sobre $u$, podemos disponer de todas las listas. Mostraremos a continuación cómo podemos actualizar éstas estructuras para obtener una solución en $S_v = S_u \backslash \{u\}$.

Notemos que los puntos reales en $S_u$ y $S_v$ son los mismos, excepto por supuesto por $u$; lo que necesitamos considerar con cuidado es cómo se afecta la lista de puntos artificiales. Sea $A_u$ la lista de puntos artificiales en $S_u$ y sea $d = v_y - u_y$. Podemos observar lo siguiente:

**Observación 6.** Dos situaciones surgen cuando nos movemos de $u \in C_j$ hacia $v$:

1. Dado un punto artificial en $A_u$, cuyo color es distinto de $j$, podemos obtener su nueva posición en la lista de puntos artificiales $A_v$, de $S_v$, al incrementar su coordenada $x$ en $d$ unidades. Más aún, todos los puntos artificiales de color distinto a $j$, preservan el mismo orden relativo al que tienen en $A_u$; los puntos con estatus de activo en $A_u$, conservan su condición en $A_v$.

2. Todos los puntos artificiales en $A_u$ de color $j$ y coordenada $x$ más pequeña que $u_x$, dependen de $u$. Entonces, sus coordenadas $x$ necesitan ser recalculadas de manera particular.

Dado lo anterior, estamos listos para demostrar lo siguiente:

**Lema 6.** *Sean $u \in C_j$ y $v$ dos puntos consecutivos en $S$, de acuerdo a su coordenada $y$, donde $v$ es el punto con ordenada más grande. Dada la lista de puntos artificiales en $S_u$, $A_u$, podemos obtener en $O(n)$ tiempo la lista de puntos artificiales $A_v$ en $S_v$.*

*Demostración.* Podemos calcular, por la Observación 6, el punto artificial asociado a cada punto en $S_v$ de color distinto de $j$ en tiempo lineal. Hay que notar que algunos de los puntos artificiales inactivos en $S_u$ pueden convertirse en activos en $S_v$. El determinar cuáles de ellos caen en dicho caso puede hacerse en tiempo lineal, ésto al barrer el plano de derecha a izquierda con una recta vertical y deteniéndonos en cada punto de $S_v$.

La lista ordenada de puntos artificiales de color $j$ en $S_v$ puede obtenerse también en tiempo lineal con la misma idea del barrido de derecha a izquierda. Si un punto en dicha lista es activo o no, puede determinarse a lo largo del barrido.

Por tanto, $A_v$ puede obtenerse al mezclar las dos listas ordenadas de puntos artificiales descritas arriba, lo cual claramente toma $O(n)$ tiempo.

■

Algo que nos será de utilidad más adelante es el asumir que la lista $H_p$, asociada a un punto inactivo $p'_i$ de color $j$, siempre existe. Este tipo de lista será una copia de la lista $H_p$ asociada al primer punto real $q \in C_j$ a la izquierda de $p_i$, el cual determina el estatus de inactivo del punto $p'_i$. De hecho, la construcción de todas las listas de este tipo puede hacerse conforme construimos las listas $H_p$ en el Lema 5, sin ningún costo extra.

Observemos que únicamente las listas $H_p$ asociadas a puntos reales o a puntos artificiales en $S_u$, con coordenada $x$ a lo más $u_x$, contienen el punto $u$. Existe un número lineal de dichas listas y necesitamos modificarlas cuando nos movemos de $u$ a $v$.

El siguiente lema introduce una idea clave, la cual es usada para actualizar cada lista $H_p$ en $S_u$, ésto para que su información quede acorde a $S_v = S_u \setminus \{u\}$, reflejando el hecho de que todos los conjuntos $Q_p$ considerados en $S_v$ tienen al punto $u$ en su frontera exterior.

**Lema 7.** *Sean $u \in C_j$ y $v$ dos puntos consecutivos en $S$ de acuerdo a su coordenada $y$, donde $v$ es el punto con la ordenada más grande. Cuando nos movemos de $u$ a $v$, toma $O(1)$ tiempo el insertar al punto apropiado $z \neq u$, de color $j$, en cada $H_p$ de $S_u$ que contiene a $u$.*

*Demostración.* Sea $z$ el punto de color $j$ en la lista $H_p$ asociada al punto $q$ que se encuentra inmediatamente a la derecha de $u$ en $S_u$. La primera lista que necesitamos modificar es la asociada al punto $q' \neq u$, cuya coordenada $x$ es la más grande y a lo más $u_x$.

