VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ALGUNOS PROBLEMAS CON OBJETOS GENERADORES CROMÁTICOS

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:
ÓSCAR TONATIUH ZAMUDIO OCÁDIZ

DIRECTOR DE TESIS:
DR. JORGE URRUTIA GALICIA
INSTITUTO DE MATEMÁTICAS - UNAM

MÉXICO, D. F. SEPTIEMBRE 2013

Índice general

Introducción

En la geometría computacional existe una serie de problemas sobre encontrar corredores y otros objetos geométricos vacíos o de medidas máximas o mínimas en un conjunto de puntos dados.

El problema del corredor vacío más ancho se originó en la planeación de movimientos de robots. El objetivo era encontrar la ruta en línea recta de mayor ancho que evitara obstaculos [1]

Posteriormente se idearon versiones más complejas de este problema: el corredor debe tener una esquina y el ángulo que forman las dos piernas del mismo debe ser recto[2] (l-corredor). En una versión posterior y más compleja del problema, la restricción a que el ángulo que forman las piernas sea recto es eliminada[3] (corredor de una esquina).

Otra versión de los anteriores problemas es la cromática: los puntos del conjunto dado tienen ahora un color y existen k colores posibles. Los objetos geométricos (corredor, l-corredor) en estas versiones de los problemas ahora deben contener en su interior un punto de cada color y se busca ahora minimizar las anchuras de dichos corredores.

Naturalmente surgen problemas similares: dado un conjunto de puntos cual es el rectangulo vacío más grande que puede obtenerse. O el circulo. Otra posible variación son las versiones k-densas, surgidas también de la planeación de movimientos de robots. El problema del corredor $k-$denso se formuló en un artículo de Chattopadhyay y Das[4], donde también se propuso un algoritmo de complejidad espacial $O(n^2 \log n)$ y complejidad temporal $O(n^2)$. Este problema está elaborado bajo la suposición de que el robot puede tolerar la colisión con un número especificado (k) de obstaculos.

Esta tesis presenta un algoritmo para encontrar corredores de una esquina cromáticos. La organización de la tesis es la siguiente. En el primer capítulo mostramos una serie de resultados que no involucran directamente corredores en conjuntos de puntos, los cuales serán utilizados en los capítulos posteriores. El capítulo segundo hace un repaso de los resultados de corredores y otros

[1] M. Houle, A. Maciel *Finding the widest empty corridor through a set of points.* Report SOCS-88. 11, McGill University, Montreal, Quebec. 1988.

[2] [Cheng1996]

[3] [DiazBanez2006] y [Das2009a]

[4] S. Chattopadhyay, P.P. Das, *The k-dense corridor problems*, Pattern Recognitions Lett. 11 (1990) 463 469.

objetos vacios en conjuntos de puntos no cromáticos. El capitulo tercero repasa resultados similares en conjuntos de puntos coloreados. Finalmente en el capítulo cuarto presentamos el algoritmo para el corredor cromático de una esquina. De manera que esta tesis cumple el doble propósito de ser una antología de resultados relacionados con corredores y otros objetos de medidas mínimas o máximas en conjuntos de puntos y de mostrar un resultado novedoso relacionado con este tipo de problemas.

Capítulo 1

Preliminares

En este capítulo resumiremos algunos resultados que, aunque no relacionados directamente con los corredores en conjuntos de puntos, son utilizados en los algoritmos que se presentan más adelante. Primero veremos la dualización, una transformación especial que transforma puntos en rectas y viceversa, y que es utilizada en los algoritmos para l-corredores cromáticos. Posteriormente veremos un resultado concerniente al problema del ordenamiento radial, resultado utilizado en el algoritmo del corredor de una esquina. Dos resultados de conteo y consulta de rangos (rectangulares y triangulares) son utilizados en los algoritmos para el corredor de una esquina (tanto en la versión no cromática como en la cromática). Finalmente el cierre convexo dinámico es utilizado en el algoritmo para el corredor de una esquina no cromático.

1.1. Dualización

Sea $p = (p_x, p_y)$ un punto en el plano. El dual de p, denotado p^* es la línea $y = p_x x - p_y$. El dual de la línea l cuya ecuación es $y = mx + b$ es el punto $l^* = (m, -b)$.

La transformación dual no está definida para líneas verticales.

Se dice que la transformación dual mapea objetos del plano primal al plano dual. Algunas propiedades son invariantes en ambos planos:

Observación 1. *Sea p un punto y l una recta no vertical. La transformación dual $o \to o^*$ tiene las siguientes propiedades:*

- *Preserva incidencias: $p \in l$ sii $l^* \in p^*$.*
- *Preserva el orden: p se halla encima de l sii l^* se halla encima de p^*.*

1.2. Ordenamiento radial

En [Lee1985] se utiliza una definición distinta de dualidad: un punto $p = (a, b)$ en el plano es el *dual* de la recta $D_p : ax + by + 1 = 0$ y viceversa. Si d

es la distancia de p al origen O de los ejes cartesianos entonces D_p es la línea perpendicular a la línea Op a distancia $1/d$ y al otro lado de O. La dualidad geométrica preserva la relación de incidencia: el punto p se halla en la línea L sii el dual de p contiene al dual de L.

Dado un conjunto de puntos $S = \{p_1, p_2, \ldots, p_n\}$, el problema del orden radial consiste en encontrar el orden en que los puntos $S \backslash \{p_i\}$ son alcanzados por una semirrecta que parte de p_i en dirección vertical hacia arriba y que gira en sentido del reloj respecto a p_i, $i = 1, 2, \ldots, n$. Si 2 o más puntos son colíneales en la semirrecta que gira son ordenados según su distancia a p_i. Una aproximación básica al problema es ordenar n veces el conjunto, una vez para cada pivote p_i, lo que toma tiempo $O(n^2 \log n)$.

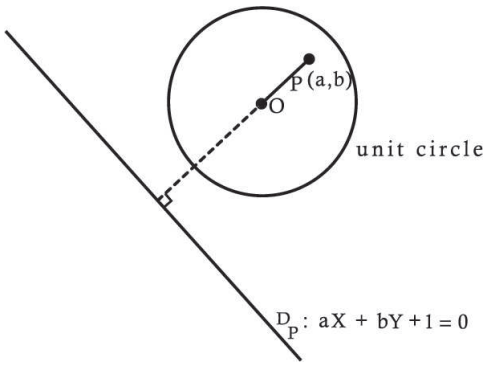


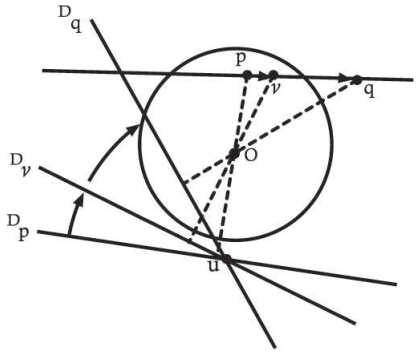
Figura 1.1: La transformación dual. [Lee1985]

Sea L la recta determinada por los puntos p y q. Sean D_p y D_q sus duales respectivos, y u el punto de intersección de ambas rectas. Por la propiedad de preservación de la incidencia de la transformación dual se tiene que L y u son duales uno del otro. Si un punto v se mueve desde p hasta q sobre el segmento pq entonces la línea dual D_v girará alrededor de u desde D_p hasta D_q. Observe que el dual de la línea determinada por O (el origen del plano cartesiano) y u se encuentra a distancia infinita de O y que las direcciones de los movimientos de v y D_v (en o contra el sentido del reloj) son ambas iguales respecto a O y u (si bien el primer movimiento es una translación y el segundo una rotación). El converso es verdadero también: si la línea Op rota alrededor de p en sentido del reloj su dual se mueve sobre D_p en sentido del reloj respecto a O. Vea las figuras 1.2 y 1.3.

Para simplificar su exposición, los autores suponen que todos los puntos en S están en el primer cuadrante del plano cartesiano. Sean $s_i, i = 1, \ldots, n$ los puntos en S y L_i sus respectivos duales. El arreglo de líneas determinado por L_i puede construirse en tiempo $O(n^2)$ [1]. Una vez tenemos el arreglo de las rectas,

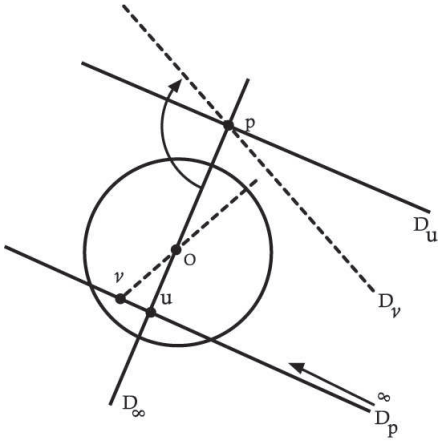
[1] B. M. Chazelle, L. J. Guibas, D. T. Lee. *The power of geometric duality.* Proc. IEEE 24th Symp. on Foundations of Computer Science (1983) 217-225. También BIT 25 (1985) 76-90.

H. Edelsbrunner, J. O'Rourke, R. Seidel. *Constructing arrangements of lines and hyperplanes with applications.* Proc IEEE 24th Symp. on Foundations of Computer Science (1983)



(a)

Figura 1.2: Movimiento de los duales sobre una línea y alrededor de un punto. Espacio primal. [Lee1985]



(b)

Figura 1.3: Movimiento de los duales sobre una línea y alrededor de un punto. Espacio dual. [Lee1985]

el cual es una gráfica $G = (V, E)$ construiremos la lista ordenada de los puntos $S\backslash\{p_i\}$ para cada p_i en tiempo $O(n)$. Sea $L(p_i)$ la línea que pasa por p_i y es perpendicular a la línea Op_i. Ambas líneas dividen al plano en 4 regiones. De acuerdo a lo explicado arriba si la línea Op_i rota alrededor de p_i en sentido del reloj entonces su dual se moverá sobre L_i en dirección del reloj respecto a O. De manera que si visitamos los puntos de intersección sobre L_i en dicho sentido determinaremos el orden de $S\backslash\{p_i\}$. El algoritmo mantendrá 4 listas

83-91.

ordenadas $LIST(p_i, q), q = 1, 2, 3, 4$ cada una correspondiente a los 4 cuadrantes mencionados arriba. P. ej., en la figura 1.4, $LIST(p_i, 2)$ consta de los puntos 1,3,4 y 6. Las listas son inicialmente vacías. Para cada punto de intersección v sobre L_i sea L_j la otra línea dual a la que v pertenece. El algoritmo determina a que cuadrante pertenece p_j y añade p_j al final de la lista correspondiente. Si hay más de 3 puntos colineales durante el barrido, el algoritmo debe realizar procesamiento extra, el cual toma tiempo $O(1)$. Los detalles pueden encontrarse en el artículo original. Finalmente las 4 listas son unidas para obtener la lista circular final que es el ordenamiento de $S\backslash\{p_i\}$, lo que lleva al