La presencia de $u$ en la lista $H_p$ de $q'$ puede eliminarse en tiempo constante, simplemente extrayendo el punto de color $j$ en la lista. La inserción de $z$, el nuevo punto de color $j$ en $H_p$, puede hacerse de la manera siguiente: sean $l$ y $l'$ los colores de los puntos que están justo antes y después de $z$ en la lista $H_p$ de $q$. Podemos obtener los puntos $r$ y $r'$ en la lista $H_p$ asociada a $q'$, de colores $l$ y $l'$ respectivamente, en tiempo constante. A causa de la Observación 5, al realizar a lo más tres comparaciones entre las coordenadas $y$ de $z$, $r$ y $r'$, es posible determinar la posición exacta en la lista $H_p$ de $q'$ donde se debe insertar a $z$. Podemos aplicar este mismo argumento de manera iterativa en el $O(n)$ listas a la izquierda.

Como ya lo mencionamos, $v$ es el punto de su color más cercano a cada cuadrante $Q_p$ que es considerado en $S_v$, lo cual puede reflejarse en cada lista $H_p$ de un punto con coordenada $x$ a lo más $v_x$ en tiempo constante. ■

Cuando nos desplazamos de $u$ a $v$, necesitamos considerar también la nueva posición de cada punto artificial $p'_i$ para actualizar algunas listas $H_p$. Teniendo en cuenta la Observación 6, el punto $p'_i$ *intercambia* su posición con cada punto real que yace entre la posición inicial de $p'_i$ en $A_u$ y su

posición final en $A_v$. En cada uno de estos intercambios, algunos cambios deben hacerse en la lista $H_p$ del punto real en turno, pero también a la lista $H_p$ de $p'_i$.

Sea $p_i \in C_m$ el punto real al cual $p'_i$ está asociado, con $m \neq j$. Si $p'_i$ es activo en $A_v$, pero no en $A_u$, entonces eso significa que existe un primer punto real $q$, de color $m$, a la derecha de $p_i$. Ya que la lista $H_p$ asociada a $p'_i$ es una copia de la lista $H_p$ de $q$, los únicos puntos reales en los que estamos interesados son los que se encuentran en el intervalo abierto $(q, p'_i)$. El siguiente lema muestra el tiempo requerido por cada uno de estos cambios de posición para hacer las actualizaciones correspondientes.

**Lema 8.** *Las listas afectadas por un intercambio entre un punto artificial $p'_i$ y un punto real $r$, cuando nos movemos de $u$ a $v$, pueden actualizarse en $O(1)$ tiempo, donde $p_i$ y $u$ tienen distinto color.*

*Demostración.* Sea $r'$ el siguiente punto real a la derecha de $r$. Podemos obtener en tiempo constante el punto $z$ en la lista $H_p$ de $r'$, que tiene el mismo color que el de $r$, y también los dos colores de los dos puntos alrededor de $z$ en dicha lista $H_p$. Entonces, debemos eliminar $r$ de la lista $H_p$ de $p'_i$ e insertar $z$ en la posición correcta; ambas operaciones pueden hacerse en tiempo constante al usar la misma técnica que se describe en el Lema 7. También marcamos a $p_i$ como miembro de $H_p \cap M_p$ en la lista $H_p$ de $r$. Por tanto, se sigue el resultado.

■

**Observación 7.** Si $u \in C_j$, pero $v \notin C_j$, entonces los puntos artificiales de color $j$, asociados a los puntos reales cuya coordenada $x$ es a lo más $u_x$, se desplazan a la izquierda. Las actualizaciones necesarias (inserciones, desactivaciones, etc.) pueden hacerse al usar las mismas ideas descritas en el lema anterior, pero en este caso, de derecha a izquierda.

Establecemos ahora el teorema siguiente, ya que al barrer el plano de abajo hacia arriba, con una recta horizontal, podemos hacer un número lineal de paradas al considerar cada punto de $S$.

**Teorema 5.1.** *Sea $S$ un conjunto $k$-coloreado de $n$ puntos en el plano en posición general. Es posible encontrar un $L$-corredor $k$-cromático de ancho mínimo, $L_w(p)$, en $O(n^2)$ tiempo.*

*Demostración.* Sea $u$ el primer punto de $S$ en el cual se detiene un barrido de recta de abajo hacia arriba. Por el Lema 5, es posible saber el $L$-corredor $k$-cromático de mínimo ancho $L_w(p)$, tal que $u$ es un punto esencial en su frontera exterior.