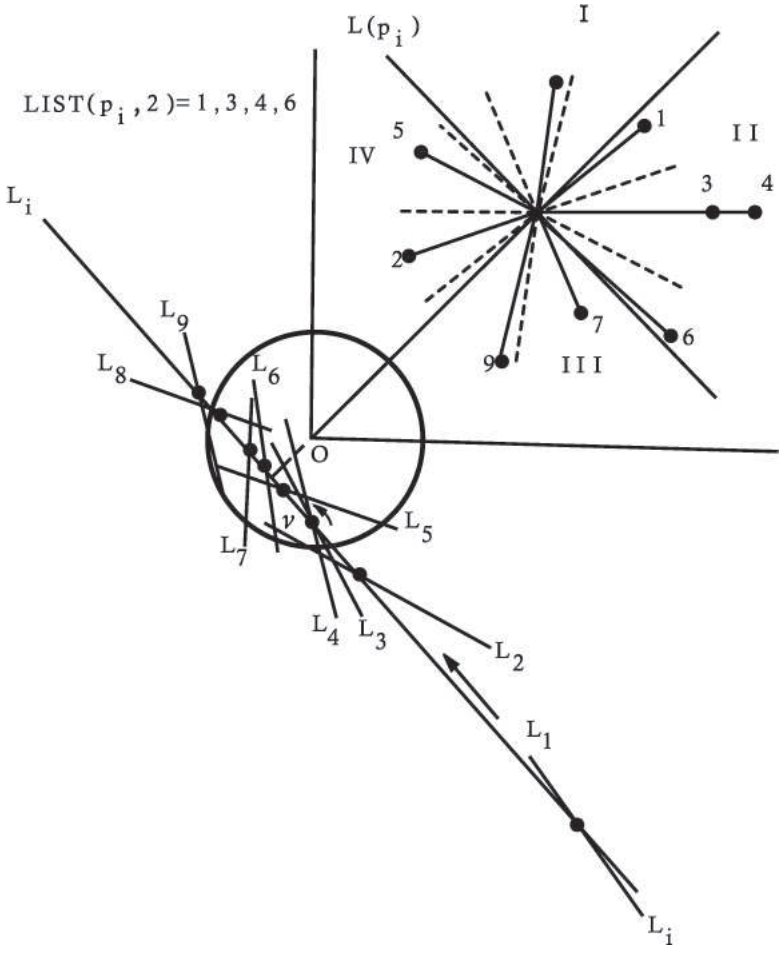


Figura 1.4: Listas $LIST(p_i, q), q = 1, 2, 3, 4$. [Lee1985]

Teorema 1. *Dado un conjunto de puntos $S = \{p_1, \ldots, p_n\}$ en el plano el problema del ordenamiento radial respecto a todos los puntos del conjunto puede resolverse en tiempo $O(n^2)$.*

1.3. Consulta de rangos rectangulares

Dado un conjunto de puntos en el espacio euclidiano n-dimensional, una *consulta de rangos rectangulares* (también llamada *consulta de rangos ortogonales*) consiste en reportar los puntos del conjunto que se encuentran dentro de una caja n-dimensional cuyos lados son paralelos a los ejes cartesianos. En particular nos enfocaremos en el caso de 2 dimensiones, la cual será utilizada por el algoritmo propuesto. Para resolver este tipo de consultas se utilizan estructuras de datos conocidos como *árboles de rangos*. En la exposición subsecuente asumiremos que estamos tratando el caso de 2 dimensiones.

Una primera aproximación a la solución de la consulta de rangos rectangulares de 2 dimensiones es realizar 2 subconsultas en una dimensión: una en las coordenadas x de los puntos y una subconsulta en la coordenada y de los puntos resultado de la primera consulta. Sea P el conjunto de tamaño n de puntos sobre los que se realizarán las consultas. Sea $[x : x'] \times [y : y']$ el rango consulta. Para hallar los puntos con coordenada x dentro del intervalo $[x : x']$ elaboramos un árbol binario de busqueda en las coordenadas x de los puntos. Buscamos dentro de dicho árbol a los valores x y x' hasta llegar a un nodo v_{split} donde las trayectorias de busqueda para ambos valores dentro del árbol se separan. La busqueda de x continua hacia el hijo izquierdo de v_{split} y en cada nodo v donde la busqueda para x proceda a la izquierda reportamos todos los puntos en el subárbol derecho de v. La busqueda de x' continua por el hijo derecho de v_{split} y en cada nodo v en que dicha busqueda proceda a la derecha se reporta todo el subárbol izquierdo de v. Finalmente se verifican las hojas μ y μ' donde ambas busquedas terminan y se reportan si están en el intervalo. De esta manera se obtienen $O(\log n)$ subárboles que contienen los puntos cuya coordenada x está en en el intervalo $[x : x']$. Vea la imagen 1.5.

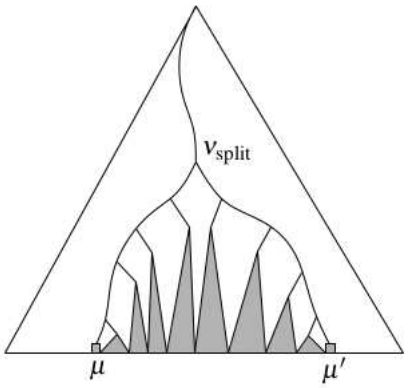


Figura 1.5: Una busqueda de rangos unidimensional. [DeBerg2008]

Dado un nodo v en el árbol de busqueda designaremos como *subconjunto canónico de* v, denotado $P(v)$, al conjunto de puntos en las hojas del subárbol

con raíz v. De manera que los puntos con coordenada x en el rango $[x : x']$ puede expresarse como la unión disjunta de $O(\log n)$ conjuntos canónicos. Del total de esos puntos, solo estamos interesados en aquellos cuya coordenada y está en el rango $[y : y']$. Para encontrarlos podemos realizar otra consulta de rangos, siempre que contemos con un árbol para búsqueda de rangos en las coordenadas y de los puntos para cada conjunto canónico $P(v)$. Con estás observaciones podemos definir el árbol de busqueda de rangos bidimensional:

- El árbol principal es un árbol de busqueda binaria T en la coordenada x de los puntos de P.
- Para cada nodo v de T se tiene un árbol de busqueda binaria en las coordenas y del conjunto canónico $P(v)$. Denotaremos dicho árbol $T_{assoc}(v)$. El nodo v tiene un apuntador a la raiz de $T_{assoc}(v)$. Llamaremos *estructura asociada a* v a $T_{assoc}(v)$.

En [DeBerg2008] se da un algoritmo para la construcción de árboles de busqueda de rangos. El algoritmo utiliza tiempo $O(n \log n)$. Además se prueba:

Lema 1. *Un árbol de rangos para n puntos en el plano utiliza espacio* $O(n \log n)$.

Lema 2. *Una consulta con un rectángulo de lados paralelos a los ejes toma tiempo* $O(\log^2 n + k)$ *donde k es el número de puntos reportados.*

Teorema 2. *Sea P un conjunto de n puntos en el plano. Un árbol de rangos para P utiliza espacio* $O(n \log n)$ *y puede ser construido en tiempo* $O(n \log n)$. *Consultando a este árbol se puede reportar los puntos de P dentro de un rango rectangular en tiempo* $O(\log^2 n + k)$, *donde k es el número de puntos reportados.*

Fraccionamiento en cascada

Al utilizar los árboles de rangos definidos en la sección anterior se deben realizar consultas para la coordenada y en $O(\log n)$ conjuntos canónicos. Cada una de estas consultas toma tiempo $O(\log n + k_v)$, donde k_v es el número de puntos reportados. Esto conlleva un tiempo total de $O(\log^2 n)$. El fraccionamiento en cascada es una técnica que mejora el tiempo de las consultas en la coordenada y a $O(1 + k_v)$, con lo que el tiempo total de la consulta se reduce a $O(\log n + k)$. Para ello se toma ventaja del hecho de que las consultas con sobre la coordenada y utilizan siempre el mismo rasgo.

Ilustraremos la idea del fraccionamiento en cascada mediante un ejemplo. Sean S_1 y S_2 dos conjuntos de objetos, cada uno de los cuales tiene una llave, la cual es un real. Suponga que los conjuntos de llaves están ordenados, respectivamente, en los arreglos A_1 y A_2. Suponga que se quiere reportar los objetos en ambos conjuntos tales que su llave se encuentra en el intervalo $[y : y']$. Una manera de resolver el problema es buscar a y mediante busqueda binaria A_1. Una vez hallado, recorremos el arreglo reportando los elementos cuya llave está en el intervalo hasta llegar a una llave mayor que y'. El mismo procedimiento se utiliza en A_2. Este algoritmo requiere realizar dos busquedas binarias. Si fuera

el caso que S_2 es un subconjunto de S_1 podriamos evitar realizar la segunda busqueda binaria de la siguiente manera: añada apuntadores en cada entrada $A_1[i]$ hacia $y_i \in A_2$, donde y_i es la entrada más pequeña en A_2 con valor mayor o igual que $A_1[i]$. Si no existe tal entrada, el apuntador será nulo. Un ejemplo se muestra en la figura 1.6

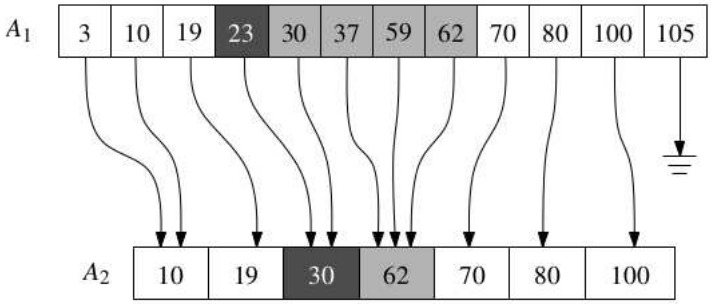


Figura 1.6: Apuntadores en el fraccionamiento en cascada. [DeBerg2008]

Para encontrar los puntos en S_1 y S_2 en el rango $[y : y']$ procedemos ahora de la siguiente manera. Primero realizamos una busqueda binaria para y en S_1 y recorremos dicho arreglo a partir de allí. Sea $A_1[i]$ la entrada en S_1 donde la busqueda por y terminó. La diferencia es que ahora no es necesario realizar la busqueda binaria en A_2: simplemente seguimos el apuntador y_i de $A_1[i]$ a la entrada de A_2 cuya llave es la más pequeña tal que es menor o igual que $A_1[i]$. Puesto que las llaves en A_2 son un subconjunto de las llaves en A_1, y_i es la entrada a la que llegaría la busqueda binaria en A_2. Así que podemos recorrer A_2 en orden ascendente desde este punto reportando las llaves en A_2 en el intervalo y, con ello, nos hemos ahorrado la segunda busqueda binaria.

Para aplicar las anteriores ideas a un árbol de rangos, note que los subconjuntos canónicos $P(lc(v))$ y $P(rc(v))$ son ambos subconjuntos de $P(v)$, donde $lc(v)$ y $rc(v)$ son, respectivamente, los hijos izquierdo y derecho de v. De manera que podemos aplicar las mismas ideas que en el ejemplo de S_1 y S_2, si bien los detalles serán un poco más complicados dado que ahora hay dos subconjuntos a los que deben mantenerse apuntadores. Para ello será necesario mantener cada subconjunto canónico $P(v)$ en un arreglo $A(v)$ en lugar de un árbol binario. Cada arreglo está ordenado según las coordenadas y de los puntos. Sea p el punto en la entrada $A(v)[i]$, y p_y su coordenada y. Si $A(v)[i]$ tiene un apuntador a la entrada de $A(lc(v))$ cuyo punto almacenado es p' entonces la coordena y de p' , p'_y, es la menor en $A(lc(v))$ que satisface $p_y \leq p'_y$. De esta manera si p es el punto en $P(v)$ con la menor coordenada y mayor o igual que un valor dado entonces p' tiene el punto con la misma propiedad en $P(lc(v))$. Definimos los apuntadores a $rc(v)$ de forma análoga. Este árbol de rangos modificado se denomina *árbol de rangos por capas*. Puede ver un ejemplo en la figura 1.7 y 1.8.

Veamos como realizar la consulta para el rango rectangular $[x : x'] \times [y : y']$

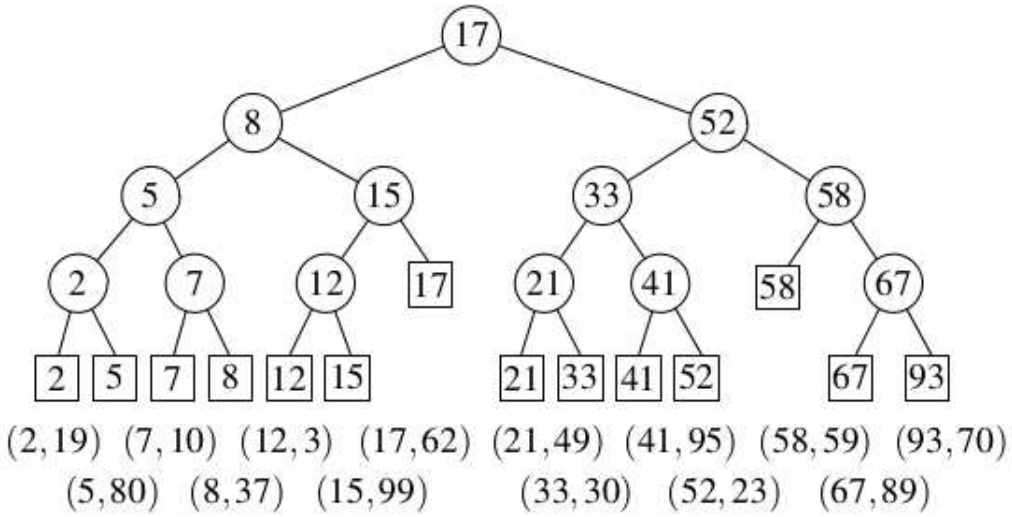


Figura 1.7: El árbol principal en un árbol de rangos por capas. Las hojas muestran sólo las coordenadas x. Abajo, los puntos almacenados. [DeBerg2008]

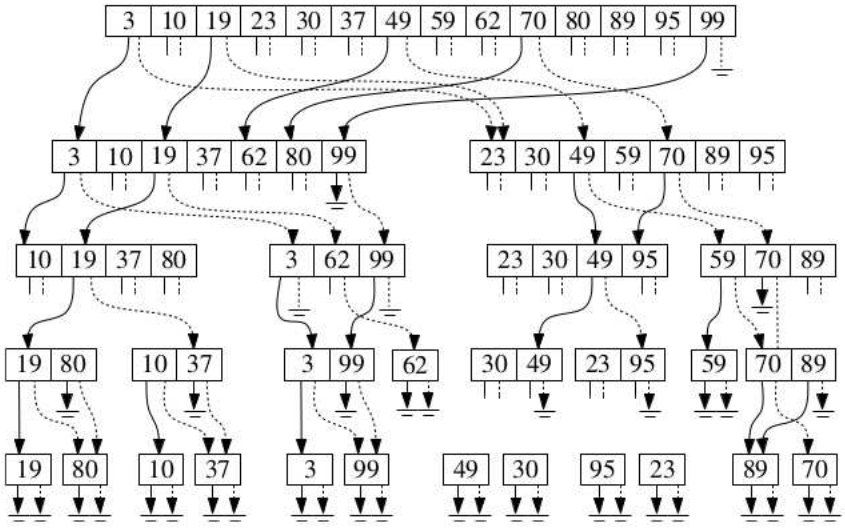


Figura 1.8: Los arreglos asociados al árbol de rangos por capas de la figura anterior. Los arreglos contienen las coordenadas y ordenadas, no se muestran todos los apuntadores. [DeBerg2008]

usando el árbol de rangos por capas. Como en el caso del árbol de rangos se busca a x y x' en T para determinar $O(\log n)$ nodos del árbol que contiene

los puntos cuya coordenada x está en el rango $[x : x']$. Sea v_{split} el nodo en que las busquedas por x y x' se separan. A partir de v_{split} estamos interesados en los nodos que son hijos derechos de nodos en los que la busqueda por x procedio a la izquierda y en los nodos que son hijos izquierdos de nodos en que la busqueda de x' procedió a la derecha. En v_{split} determinamos la entrada de $A(v_{split})$ cuya coordenada y es la menor que satisface a la vez ser mayor o igual que y. Esto puede hacerse mediante busqueda binaria en tiempo $O(\log n)$. A partir de allí, conforme la busqueda de x y x' procede, mantenemos en todo momento registro de dichas posiciones en los arreglos de los nodos visitados utilizando los apuntadores en los arreglos asociados, lo cual puede realizarse en tiempo constante. De esta manera utilizaremos tiempo $O(1 + k_v)$ para reportar los puntos en cada nodo v de los $O(\log n)$ nodos que produjo la busqueda por el rango $[x : x']$, y utilizaremos en total tiempo $O(\log n + k)$ para reportar todos los puntos en el rango rectangular $[x : x'] \times [y : y']$.

En general la técnica de fraccionamiento en cascada puede aplicarse a cualquier número d de dimensiones, reduciendo el tiempo de busqueda para un árbol de rangos en tiempo logarítmico:

Teorema 3. *Sea P un conjunto de n puntos en el espacio euclidiano de d dimensiones, $d \geq 2$. Un árbol de rangos por capas para P usa espacio $O(n \log^{d-1} n)$ y puede ser construido en tiempo $O(n \log^{d-1} n)$. Usando dicho árbol es posible resolver consultas de rangos rectangulares en tiempo $O(\log^{d-1} n + k)$ donde k es el número de puntos reportados.*

1.4. Conteo de rangos triangulares en 2D

Sea Q un conjunto de n puntos en el plano. Para desarrollar un método para el conteo de puntos al arrastrar un segmento, los autores describen primero estructuras para el conteo y reporte de puntos en el semiplano dada una línea de consulta l. Esto es, dada la línea l, reportar cuantos y cuales puntos de Q se encuentran en uno de los semiplanos abiertos que l determina.

1.4.1. Rangos determinados por semiplanos

Sea Q^* el conjunto de los duales de los puntos de Q y $A(Q^*)$ el arreglo determinado por las líneas en Q^*. A partir de $A(Q^*)$ se construye una estructura de datos que almacena los niveles de $A(Q^*)$. Dicha estructura será llamada *estructura de niveles.*

Definición 1. *Un punto q en el plano dual está en el nivel $\theta, 0 \leq \theta \leq n$ si hay exactamente θ lineas de Q^* estríctamente abajo de q. El nivel-θ de $A(Q^*)$ es la cerradura de los puntos en las líneas de Q^* cuyo nivel es θ y se denota λ_θ.*

Las aristas de λ_θ forman una cadena que se extiende por el plano dual de $x = -\infty$ a $x = \infty$. Cada vértice de $A(Q^*)$ aparece en 2 niveles consecutivos, y cada arista de $A(Q^*)$ aparece en un único nivel. La figura 1.9 muestra un ejemplo de los distintos niveles en un arreglo.

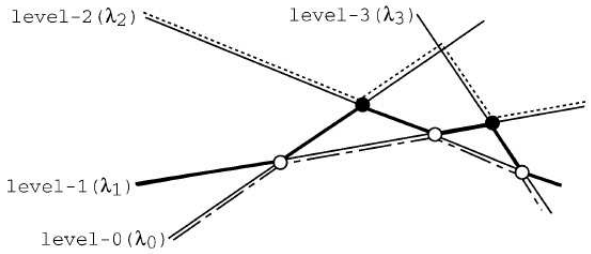


Figura 1.9: Niveles en un arreglo de lineas. [Goswami2004]

Definición 2. *La estructura de niveles es un arreglo A cuyos elementos corresponden a los niveles $\theta = 0, 1, 2, \ldots, n$ del arreglo $A(Q^*)$. Cada elemento del arreglo, el cual representa a un nivel θ, tiene un arreglo de los vertices de α_θ, ordenados de izquierda a derecha.*

Dada la línea de consulta l el problema del conteo para los semiplanos definidos por l se resuelve encontrando dos niveles consecutivos θ y $\theta+1$ tal que l^* está entre α_θ y $\alpha_{\theta+1}$. Dichos niveles se determinan mediante una busqueda binaria en 2 etapas en la estructura de niveles. De manera que dicha consulta puede ser realizada en tiempo $O(\log^2 n)$. Mejorando la estructura de niveles[2] mediante un procedimiento similar al fraccionamiento en cascada[3] es posible reducir el tiempo de consulta en un factor logarítmico, lo cual lleva al

Teorema 4. *Un conjunto de n puntos en el plano puede ser procesado usando tiempo y espacio $O(n^2)$ de manera que sea posible resolver consultas de conteo respecto a semiplanos determinados por una línea en tiempo $O(\log n)$. Si se requiere reportar el conjunto de puntos que dicha consulta cuenta, se requiere de tiempo adicional $O(k)$, donde k es el tamaño de la salida.*

1.4.2. Rangos triangulares

En este problema, dado un triángulo $\triangle$, el objetivo es resolver las consultas de conteo y reporte de puntos que se encuentran dentro de $\triangle$.