Lo que necesitamos entonces es contar el número de actualizaciones que puede haber durante el resto del barrido de abajo hacia arriba. Cuando la recta pasa al siguiente punto, llamémoslo $v$, el Lema 6 nos garantiza que la nueva lista ordenada de puntos reales y puntos artificiales (activos) en $v$ se obtiene en $O(n)$ tiempo. Así, un barrido de izquierda a derecha, que se detiene en cada punto de esta lista, puede realizarse.

Las listas $V_p$, cuyos puntos están ordenados por coordenada $x$, pueden actualizarse sin ningún problema. La clave para lograrlo reside en que la $x$-distancia de un punto al conjunto $Q_p$ en turno

siempre aumenta conforme el barrido se desplaza a la izquierda. Cuando un punto real se considera, sólo necesitamos hacer un borrado y una inserción de un elemento en $V_p$; ambas operaciones pueden hacerse en tiempo constante. Para el caso en que nos detenemos en un punto activo de color $i$, sólo necesitamos establecer que el punto de dicho color en $V_p$ no es un miembro de $V_p \cap M_p$. Claramente, el punto en $V_p$ con máxima $x$-distancia se puede mantener también en tiempo constante a lo largo de este barrido de derecha a izquierda.

Por el Lema 7, en cada parada del barrido de abajo hacia arriba, necesitamos gastar $O(n)$ tiempo con el fin de eliminar la presencia del punto anterior en que nos detuvimos de las listas $H_p$. Notemos que el número de intercambios, es decir, el número de actualizaciones adicionales que se hacen a las listas $H_p$ no es el mismo para cada parada del barrido de abajo hacia arriba. Entonces, podemos usar el siguiente argumento amortizado para contar el número total de intercambios que se sucitan durante todo el barrido. Sea $q \in C_j$ un punto en $S$, por el Lema 8 y la Observación 7, un punto artificial $p'_i \in C_m$, con $j \neq m$, intercambia su posición con $q$ a lo más dos veces. Así, el número de veces que $q$ intercambia de posición con un punto artificial, de cualquier color, es menor que $\sum_{i=1}^{k} 2|C_i| = 2n$. Entonces, el número total de intercambios al final del barrido de abajo hacia arriba es de $O(n^2)$, ya que $q$ puede ser cualquiera de los $n$ puntos en $S$.

Como consecuencia, el $L$-corredor $k$-coloreado de ancho mínimo, tal que un punto $v$ es un punto esencial en su frontera exterior, pude determinarse en tiempo lineal después de haberse actualizado todas las listas $H_p$ y $V_p$ en cada parada $v$ del barrido de abajo hacia arriba.

Por lo tanto, al final del barrido de abajo hacia arriba, obtenemos en $O(n^2)$ tiempo el $L$-corredor $k$-cromático de ancho mínimo en $S$.

■

## 5.3. Suma de $w_x$ y $w_y$ con mínimo valor

En esta sección tratamos con un problema de optimización distinto: encontrar el $L$-corredor $k$-cromático $L(p, q)$, tal que la suma de sus anchos es mínima.

El siguiente lema caracteriza una solución para este problema y su prueba se deja al lector, ya que ésta es practicamente igual a la del Lema 4.

**Lema 9.** *Sea $S$ un conjunto $k$-coloreado de $n$ puntos en el plano en posición general. Entonces, existe un $L$-corredor $k$-cromático que minimiza la suma de sus anchos y satisface lo siguiente:*

*i) Cada uno de los rayos que pertenece a la frontera exterior de $L(p, q)$ contiene un punto esencial.*

*ii) Cada uno de los rayos que forman parte de la frontera interior de $L(p, q)$ contiene un punto esencial.*

El algortimo propuesto para resolver este problema usa las misma ideas básicas descritas en la Sección 5.2: dado un conjunto $Q_p$, encontramos un conjunto $Q_q$ que nos permita generar un $L$-

corredor $k$-cromático con las características deseadas. Así, llevaremos nuevamente a cabo un doble barrido con rectas, uno vertical y otro horizontal, ésto con el fin de procesar todos los posibles conjuntos $Q_p$ inducidos por pares de puntos en $S$.