Sea $P = \{p_1, \ldots, p_n\}$ un conjunto de puntos en el plano. Una línea que divide a P en dos conjuntos no vacíos se denomina un *corte*. Si el corte divide a P en dos conjuntos P_1 y P_2 cuyas cardinalidades difieren en a lo más 1 se dice que el corte es *balanceado*. Para un conjunto P dado el corte balanceado puede no ser único. Para los propósitos de la exposición subsecuente, puede utilizarse cualquiera de ellos. Los conjuntos P_1 y P_2 son divididos recursivamente mediante cortes balanceados. El proceso continua hasta que todas las particiones tienen exactamente un elemento.

[2]S.C. Nandy, S. Das, P.P. Goswami, *An efficient k-nearest neighbors searching algorithm for a query line*, Theoret. Comput. Sci. 299 (2003) 273-288.

[3]B. Chazelle, L.J. Guibas,*Fractional cascading II. Applications*, Algorithmica 1 (1986) 163-191.

Parte del preprocesamiento para resolver las consultas de rangos triangulares consiste en elaborar una estructura $T(P)$, llamada *árbol de partición*, la cual está basada en la partición de P mediante cortes balanceados. El nodo raíz corresponde al conjunto P, sus 2 hijos a los conjuntos P_1 y P_2 y así recursivamente.

Cada nodo v de $T(P)$ tiene además (i) el conjunto de puntos P_v (ii) un campo entero χ_v indicando el tamaño de P_v (iii) la línea I_v que es el corte balanceado de P_v.

Definición 3. *Sea Q un conjunto de puntos dividido por un corte balanceado I en dos conjuntos Q_1 y Q_2. Un* corte de sandwich *de Q_1 y Q_2 es una linea que es un corte balanceado para Q_1 y Q_2.*

Para definir el árbol $T(P)$ de forma única se utilizan *cortes de sandwich* como los cortes balanceados de los nodos hijos de cada nodo interno de $T(P)$.

Lema 3. [4] *Sea P un conjunto de n puntos. El árbol de partición de P puede ser construido en tiempo $O(n \log n)$.*

Una vez se tiene $T(P)$ se añaden al mismo dos estructuras auxiliares, SS_1 y SS_2, las cuales ayudan a resolver, respectivamente, las consultas de cuenta y reporte.

Estructura auxiliar SS_1.

Sea v un nodo en $T(P)$. P_v^* es el conjunto de lineas que son los duales de los puntos de P_v. A partir de esta se crea la estructura de niveles aumentada $A(P_v^*)$ definida anteriormente y se añade al nodo v. Dicha estructura de niveles se aumentará usando el

Lema 4. *Sea v un nodo de $T(P)$ y w un nodo hijo de v. Entonces cualquier celda de $A(P_v^*)$ está contenida totalmente en una celda de $A(P_w^*)$.*

Utilizando dicho lema los autores aumentan la estructura $A(P^*)$ añadiendo a cada celda C dos apuntadores a las celdas $C_L \in A(P_u^*)$ y $C_R \in A(P_w^*)$ que contienen a dicha celda en los arreglos de líneas de u y w los cuales son, respectivamente, los hijos izquierdo y derecho de v. Los autores prueban el

Teorema 5. *El tiempo y espacio requerido para crear la estructura SS_1 para todos los nodos internos de $T(P)$ es $O(n^2)$.*

Estructura auxiliar SS_2

La estructura SS_2 se añade también a cada nodo v de $T(P)$. Suponga que v está en la i-ésima capa de $T(P)$. A v corresponde una región del plano R_v la cual está determinada por los cortes balanceados de los ancestros de v. De manera que la frontera de R_v tiene a lo más i lados. Dichas aristas se almacenan

[4] H. Edelsbrunner, E. Welzl, *Halfplanar range search in linear space and $O(n^{.695})$ query time*, Inform. Process. Lett. 23 (1986) 289-293.

en un arreglo. Sea I una arista en dicho arreglo, y π un punto en dicha arista. Si rotamos 180° una semirecta alrededor de π y dentro de R_v obtendremos una lista de los puntos en R_v según el orden en que la semirrecta incidió en ellos. Obtendremos semejante lista para todos los puntos en I en todos las aristas de la frontera de R_v. Observe que alrededor de π en I puede existir un intervalo en que dicha lista no cambia. Para determinar todos estos intervalos, una cada par de puntos en R_v mediante una recta. Estas rectas determinan intervalos en la frontera de R_v, a lo más $O(|P_v|^2)$ de tales intervalos. Dos de tales intervalos consecutivos difieren únicamente en 2 puntos, los cuales intercambian posición en las listas respectivas. De manera que es posible almacenar dichas listas en espacio $O(|P_v|^2)$. Utilizando una estructura de datos para listas casi identicas[5] es posible llegar al

Lema 5. *El tiempo y espacio requeridos para crear y almacenar la estructura SS_2 en todos los nodos de $T(P)$ es $O(n^2)$.*

Consulta de conteo para rangos triangulares

Para resolver la consulta se recorre el árbol $T(P)$. Se mantiene un contador global $COUNT$, inicializado a 0, el cual tendra al final del recorrido en número de puntos en el triángulo de consulta $\triangle$.

Al recorrer $T(P)$ si se alcanza una hoja el valor de $COUNT$ es aumentado en 1 si dicha hoja está dentro de $\triangle^*$. Si el recorrido pasa por un nodo interno v entonces su corte I_v puede dividir o no a $\triangle^*$. Acorde a ello se designa a v como un nodo divisor o un nodo no divisor.

Si v es un nodo no divisor entonces el recorrido de $T(P)$ procede por aquel de sus hijos que contiene a $\triangle^*$. Por otra parte, si v es un nodo divisor entonces $\triangle^*$ es dividida en 2 regiones de consulta, cada una de ellas convexa y de alguno de los siguientes tipos:

- tipo 0: esta región no contiene ninguna de las esquinas de $\triangle$.
- tipo 1: esta región contiene una esquina de $\triangle$.
- tipo 2: esta región contiene 2 esquinas de $\triangle$.

Al dividirse $\triangle$ por primera vez da origen a una región de tipo 1 y una región de tipo 2. En las divisiones sucesivas:

- Una región de tipo 2 puede dividirse en (i) una región de tipo 0 y una región de tipo 2, o bien (ii) dos regiones de tipo 1. En el caso (i) se realiza el conteo de puntos dentro de la región de tipo 0 en el hijo de v correspondiente (conteo descrito más adelante) y el recorrido de $T(P)$ procede en el hijo de v que contiene a la región de tipo 2. En el caso (ii) el recorrido procede recursivamente en ambos hijos de v.

[5]

- Una región de tipo 1 se divide en una región de tipo 0 y una región de tipo 1. Se realiza el conteo de puntos dentro de la región de tipo 0 y el recorrido procede recursivamente en el hijo de v que contiene a la región de tipo 1.

Los autores prueban el

Lema 6. *Durante todo el recorrido de $T(P)$ para la consulta determinada por $\triangle$ el número de regiones de tipo 0 generadas es $O(\log n)$.*

Conteo de puntos en una región de tipo 0 Los autores prueban el

Lema 7. *El número de esquinas de $\triangle$ que pueden aparecer en la frontera de una región de tipo 0 es a lo más 3.*

En la figura 1.10 pueden observarse las implicaciones del lema 6. La frontera de $\triangle$ está en líneas sólidas, los cortes de $T(P)$ en líneas punteadas.

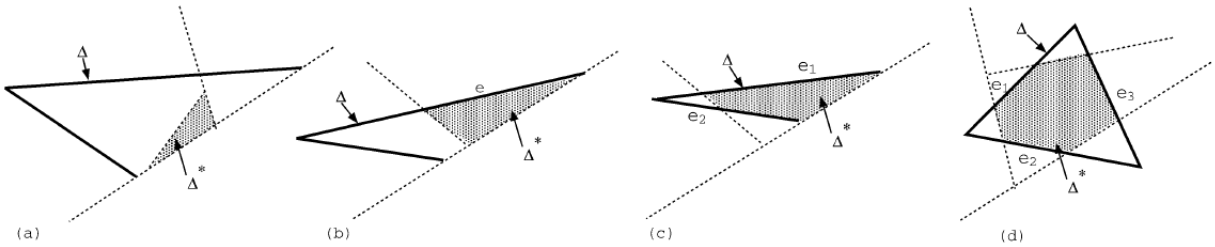


Figura 1.10: Las posibles regiones de tipo 0. [Goswami2004]

A continuación, los autores explican como realizar el conteo de puntos en la región 0 utilizando la cpmtep de rangos determinados por semiplanos. Hay 4 posibles casos, correspondientes al número de esquinas de $\triangle$ dentro de la región de tipo 0. En todos los casos el conteo puede obtenerse mediante conteo de rangos determinados por semiplanos. El lector interesado puede encontrar los detalles en el artículo original. Así se obtiene el

Lema 8. *El conteo de puntos en una región de tipo 0 puede realizarse en tiempo $O(\log n)$.*

Utilizando los lemas anteriores es posible realizar el conteo de rangos triangulares en tiempo $O(\log^2 n)$. En este punto los autores explican como mejorar dicho tiempo a $O(\log n)$ usando la estructura SS_1. Para ello sean l_1, l_2 y l_3 las líneas que incluyen a los lados de $\triangle$. Entonces se buscan las celdas en SS_1 que contienen a sus respectivos duales l_1^*, l_2^* y l_3^*. Al recorrer $T(P)$, conforme se pase de un nodo v a sus hijos, la celda correspondiente al punto en la estructura SS_1 es obtenida mediante los apuntadores a C_L y C_R en tiempo $O(1)$. Al requerirse un conteo de puntos en una región de tipo 0 dicha consulta puede responderse entonces en $O(1)$ pues lo único que se requiere es el nivel del punto dual (l_1^*, l_2^* o l_3^* según corresponda) en $A(P_v^*)$, lo cual requiere $O(1)$ pues dicho punto ya está en su celda correspondiente. De esta forma se llega al

Teorema 6. *Dado un conjunto de n puntos, estos pueden ser pre-procesados en tiempo y espacio $O(n^2)$ para responder a consultas de conteo de rangos triangulares en tiempo $O(\log n)$.*

Consulta de reporte de subconjunto en el rango triangular

Esta consulta también se realiza recorriendo $T(P)$. En cada nodo divisor, si la región es de tipo 1 o 2, la división se realiza de manera similar a la consulta de conteo. Al procesar una región de tipo 0 hay también 4 casos correspondientes a los tipos de región tipo 0 mencionadas arriba:

- En el caso 1 se reportan todos los puntos en P_v.
- En el caso 2 el subconjunto de P_v a uno de los lados de la arista e (uno de los lados de $\triangle$) se reporta usando la estructura SS_1.
- En el caso 3 la región de consulta en el nodo v está acotada por dos aristas, e_1 y e_2, de $\triangle$. Vea un ejemplo en la figura 1.11. La arista e_1 (respectivamente e_2 interseca R_v en los puntos α_1 y α_2 (respectivamente β_1 y β_2). La región de interés es dividida mediante la diagonal $\alpha_1\beta_2$ (línea punteada en la figura). Los puntos α_1 y β_2 están respectivamente los lados I y J de R_v. Mediante búsqueda binaria se determina su correspondiente intervalo en ambos lados, y se utilizan las listas de puntos en SS_2 correspondientes a dichos intervalos para reportar los puntos en las regiones de interés (las regiones angulares sombreadas de la figura)[6]. La complejidad total de este procedimiento es $O(\log n + \kappa)$ donde κ es el número de puntos reportados.
- En el caso 4 la región de consulta está acotada por 3 aristas e_1, e_2 y e_3. El recorrido procederá recursivamente en ambos hijos de v en $T(P)$. Si v es un nodo divisor, puede generar a lo más una región de consulta acotada por 3 aristas. El recorrido continúa por dicha región. La otra región generada por la división está acotada por 1 o 2 aristas, que se tratan como los casos anteriores. Este caso ocurre a lo más $O(\log n)$ veces y requiere entonces $O(\log^2 n + \kappa)$ tiempo.

De esta forma se obtiene el

Teorema 7. *Dados n puntos en el plano, estos pueden ser pre-procesados en tiempo y espacio $O(n^2)$ para lograr responder a consultas de reporte de subconjuntos mediante rangos triangulares en tiempo $O(log^2 + \kappa)$.*

Consulta de arrastre de segmento

Así se llega finalmente a la exposición del resultado buscado. Sea $\sigma = [\alpha, \beta]$ un segmento con extremos α y β. La consulta de arrastre del segmento σ consiste

[6]Una descripción completa del procedimiento se encuentra en: R. Cole, *Searching and storing similar lists*, J. Algorithms 7 (1986) 202-230.

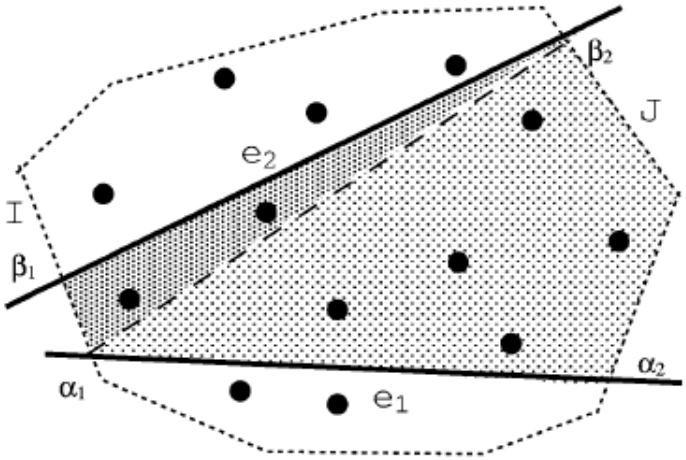


Figura 1.11: Ejemplo de consulta a SS_2. [Goswami2004]

en reportar los primeros k puntos que σ tocaría al ser arrastrada perpendicularmente hacia arriba o abajo. k es una constante especificada al momento de realizar la consulta.

Sea C_σ un corredor determinado por dos lineas paralelas L_1 y L_2 que pasan respectivamente por los puntos α y β y son perpendiculares a σ. Los puntos dentro del corredor son dividios en dos conjuntos, P_{above} y P_{below}, por σ. Sea l_σ la línea que contiene al segmento σ, y suponga que l^*_σ se ubica entre los niveles λ y $\lambda + 1$ de $A(P^*)$. Si $\lambda < k$ entonces σ incide en a lo más $\lambda < k$ puntos al ser arrastrado hacia arriba, de manera que es necesario reportar todos los puntos en P_{above}. Si $\lambda > k$ entonces es necesario encontrar otro segmento $\hat{\sigma}$ paralelo a σ que toda las dos fronteras de C y tal que el número de puntos dentro de la región R, determinada por $\sigma, \hat{\sigma}, L_1$ y L_2, es k. De manera que la consulta requiere 2 pasos: (i) determinar $\hat{\sigma}$ y (ii) reportar los puntos en R.

Para determinar $\hat{\sigma}$ trace un rayo vertical desde l_σ hacia abajo. Sea $e \in A(P^*)$ una arista de nivel θ, $\theta < \lambda - k$, a la que dicho rayo cruza. Sea $p*$ la línea que contiene a e. Trazamos una linea paralela a σ por el punto p, y sea $\hat{\sigma}$ el segmento de dicha línea comprendido entre L_1 y L_2. Esto define una región acotada por $\sigma, \hat{\sigma}, L_1$ y L_2 a la que denotaremos R_θ, donde κ_{theta} es el número de puntos en R_θ, el cual será computado dividiendo R_θ mediante una diagonal y aplicando conteo de rangos triangulares a los dos triángulos resultantes. Vea la figura 1.12.

Los autores prueban el

Lema 9. *Sean e_i y e_j dos aristas a niveles i y j, $i < j$, y R_i y R_j las respectivas regiones determinadas como se detalló arriba. Sean κ_i y κ_j los puntos dentro de dichas regiones. Entonces $\kappa_i > \kappa_j$*

El anterior lema permite aplicar busqueda binaria en $A(P^*)$ para determinar a $\hat{\sigma}$. De manera que tenemos el

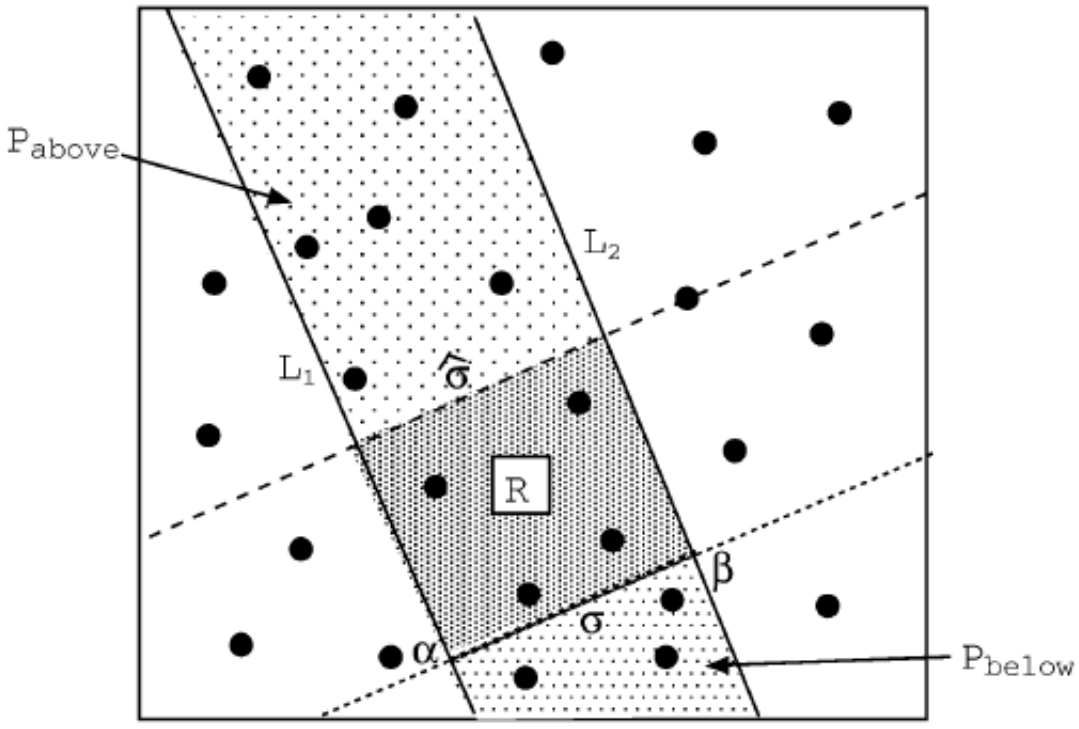


Figura 1.12: Consulta de arrastre de segmento. [Goswami2004]

Lema 10. *Es posible determinar al rectángulo R arriba del segmento σ que contiene k puntos en tiempo $O(\log^2 n)$.*

Y finalmente hay que reportar los puntos dentro de R, lo que añade tiempo k y da como resultado el

Teorema 8. *Un conjunto de n puntos en el plano puede ser pre-procesado en tiempo y espacio $O(n^2)$ de forma que se puedan responder consultas de arrastre de segmentos en tiempo $O(k + \log^2 n)$, donde k, el número de puntos, es un parámetro de entrada a la consulta.*

1.5. Cierre convexo dinámico: algoritmo de Jakob

El cierre convexo de un conjunto de puntos en el plano es uno de los objetos más importantes en la geometría computacional. Se define como un polígono convexo cuyos vértices son puntos del conjunto y tal que todos los puntos del conjunto dado se encuentran dentro del polígono. Computar el cierre convexo de n puntos estáticos puede lograrse en tiempo óptimo utilizando métodos como el

sondeo de Graham[7] o la variante del mismo basada en el barrido con una línea vertical de Andrew[8]. Existen también algoritmos sensibles a la salida[9].

En la versión dinámica del problema el conjunto de puntos S cuyo cierre convexo debe computarse cambia mediante inserciones y eliminaciones de puntos. La inserción o eliminación de un único punto puede modificar el cierre convexo en $|S|-2$ puntos, por lo que en muchas aplicaciones mantener el cierre dinámico resulta poco eficiente. En vez de eso se usan estructuras de datos que permiten resolver consultas específicas, por ejemplo: el punto extremo del cierre convexo en una dirección dada $\vec{a}$, las tangentes al cierre convexo que pasan por un punto dado p_e, determinar si un punto dado p_c está dentro del cierre convexo, determinar las aristas del cierre convexo intersecadas por una línea dada l_b, determinar los puentes (aristas comunes) con otro cierre convexo C. Estas consultas se ilustran en la figura 1.13. También es deseable en ciertas aplicaciones consultar o contar una secuencia de puntos consecutivos en el cierre convexo.