Usamos una vez más las listas $H_p$ y $V_p$, pero en este caso sus elementos están ordenados en orden creciente de acuerdo a su coordenada $y$ y $x$, respectivamente. Sea $h_i$ el $i$ésimo elemento de $H_p$, con $1 \leq i \leq k$; definimos $H_p^i = \{h_1, \ldots, h_i\}$. Análogamente, sea $V_p^i$ el conjunto que contiene los primeros $i$ puntos de $V_p$. Podemos ver, de manera directa, que la Observación 5 sigue siendo válida para el problema que tratamos en esta sección.

Supongamos que hemos encontrado un $L$-corredor solución $L(p, q)$. La propiedad *ii*) del Lema 9 nos dice que hay dos puntos, $h_i \in H_p$ y $v_j \in V_p$, cuyos colores son diferentes y definen el conjunto $Q_q$. Entonces, no existe un punto en $H_p^i$ con el mismo color del punto $v_j$, y también no hay un punto en $V_p^j$ con el mismo color del punto $h_i$. Notemos que $H_p^i \cup V_p^j$ es un conjunto $k$-coloreado.

Dado un conjunto $Q_p$, nuestro algoritmo itera sobre cada elemento de $V_p$, comenzando con el último. Para cada $v_i$, encontramos un punto $h_j \in H_p$, aquel con mínimo índice tal que $V_p^i \cup H_p^j$ es un conjunto $k$-coloreado. Así, asumamos que al estar en $v_i$, su elemento correspondiente es $h_j$, lo cual nos permite definir un conjunto $Q_p$. Observemos que cuando pasamos al punto $v_{i-1}$, no necesitamos revisar nuevamente los elementos $h_l$ con $l > j$. Por tanto, esta iteración sobre $V_p$ toma $O(k)$ tiempo amortizado. Realizamos también este último proceso, pero iterando sobre $H_p$ en orden decreciente. Conforme consideramos distintos conjuntos $Q_p$, preservamos la mejor opción.

Con todo lo anterior, este resultado se sigue inmediatamente:

**Teorema 5.2.** *Sea $S$ un conjunto $k$-coloreado de $n$ puntos en el plano en posición general. Es posible encontrar un $L$-corredor $k$-cromático $L(p, q)$, el cual minimiza $w_x + w_y$, en $O(n^2 k)$ tiempo.*

## 5.4. Versión no orientada

En esta sección, una vez más estamos interesados en encontrar un $L$-corredor $k$-cromático con mínimo ancho en un conjunto $k$-coloreado $S$ de puntos en el plano. No obstante, en contraste con los resultados presentados en la Sección 5.2, aquí abordamos este problema considerando rotaciones del plano.

Dado un ángulo $\theta$, con $0 \leq \theta < 2\pi$, llamamos $S_\theta$ al conjunto de puntos al cual se mapea $S$ cuando rotamos el plano $\theta$ grados alrededor del orígen. $L_\theta(p)$ denotará un $L$-corredor $k$-cromático de mínimo ancho para $S_\theta$, $w_\theta$ denota su ancho , y $\partial Q_p^\theta$ representa su frontera exterior. El algoritmo que presentamos a continuación encuentra un ángulo $\theta$ tal que $w_\theta$ se minimiza, ésto en $O(n^3 \alpha(n)\, \alpha(k) \log k)$ tiempo.

Notemos que el Lema 4 es válido también para una solución $L_\theta(p)$ de este problema. Entonces, a partir de ahora, supondremos que cada uno de los rayos de la frontera exterior de un $L$-corredor solución pasa a través de un punto esencial diferente. Los resultados presentados más adelante pueden ser modificados fácilmente para el caso cuando estos dos puntos son el mismo.

Sean $r, r' \in S$ los puntos esenciales en cada uno de los rayos de $\partial Q_p^\theta$ de un $L$-corredor solución $L_\theta(p)$. El punto $p$ debe estar en uno de los semicírculos cuyos extremos son $r$ y $r'$; véase Figura 5.3.