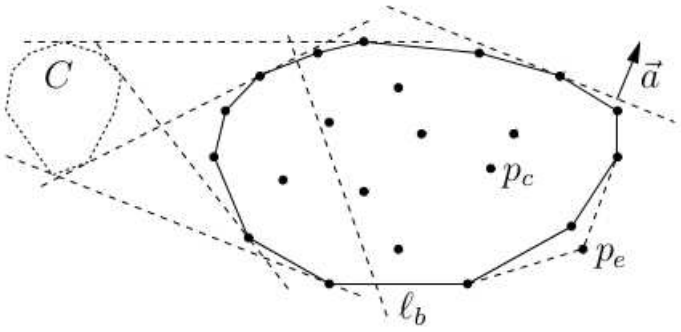


Figura 1.13: Algunas consultas relacionadas al problema del cierre convexo dinámico.*Fuente [Brodal2003]*

En su tesis doctoral Riko Jakob desarrolló una estructura de datos para estos problemas, con la cual se logra el

Teorema 9. *Existe una estructura de datos para el problema del cierre convexo dinámico con la cual es posible realizar inserciones y eliminaciones de puntos en tiempo amortizado $O(\log n)$, y consultas de puntos extremos, tangentes y puntos vecinos en tiempo $O(\log n)$, donde n es el tamaño del conjunto de puntos antes de realizar la operación. El espacio utilizado es $O(n)$.*

[7]R. L. Graham, *An efficient algorithm for determining the convex hull of a finite planar set*, Information Processing Letters 1 (4) (1972) 132-133.

[8]A. M. Andres, *Another efficient algorithm for convex hulls in two dimensions*, Information Processing Letters 9 (5) (1979) 216-219.

[9]D. G. Kirkpatrick, R. Seidel *The ultimate planar convex hull algorithm?*, SIAM J. Comput. 15 (1) (1986) 287-299.

T. M. Chan, *Optimal output-sensitive convex hull algorithms in two and three dimensions*, Discrete Comput. Geom. 16 (4) (1996) 361-368, 11th Anual Symposium on Computational Geometry (Vancouver, BC, 1995).

Los detalles de la estructura son complejos, el lector interesado puede encontrar un resumen en [Brodal2003][10].

[10] El documento completo del resultado es: Riko Jakob, *Dynamic planar convex hull*. Ph. Dissertation. BRICS dissertation series. May 2002. Accesible en red en `http://www.brics.dk` y `ftp://ftp.brics.dk`

Capítulo 2

Antecedentes no cromáticos

En este capítulo haremos un repaso de los resultados que involucran corredores vacíos en conjuntos de puntos no coloreados: el l-corredor, el corredor de una esquina y los corredores k-densos. En todos los casos los corredores son vacíos, esto es, no contienen en su interior puntos del conjunto de entrada.

2.1. L-corredor vacio: algoritmo de Cheng

Dado un conjunto de puntos no coloreados, Cheng da un algoritmo de tiempo $O(n^3)$ para determinar un L-corredor vacío. Para ello se prueba que es suficiente considerar L-corredores que pueden ser determinados por 3 puntos y una deformación rotacional computable en $O(1)$ (amortizado).

Un *corredor de 1-eslabón* o simplemente un *corredor* a través de un conjunto de puntos S es la region abierta del plano acotada por dos líneas paralelas que intersecan el cierre convexo de S. Vea la figura 2.1. El *ancho* del corredor es la distancia entre las 2 rectas que lo limitan. El problema de computar un corredor vacío de ancho máximo en un conjunto de puntos S puede ser resuelto en $O(n^2)$ [1].

Un *eslabón* de un L-corredor está formado por 2 rayos paralelos y un segmento de recta, los cuales forman un trapezoide no acotado. Vea la figura 2.2. Un L-corredor está formado por la unión de 2 eslabones perpendiculares. Los eslabones no necesitan ser paralelos a los ejes coordenados (por lo que nos referiremos al problema como *no orientado*). El *ancho* de un eslabón se define como la distancia entre los rayos paralelos que lo forman y el *ancho del L-corredor* se define como el menor de los anchos de sus eslabones.

[1] M. Houle, A. Maciel. *Finding the widest empty corridor through a set of points.* Tech. Rept. TR SOCS-88.11. Dept. of Computer Science, McGill University, Montreal, Canada, 1988.

R. Janardan, F. Preparata. *Widest corridor problems.* Tech. Rept. TR 93-17. Dept. of Computer Science, University of Minnesota (Twin cities), 1993; also in *Proc. 5th Canadian conf. on Computational Geometry (1993) 426-431.*

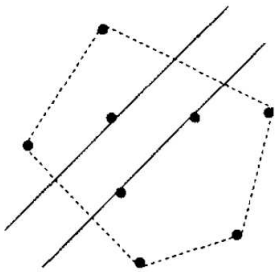


Figura 2.1: Un corredor de 1-eslabón. *[Cheng1996]*

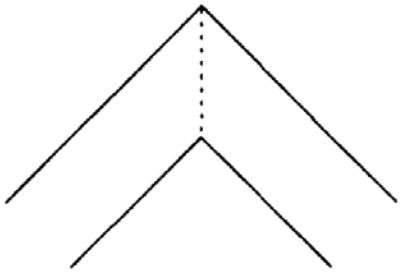


Figura 2.2: Eslabones de un L-corredor. *[Cheng1996]*

Para evitar situaciones en que el L-corredor tan solo "araña" el cierre convexo de los puntos S que constituyen el ejemplar del problema (vea figura 2.3) es necesario que las 2 zonas que resultan al remover el L-corredor del plano contengan puntos de S.

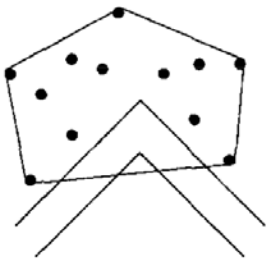


Figura 2.3: Un L-corredor que únicamente *araña* al cierre convexo de S. *[Cheng1996]*

2.1.1. El algoritmo

Si uno de los eslabones de un L-corredor de ancho máximo puede ser extendido a un corredor de 1-eslabón vacío, entonces el ancho del L-corredor más ancho es igual al ancho de dicho corredor de 1-eslabón vacío (ver figura 2.4). Por ello el primer paso del algoritmo de Cheng es determinar en tiempo $O(n^2)$ un corredor de 1-eslabón de ancho máximo y a adir despues un eslabón de ancho mayor o igual fuera del cierre convexo de S.

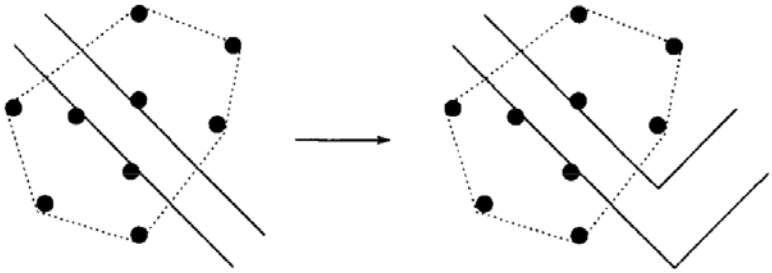


Figura 2.4: Formando un L-corredor a partir de un eslabón expandible a un corredor vacío. *[Cheng1996]*

Entonces es posible asumir que los eslabones del L-corredor de ancho máximo no son extendibles a un corredor de 1-eslabón vacío. La frontera del corredor que contiene una esquina convexa (respectivamente cóncava) respecto al interior del corredor es denominada la frontera *exterior* (respectivamente interior). Cada una de dichas fronteras está formada por dos rayos, los cuales serán denominados *piernas* del corredor. Un L-corredor sera llamando *no expandible* si cada pierna contiene un punto de S y cada pierna en la frontera exterior contiene un punto de S en su interior relativo.

A continuación Cheng demuestra que para cada L-corredor vacío C hay un L-corredor no expandible de ancho mayor o igual al de C.

El conjunto de puntos $S = \{p_i | i = 1, \ldots, n\}$ determina $n(n-1)$ vectores de un punto del conjunto a otro. El algoritmo hace girar el eje y en el sentido del reloj, y examina cada uno de estos vectores en el orden en que determinado al volverse dicho vector paralelo a la dirección positiva del eje y en su rotación. Dado un vector $p_2\vec{p}_1$ el efecto de rotar el eje y positivo hasta ser paralelo a dicho vector es similar a rotar el plano en sentido contrario al reloj hasta que $p_2\vec{p}_1$ apunte verticalmente hacia arriba. Sea x_0 la coordenada x del vector.

Para la ejecución del algoritmo se requieren 6 tipos de consultas, cada una ejecutable en tiempo $O(1)$ amortizado:

1. Para cada p_i obtener los 2 puntos que serían alcanzados primero si un rayo horizontal que parte a la derecha de p_i fuera rotado en y contra el sentido del reloj. Vea figura 2.5

2. Dado p_i obtener los 2 puntos que serían alcanzados primero si un rayo vertical que parte hacia arriba desde p_i fuera rotado en y contra el sentido

del reloj.

3. Dado p_i obtener p_j tal que $x(p_j) > x_0$, $y(p_j) > y(p_i)$ y $x(p_j)$ es mínimo. Vea figura 2.6. Análogamente obtener, dado p_i, p_J tal que $x(p_j) > x_0$, $y(p_j) > y(p_i)$ y $y(p_j)$ mínimo.

4. Dado p_i obtener p_j tal que $x(p_j) < x_0$, $y(p_j) > y(p_i)$ y $x(p_j)$ es máximo.

5. Sea $circle(p_i, p_j)$ el circulo cuyo diámetro es $p_i\bar{p}_j$. Dados p_i y p_j tales que $x(p_i) \geq x(pj)$ y $y(p_i) \leq y(p_j)$, obtener los dos puntos dentro o en $circle(p_i, p_j)$ que serían alcanzados primero por un rayo horizontal que sale de p_i hacia la izquierda y rota respectivamente en y contra el sentido del reloj. Vea la figura 2.7.

6. Dado p_i y p_j tales que $x(p_i) \geq x(p_j)$ y $y(p_i) \leq y(p_j)$ obtener los los puntos dentro o en $circle(p_i, p_j)$ que serían alcanzados primero si un rayo vertical que sale de p_J hacia abajo fuese rotado respectivamente en y contra el sentido del reloj.