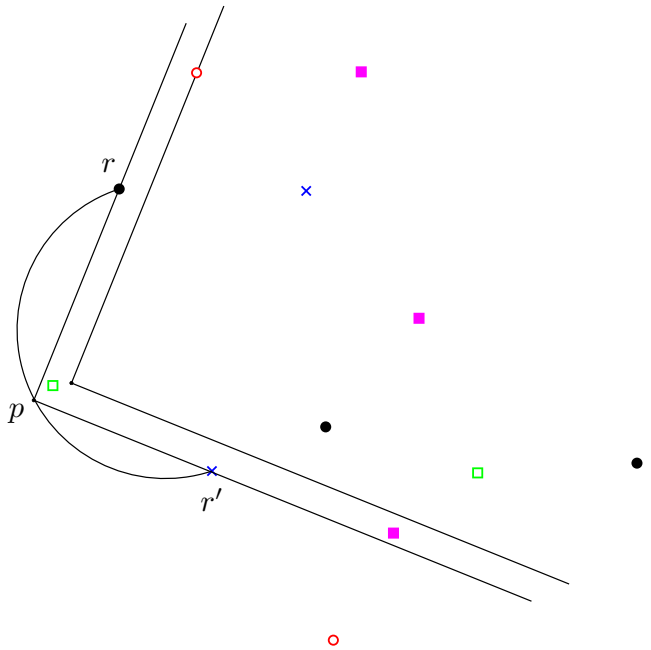


Figura 5.3: Un $L$-corredor $k$-cromático de ancho mínimo .

Así como lo hicimos antes, la idea detrás de nuestro algoritmo para encontrar una solución es considerar diferentes conjuntos $Q_p^\theta$. Dados dos puntos $q$ y $q'$, ambos en $S$, sea $Q_p^\theta$ un conjunto cuya frontera pasa a través de $q$ y $q'$. Conforme movemos el punto $p$ sobre uno de los semicírculos con extremos en $q$ y $q'$, los puntos en $S$ pueden entrar, permanecer o salir del conjunto $Q_p^\theta$ inducido. En cada instante, necesitamos saber, para cada clase cromática, cuál es el punto más cercano a $\partial Q_p^\theta$ en el interior de $Q_p^\theta$. Si supiéramos estos puntos, entonces una posible solución estaría dada por el punto del color más lejano que minimiza su distancia en todo el circuito hecho por $p$ desde $q$ hasta $q'$.

Con el fin de encontrar dichos puntos más cercanos a $\partial Q_p^\theta$, usamos la dualidad geométrica punto-recta: dado un punto $p$ y una recta $\ell$, denotamos como $p*$ y $\ell*$ sus duales, respectivamente. Así, sea $\mathcal{D}$ el arreglo de rectas duales asociadas a los puntos en el conjunto $S$; sean también $\overline{x}$ y $\overline{y}$ las dos rectas que contienen la frontera de un conjunto $Q_p^\theta$, el cual pasa por $q$ y $q'$.

Observemos que el movimiento de $p$, desde $q$ hacia $q'$, equivale a barrer el espacio dual con dos rayos verticales, los cuales empiezan a moverse a partir de $\overline{x}*$ y $\overline{y}*$ y su distancia relativa es $\frac{\pi}{2}$. No obstante, no barreremos el espacio dual con dichos rayos, en su lugar, construimos una subestructura de $\mathcal{D}$ para obtener la información necesaria.

Denotemos como $m(\overline{x})$ a la pendiente de la recta $\overline{x}$. Dado un conjunto $Q_p^\theta$, no es difícil ver que cada punto $s \in S$ entra y sale de $Q_p^\theta$ a lo más una vez. Considerando este hecho, asociamos a cada

punto $s$ el conjunto $r^x$:

$$r^x = \{ (a, b) \in s* \mid m(\overline{x}) = a \Rightarrow s \in Q_p^\theta \}$$

Entonces, para cada color $i$, con $1 \leq i \leq k$, podemos construir un arreglo $A_i^x$, el cual consiste de todos los conjuntos $r^x$ asociados a los puntos de color $i$ en $S$. Observemos que cada arreglo $A_i^x$ contiene $O(n)$ segmentos de recta; el punto de color $i$ más cercano al rayo $\overline{x}$ en el interior de $Q_p^\theta$, cuando $\overline{x}$ tiene pendiente $a$, está dado por la envolvente inferior de los elementos en $A_i^x$. Análogamente se puede definir el conjunto $A_i^y$, ésto al considerar como referencia el rayo $\overline{y}$.

Así, nuestro objetivo es traslapar los dos subarreglos, $A_i^x$ y $A_i^y$ , de tal manera que la envolvente inferior del arreglo resultante, $A_i$, nos proporcione el punto de color $i$ más cercano a $\partial Q_p^\theta$, ésto en cada instante dell movimiento que realiza $p$ desde $q$ hacia $q'$. Con el fin de lograr este traslape, es necesario obtener dichos subarreglos $A_i^x$ y $A_i^y$ en el espacio dual, al establecer $q$ y $q'$, respectivamente, como el orígen del espacio primal. Por tanto, si $s$ es el punto de color $i$ más cercano a la frontera de $Q_p^\theta$ para una orientación $\theta$ del rayo $\overline{x}$, entonces el punto en $s*$ con abcisa $a$ es parte de la envolvente inferior de $A_i$.

Se sabe que la envolvente inferior de un arreglo cuyos elementos son segmentos de recta, como es el caso de $A_i$, puede obtenerse en $O(n \log n)$ tiempo [66] (Corolario 6.6). La complejidad combinatoria de dicha envolvente superior es $O(n\,\alpha(n))$ (Corolario 2.17), donde $\alpha(\cdot)$ denota la función inversa de Ackerman. Sea $\mathcal{A}$ el arreglo formado por todas las envolventes inferiores $L(A_i)$, con $1 \leq i \leq k$. El punto más cercano del color más lejano a $\partial Q_p^\theta$ se obtiene entonces al determinar el mínimo de la envolvente superior de $\mathcal{A}$. El arreglo $\mathcal{A}$ se puede ver como un arreglo con $k$ cadenas polígonales, cuyo número total de segmentos en $O(n\,\alpha(n))$, entonces su envolvente superior puede calcularse en $O(n\,\alpha(n)\,\alpha(k) \log k)$ tiempo.

Consecuentemente, al aplicar el algoritmo previo para cada par de puntos en $S$, podemos establecer el resultado siguiente.

**Teorema 5.3.** *Sea $S$ un conjunto $k$-coloreado de $n$ puntos en el plano en posición general. Podemos encontrar en $O(n^3\alpha(n)\alpha(k) \log k)$ tiempo un ángulo $\theta$, tal que el $L$-corredor $k$-cromático $L_\theta(p)$ tiene ancho mínimo.*

## 5.5. Comentarios finales

La investigación que puede iniciarse directamente, con base en los resultados presentados a lo largo de este capítulo, bien podría ser el encontrar algún detalle o idea clave que nos ayude a mejorar cualquiera de las complejidades de los algoritmos mostrados, tal y como fue el caso del algoritmo final que se presentó en la Sección 5.2. Probablemente, en el caso no orientado de mínimo ancho y en el orientado que minimiza la suma de sus anchos es donde existan más posibilidades de hacer dicha mejora.

También sería deseable el establecer una cota mínima de complejidad para alguna de las ver-

siones aquí estudiadas; inclusive, se podría plantear y estudiar la variante no orientada de un $L$-corredor $k$-cromático que minimice la suma de sus anchos.

Como se mencionó al inicio de este capítulo, el estudio de objetos $k$-cromáticos bajo ciertas restricciones es un área relativamente nueva. Por tanto, hay todavía mucho que investigar al respecto y tal vez para comenzar, uno podría preguntarse por el triángulo $k$-cromático de área mínima en un conjunto de puntos.

# Referencias


[1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. Smallest color-spanning objects. *Lecture Notes in Computer Science*, 2161:278–289, 2001.

[2] P. K. Agarwal and S. Suri. Surface approximation and geometric partitions. In *SODA '94: Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 24–33, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[3] J. Akiyama and J. Urrutia. Simple alternating path problem. *Discrete Mathemathics*, 84:101–103, 1990.

[4] C. Bautista-Santiago, J. Cano-Vila, J. M. Díaz-Báñez, H. González-Aguilar, D. Lara, and J. Urrutia. *L*-corredor *k*-cromático. In *Actas de los XIII Encuentros de Geometría Computacional*, 2009.

[5] K. P. Bennett and E. J. Bredensteiner. Duality and geometry in SVM classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 161–167. ACM, 2000.

[6] J. L. Bentley. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27:865–873, 1984.

[7] S. Bereg, S. Cabello, J. M. Díaz-Báñez, P. Pérez-Lantero, C. Seara, and I. Ventura. The class cover problem with boxes. In *Proceedings of the 26th European Workshop on Computational Geometry*, pages 69–72, 2010.