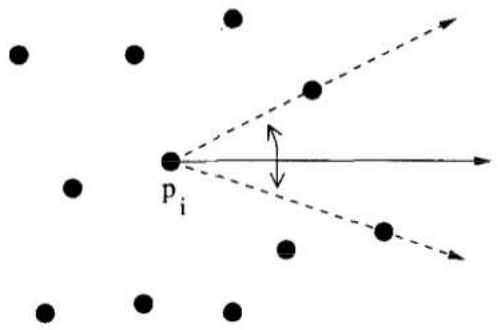


Figura 2.5: Consulta tipo 1. *[Cheng1996]*

Las técnicas utilizadas para responder los 6 tipos de consultas a tiempo amortizado $O(1)$ son:

Consultas 1 y 2 Para cada punto p_k se almacenan los restantes puntos de S en una lista circular doblemente ligada $list(p_k)$, la cual está ordenada angularmente respecto a p_k. Computar estas listas para cada punto de S puede realizarse en $O(n^2 \log n)$. Al realizar la primera consulta de tipo 1 sobre p_k se realiza una busqueda líneal en la lista $list(p_k)$ y se almacena la posición en que se halló la respuesta. El algoritmo visita los restantes puntos de acuerdo al orden angular respecto a p_k sin retroceder, de manera que si se responden las consultas siguientes a través de busqueda líneal a partir de la última posición visitada se logra el costo amortizado de $O(1)$.

Consultas 3 y 4 Denotemos $S_{p_2\vec{p}_1}$ al conjunto S rotado en sentido contrario a las manecillas del reloj de forma que $p_2\vec{p}_1$ apunte hacia arriba. Si $S_{p_2\vec{p}_1}$

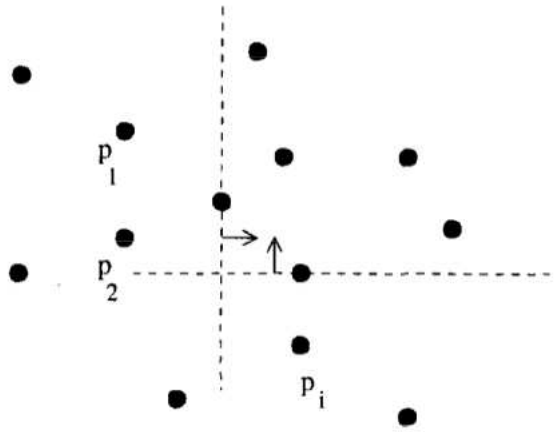


Figura 2.6: Consulta tipo 3. *[Cheng1996]*

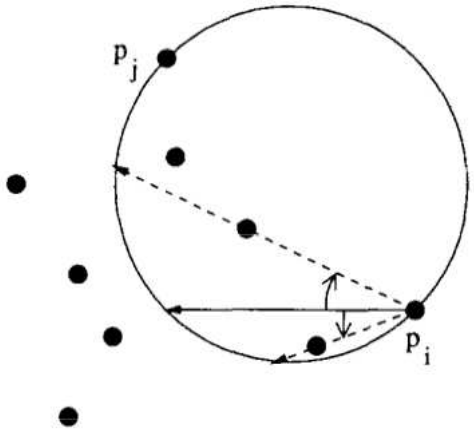


Figura 2.7: Consulta tipo 5. *[Cheng1996]*

estuviera ordenado según las coordenadas y de los puntos en forma no decreciente, podriamos, mediante busqueda líneal, resolver las consultas de tipo 3 y 4 para cada punto p_k encontrado durante la busqueda en $O(1)$ amortizado. De forma que el tiempo total para responder dichas consultas para todos los vectores sería $O(n^3)$. Sin embargo, si ordenaramos cada $S_{p_2\vec{p}_1}$ el tiempo ejecución total sería $O(n^3 \log n)$. Para evitarlo, Cheng observa que S induce $O(n^2)$ segmentos distintos entre cualquiera 2 puntos del conjunto. Sea $segment(S)$ la lista de estos segmentos, ordenada según la pendiente en $O(n^2 \log n)$. A continuación ordene S según la coordenada y y almacene el resultado en un árbol de busqueda persistente T. Cheng hace notar que, conforme el conjunto S es rotado, al encontrar cada segmento solo los 2 puntos que forman dicho segmento cambian de orden en

T. Dicha operación puede realizarse en T de forma persistente y sin necesidad de rebalancear T, de manera que, en total, el ordenamiento según la coordenada y de cada $S_{p_2\vec{p}_1}$ puede obtenerse en tiempo total amortizado $O(n^2)$.

Consultas 5 y 6 Para cada par de puntos p_i y p_j computamos una lista circular doblemente ligada $cir\_list(p_i, p_j)$ (respectivamente $cir\_list(p_j, p_i)$) de los puntos en $circle(p_i, p_j)$ (respectivamente de los puntos en $circle(p_j, p_i)$) ordenados según su ángulo respecto a p_i (respectivamente p_j). Estas listas pueden obtenerse de $list(p_i)$ (respectivamente $list(p_j)$), incluyendo un punto de $list(p_i)$ (respectivamente p_j) en $cir\_list(p_i, p_j)$ (respectivamente $cir\_list(p_j, p_i)$) sii dicho punto se encuentra dentro de $circle(p_i, p_j)$. La forma en que estás listas son recorridas por el algoritmo es similar a la utilizada en las listas $list(p_i)$, de manera que el costo amortizado es también $O(1)$.

El algoritmo computa 4 tipos de L-corredores candidatos:

Tipo 1 p_1 y p_2 estarán en la frontera exterior. Vea figura 2.8. Para cada p_k con $y(p_k) \leq y(p_2)$ y $x(p_k) \geq x_0$, sea la frontera exterior del candidato la L determinada por p_1, p_2 y p_k. Luego encontramos mediante consultas de tipo 3 los otros 2 puntos que determinan la frontera interior del corredor.

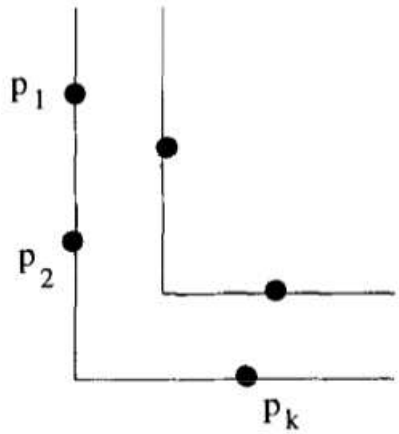


Figura 2.8: Candidato tipo 1. *[Cheng1996]*

Tipo 2 Por cada candidato de tipo 1 se construye un candidato de tipo 2 mediante deformaciones rotacionales. Para ello sea p_k el punto en la pierna horizontal exterior, y p_i y p_j los puntos en las piernas vertical y horizontal interiores. Vea la figura 2.9. Primero rotamos el corredor en sentido del reloj, manteniendo los puntos p_2 y p_k en la frontera exterior del mismo, y a los puntos p_i y p_j en la frontera interior. La rotación se detiene tan pronto como una de las fronteras alcanza a un punto más de S. Mediante consultas

1 (respectivamente 2) podemos determinar el ángulo θ_1 (respectivamente θ_2) que las piernas horizontales (respectivamente verticales) que pasan por p_k y p_j (respectivamente p_2 y p_i) pueden rotar antes de incidir sobre otro punto de S. El ángulo θ_3 (respectivamente θ_4) que la esquina de la frontera exterior entre p_2 y p_k (respectivamente entre p_i y p_j) puede rotar antes de incidir sobre otro punto de S puede determinarse mediante una consulta de tipo 5 (respectivamente de tipo 6). De manera que la rotación posible es mín$\{\theta_i | i = 1, 2, 3, 4\}$. Sea C_1 el corredor obtenido. Obtenemos de forma similar el corredor C_2 rotando al candidato 1 en sentido contrario al reloj, y hacemos que el candidato de tipo 2 sea el más ancho entre C_1 y C_2.

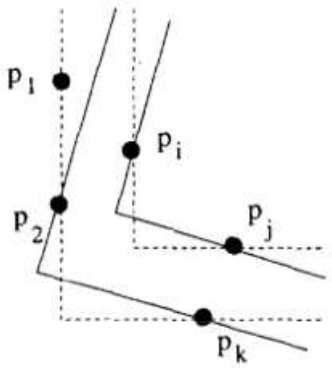


Figura 2.9: Candidato tipo 2. *[Cheng1996]*

Tipo 3 Este caso trata candidatos de tipo 1 degenerados, a los cuales aplica una deformación rotacional. El primer paso consiste en construir un corredor de tipo 1 degenerado. Para ello sea la frontera exterior del candidato la L que pasa por p_1, p_2 y p_k. Use una consulta de tipo 3 para encontrar el punto con menor coordenada y en la región abierta convexa acotada por la frontera exterior. Sea p_j dicho punto o p_1, dependiendo de cual tenga menor coordenada y, y sea la frontera interior del candidato la L que pasa por p_1 y p_j. Aplique una deformacion rotacional en el sentido contrario al reloj (Ver figura 2.10.a). Mantenga a p_2 y p_k (respectivamente p_1 y p_j) en la frontera exterior (respectivamente interior) y detenga la rotación tan pronto alguna pierna incida en otro punto de S. Sea C_1 el corredor más ancho así obtenido. Si $y(p_j) \geq y(p_2)$ entonces el candidato 3 es C_1. En otro caso, aplique una rotación en sentido del reloj al candidato, manteniendo a p_1 y p_k (respectivamente p_2 y p_j) en la frontera exterior (respectivamente interior), terminando la rotación tan pronto alguna de las piernas incida en otro punto de S (Ver figura 2.10.b). Sea C_2 el corredor obtenido, y entonces el candidato de tipo 3 es el más ancho de C_1 y C_2.

Tipo 4 Los candidatos de este tipo tienen a p_1 y p_2 en la frontera interior. Para p_k tal que $y(p_k) \leq y(p_2)$ encuentre mediante una consulta de tipo 4

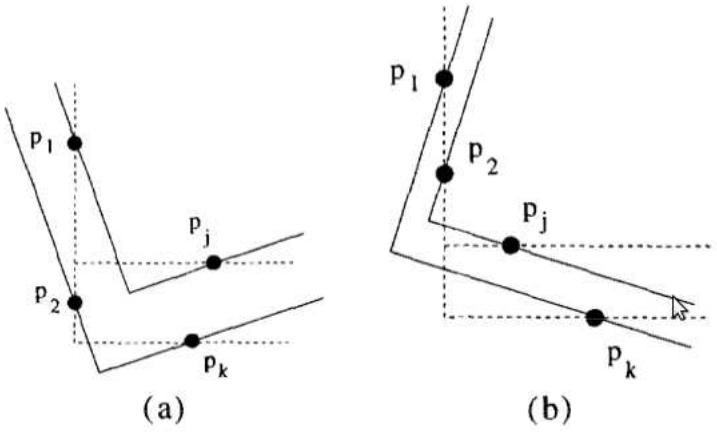


Figura 2.10: Candidato tipo 3. *[Cheng1996]*

el punto p_j de coordenada x máxima tal que $x(p_j) < x_0$ y $y(p_j) > y(p_k)$. Si $x(p_j) \leq x(p_k)$ entonces la frontera exterior del candidato es la L que pasa por p_j y p_k. Vea la figura 2.11. En otro caso, aborte (pues p_k no puede estar en la frontera exterior del candidato). Si p_k está en la frontera exterior, use una consulta de tipo 3 para hallar p_i, el cual tiene coordenada y mínima en la región convexa abierta determinada por la frontera exterior del candidato, y sea la frontera interior del candidato la L que pasa por p_1, p_2 y p_i.