[8] S. Bespamyatnikh and M. Segal. Enumerating longest increasing subsequences and patience sorting. *Information Processing Letters*, 76:7–11, 2000.

[9] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14:463–479, 1995.

[10] A. H. Cannon and L. J. Cowen. Approximation algorithms for the class cover problem. *Annals of Mathematics and Artificial Intelligence*, 40(3-4):215–223, 2004.

[11] T. M. Chan. Low-dimensional linear programming with violations. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 570–583. IEEE Computer Society, 2002.

[12] R. Cole, M. Sharir, and C. K. Yap. On *k*-hulls and related problems. *SIAM Journal on Computing*, 16:61–77, 1987.

[13] C. Cortés, J. M. Díaz-Báñez, P. Pérez-Lantero, C. Seara, J. Urrutia, and I. Ventura. Bichromatic separability with two boxes: A general approach. *Journal of Algorithms*, 64:79–88, 2009.

[14] C. Cortés, J. M. Díaz-Báñez, and J. Urrutia. Finding enclosing boxes with empty intersection. In *23rd. European Workshop on Computational Geometry*, pages 185–188, 2006.

[15] N. Cristianini and J. Shaw-Taylor. *Support Vector Machines*. Cambridge University Press, 2000.

[16] S. Das, P. P. Goswami, and S. C. Nandy. Recognition of minimum width color-spanning corridor and minimum area color-spanning rectangle. *Lecture Notes in Computer Science*, 3480:827–837, 2005.

[17] S. Das, P. P. Goswami, and S. C. Nandy. Smallest color-spanning object revisited. *International Journal of Computational Geometry and Applications*, 19(5):457–478, 2009.

[18] J. G. DeVinney. *The class cover problem and its applications in pattern recognition*. PhD thesis, The Johns Hopkins University, 2003.

[19] R. B. K. Dewar, S. M. Merritt, and M. Sharir. Some modified algorithms for dijkstra's longest upsequence problem. *Acta informatica*, 18:1–15, 1982.

[20] J. M. Díaz-Báñez, G. Hernández, D. Oliveros, A. Ramírez-Vigueras, J. A. Sellarès, and J. Urrutia. Computing shortest heterochromatic monotone routes. *Operational Research Letters*, 36:684–687, 2008.

[21] J. M. Díaz-Báñez and I. Ventura. Localización no puntual: Una perspectiva desde la Geometría Computacional. In B. Pelegrín, editor, *Avances en localización de servicios y sus aplicaciones*, pages 165–190. Universidad de Murcia, Spain, 2004.

[22] E. W. Dijkstra. Some beautiful arguments using mathematical induction. *Acta informatica*, 13:1–8, 1980.

[23] D. P. Dobkin, D. Gunopulos, and W. Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *Journal of Computer and System Sciences*, 52(3):453–470, 1996.

[24] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons Inc., 2001.

[25] J. Eckstein, P. L. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its application to data analysis. *Comput. Optim. Appl.*, 23(3):285–298, 2002.

[26] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangemets of lines and hiperplanes with applications. *SIAM Journal on Computing*, 15:341–363, 1986.

[27] D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. Finding minimum area $k$-gons. *Discrete and Computational Geometry*, 7:45–58, 1992.

[28] H. Everett, J. Robert, and M. van Kreveld. An optimal algorithm for the ($\leq k$)-levels, with applications to separation and transversal problems. In *Proceedings of the 9th Annual Symposium on Computational Geometry*, pages 38–46, San Diego, CA, USA, 1993. ACM.

[29] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of ACM*, 45(4):634–652, 1998.

[30] S. Fekete. Personal communication, 1992.

[31] P. Fischer. Sequential and parallel algorithms for finding a maximum convex polygon. *Computational Geometry: Theory and Applications*, 7:187–200, 1997.

[32] A. R. Forrest. *Curves and surfaces for computer aided design*. PhD thesis, University of Cambridge, 1968.

[33] A. R. Forrest. Computational geometry. In *Proceedings of the Royal Society of London*, pages 187–195. The Royal Society, 1971.

[34] A. R. Forrest. Computational geometry - achievements and problems. *Computer Aided Geometric Design*, pages 17–44, 1974.

[35] C. C. Foster. Information storage and retrieval using AVL trees. In *Proceedings of the ACM 20th National Conference*, pages 192–205, 1965.

[36] M. L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11:29–35, 1975.

[37] A. Gajentaan and M. H. Overmars. On a class of $o(n^2)$ problems in Computational Geometry. *Computational Geometry*, 5(3):165–185, 1995.

[38] M. Garey and D. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[39] F. Gómez, F. Hurtado, S. Ramaswami, and V. Sacristán. Implicit convex polygons. *Journal of Mathematical Modelling and Algorithms*, 1(1):57–85, 2002.

[40] J. E. Goodman and R. Pollack. Semispaces of configurations, cell complexes of arrangements. *Journal of Combinatorial Theory. Series A.*, 37(3):257–293, 1984.

[41] P. Hammer and T. Bonates. Logical analysis of data, An overview: From combinatorial optimization to medical applications. *Annals of Operations Research*, 148:203–225, 2006.

[42] S. Har-Peled and V. Koltun. Separability with outliers. *Lecture Notes in Computer Science*, 3827:28–39, 2005.

[43] T. Hatie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, Berlin, Germany, 2001.

[44] D. Haussler and E. Welzl. $\varepsilon$-nets and simplex range queries. *Discrete and Computational Geometry*, 2:127–151, 1987.

[45] J. E. Hopcroft and R. M. Harp. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1987.

[46] M. E. Houle. Algorithms for weak and wide separation of sets. In *Proceedings of the International Workshop on Discrete Algorithms and Computing*, pages 61–68, 1989.

[47] M. E. Houle. *Weak Separation of Sets*. PhD thesis, McGill University, 1989.

[48] M. E. Houle. Algorithms for weak and wide separation of sets. *Discrete Applied Mathematics*, 45(2):139–159, 1993.

[49] F. Hurtado, M. Noy, P. A. Ramos, and C. Seara. Separating objects in the plane with wedges and strips. *Discrete Applied Mathematics*, 109(1-2):109–138, 2001.

[50] D. P. Huttenlochar, K. Kadem, and M. Sharir. The upper envelope of voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9(1):267–291, 1993.

[51] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[52] A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane - a survey. In *Discrete and Computational Geometry, The Goodman-Pollack Festschrift, vol. 25 of Algorithms and Combinatorics*, pages 551–570. Springer-Verlag, 2003.

[53] M. Kudo, Y. Torii, Y. Mori, and M. Shimbo. Approximation of class regions by quasi convex hulls. *Pattern Recognition Letters*, 19(9):777–786, 1998.

[54] D. Lichteinstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11:329–343, 1982.

[55] D. Marchette. Class cover catch digraphs. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):171–177, 2010.

[56] J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: small $\varepsilon$-nets for disks and halfspaces. In *Proceedings of the 6th Annual ACM Symposium on Computational Geometry*, pages 16–22, 1990.

[57] N. Megiddo. Linear-time algorithms for linear programming in $R^3$ and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[58] J. S. B. Mitchell. Approximation algorithms for geometric separation problems. Technical report, Dept. of Applied Math. and Statistics, SUNY at Stony Brook, July 1993.

[59] M. Orlowski and M. Pachter. An algorthm for the determination of a longest increasing subsequence in a sequence. *Computers & Mathematics with Applications*, 17:1073–1075, 1989.

[60] P. Pérez-Lantero. *Geometric optimization for classification problems*. PhD thesis, Universidad de Sevilla, 2010.

[61] C. Schensted. Longest increasing and decreasing subsequences. *Canadian Journal of Mathemathics*, 13:179–191, 1961.

[62] C. Seara. *On Geometric Separability*. PhD thesis, Universitat Politècnica de Catalunya, 2002.

[63] M. I. Shamos. Geometric complexity. In *Proceedings of the 7th Annual ACM Symposium on Theory of Computing*, pages 224–233. ACM, 1975.

[64] M. I. Shamos. *Computational Geometry*. PhD thesis, Yale University, 1978.

[65] M. I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the16th Annual Symposium on Foundations of Computer Science*, pages 151–162, 1975.

[66] M. Sharir and P. K. Agarwal. *Davenport-Shinzel Sequences and their Geometric Applications*. Cambridge, 1995.

[67] J. Stoer and C. Witzgall. *Convexity and Optimization in Finite Dimensions*. Springer - Verlag, 1970.

[68] S. Tokunaga. Intersection number of two connected geometric graphs. *Information Processing Letters*, 59:331–333, 1996.

[69] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.