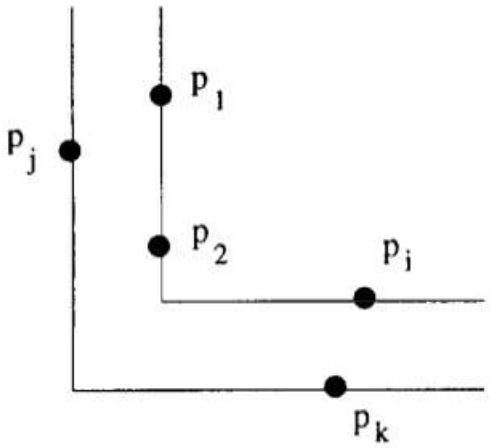


Figura 2.11: Candidato tipo 4. *[Cheng1996]*

Tipo 5 Los candidatos de este tipo son obtenidos mediante deformaciones rotacionales de candidatos de tipo 4. Sean p_1, p_2 y p_i los puntos en la frontera

del candidato de tipo 4, y p_j y p_k los puntos en la frontera exterior, como en la figura 2.11. Rote al corredor en el sentido del reloj (respectivamente en contra) mientras p_1 y p_i se mantienen en la frontera interior (respectivamente p_2 y p_i). La frontera exterior pasa en ambos casos por p_j y p_k. La rotación termina tan pronto como alguna pierna incide en otro punto de S. De manera semejante a los candidatos de tipo 2, estos candidatos pueden obtenerse en $O(1)$.

Finalmente Cheng demuestra que un corredor vacío de ancho máximo se encuentra entre los candidatos examinados (Teorema 3 de [Cheng1996]), de manera que el problema puede ser resuelto en $O(n^3)$.

2.2. Corredor de una esquina vacio

2.2.1. El algoritmo de Díaz-Bañez, López y Sellarès

Un *corredor* $c(l, l')$ es la regíon del plano acotada por dos líneas paralelas l y l'. El ancho de c es la distancia $d(l, l')$ entre las 2 líneas que lo acotan. Un *eslabón* es la región abierta determinada por dos rayos paralelos, $r(L) = p + \vec{v}t$ y $r'(L) = p' + \vec{v}t$, y un segmento apierto $s(L) = pp'$, los cuales forman un trapezoide no acotado. Los puntos de $s(L)$ pertenecen al eslabón, no asi los de $r(L)$ y $r'(L)$. El ancho del eslabón, denotado $\omega(L)$ es la distancia entre los rayos $d(r(L), r'(L))$.

Un corredor de una esquina C es la unión de 2 eslabones L y L' que comparten únicamente el segmento $s(L) = s(L')$. C es una región abierta con una *frontera exterior*, la cual contiene una esquina convexa respecto al interior del corredor, y una *frontera interior*, la cual contiene una esquina concava. Cada una de las fronteras consta de dos rayos, los cuales se denominan *piernas de la frontera*. Denotaremos $r(L)$ y $r(L')$ (respectivamente $r'(L)$ y $r'(L')$) a las piernas de la frontera exterior (respectivamente interior). El ancho del corredor C, denotado $\omega(C)$, es el menor de los anchos de los dos eslabones. El ángulo $\alpha(C)$, $0 \leq \alpha \leq \pi$ del corredor $C = (L, L')$ es el ángulo determinado por los rayos $r(L)$ y $r(L')$.

Sea S un conjunto de n puntos en el plano. Un corredor c que interseca el cierre convexo $CH(S)$ de S es vacío si no contiene puntos de S. Un corredor vacío debe intersecar a $CH(S)$ pues de otra forma el corredor vacío más ancho no está bien definido. Un corredor de una esquina C es vacío si no contiene puntos de S y, al quitar a C del plano, divide al mismo en 2 regiones no acotadas, cada una de las cuales contiene al menos un punto de S. Note que en el caso de corredores con una esquina no basta con que C interseque a $CH(S)$, pues esto permitiría corredores que simplemente .ªra an.el exterior de S sin pasar en realidad .ª través"de S.

Dado que un corredor vacío C puede ser considerado un corredor de una esquina con $\alpha(C) = \pi$, el corredor de una esquina que el algoritmo de [DiazBanez2006] hallará es al menos tan ancho como un corredor vacío. Este caso puede deter-

minarse en tiempo $O(n^2)$, de suerte que el algoritmo considera únicamente los casos $0 < \alpha(C) < \pi$.

Es claro que existe un corredor óptimo C que contiene al menos un punto de S en cada pierna, pues de otra forma el ancho de uno o ambos eslabones puede ser aumentado separando las piernas de uno o ambos eslabones hasta que incidan en un punto de S.

Se dice que un eslabón es *localmente de ancho máximo* si cada pierna contiene al menos un punto de S y no es posible aumentar el ancho del eslabón rotando las piernas alrededor de los puntos de S en que inciden. Desde luego, semejante rotación debe aplicarse a ambas piernas de manera que continuen paralelas. A partir de esta observación obtenemos el

Lema 11. *Un eslabón L es localmente de ancho máximo sii satisface alguna de las condiciones siguientes:*

1. *Existen puntos p_1 y p_2 de S en la pierna exterior $r(L)$ y un punto $q \in S$ en la en la pierna interior $r'(L)$ tal que los ángulos qp_1p_2 y qp_2p_1 son ambos agudos.*

2. *Existen puntos $p_1, p_2 \in S$ en la pierna interior $r'(L)$ y un punto $q \in S$ en la pierna exterior $r(L)$ tal que los ángulos qp_1p_2 y qp_2p_1 son ambos agudos.*

3. *Existen dos puntos $p, q \in S$ en las piernas interior y exterior de L, respectivamente, y tales que el segmento pq es ortogonal a ambas piernas.*

A partir del anterior lema es posible determinar que solo existen 6 tipos de corredores de ancho máximo, los cuales serán denominados $(21, 21)$, $(21, 11)$, $(21, 12)$, $(12, 11)$, $(12, 12)$ y $(11, 11)$, dependiendo del tipo de los eslabónes que los forman. Vea la figura 2.12.

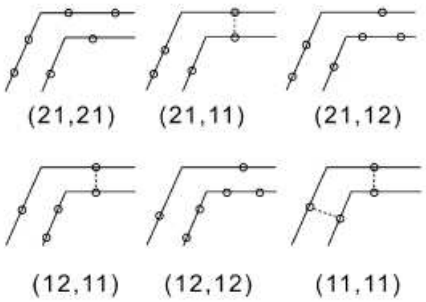


Figura 2.12: Los 6 tipos de corredores en el algoritmo de [DiazBanez2006]. *[DiazBanez2006].*

Los autores prueban también el siguiente

Lema 12. *Siempre existe un corredor óptimo de una esquina que es localmente de ancho máximo.*

Antes de desarrollar el algoritmo, los autores necesitarán otras definiciones. Sean p y q dos puntos. Denotaremos l_{pq} a la línea que pasa por ambos puntos. Sea $t \notin l_{pq}$. Denotamos $H_l^+(t)$ (respectivamente $H_l^-(t)$) al semiplano abierto determinado por l que contienen (respectivamente no contiene) a t. Si $l = l_{pq}$ escribimos simplemente $H_{pq}^+(t)$ (respectivamente $H_{pq}^-(t)$. Si l, m son dos rectas no paralelas y p, q dos puntos denotamos $S_{lm}^{++}(p,q)$ a los puntos de S en el interior de $H_l^+(p) \cap H_m^+(q)$. Las otras 3 regiones determinadas por l y m se denotan $S_{lm}^{--}(p,q), S_{lm}^{-+}(p,q)$ y $S_{lm}^{+-}(p,q)$.

Además se hace uso de la siguiente

Observación 2. *Sea H un semiplano acotado por una línea l y sea P un conjunto de m puntos en H. El punto de P más cercano a l es un vértice del cierre convexo $CH(P)$ de P. Si se tiene computado $CH(P)$ dicho punto puede determinarse en tiempo $O(\log m)$.*

Con base en las anteriores definiciones y observación, los autores presentan su algoritmo. Para simplificar la exposición, se asume que no existen 3 puntos de S colíneales. Dicha hipótesis puede ser desechada usando una técnica conocida como *simulation of simplicity*[2]. Los autores consideran varios casos:

Casos $(21,21), (21,11), (21,12)$**.**

Todos los casos en que al menos una pierna exterior contiene 2 puntos son tratables de forma similar. Asuma que el corredor óptimo tiene a los puntos p_1, p_2 en una pierna exterior y un punto q_1 en la otra. Es posible que $q_2 = p_2$, esto es, que la esquina convexa del corredor sea un punto de S que pertenece a ambas piernas exteriores. A partir de ello se calculan todos los corredores de tipos $(21,21), (21,11), (21,12)$ para cualesquiera tres puntos $p_1, p_2, q_1 \in S$.

El primer paso es computar, para cada $q_1 \in S$, el orden radial de $S \backslash \{q_1\}$ según una línea l que pasa por q_1 rota alrededor de q_1. La orientación inicial de l es arbitraria, los ángulos de rotación se encuentran en el rango $[0, \pi)$, lo que garantiza que cada punto es visitado una única vez.

Ahora, para 2 puntos $p_1, p_2 \in S \backslash \{q_1\}$ sea $m = l_{p_1 p_2}$ y $S' = S \cap H_m^+(q_1)$. El algoritmo considerará todos los corredores con p_1 y p_2 en una pierna exterior y q_1 en la otra, y recordará el ancho del máximo corredor encontrado. Los puntos de S' son visitados según el orden radial computado previamente. Al inicio del barrido $l = l_{q_1 p_2}$. El algoritmo procede en la dirección que cause a la intersección de $l_{p_1 p_2}$ y l alejarse más de p_1 (dicha dirección puede ser en o contra el sentido de las manecillas del reloj dependiendo de las posiciones de p_1, p_2 y q_1). Vea la figura 2.13 para un ejemplo. La etiquetas en los puntos indican en orden en que los puntos de S' son visitados por el barrido.

Conforme el barrido procede, comenzando de $l_{q_1 p_2}$ y hasta el ángulo $q_1 p_2 p_1$ el conjunto $P = S_{lm}^{++}(p_1, q_1)$ cambia. El algoritmo calcula en todo momento $CH(P)$ actualizándolo mediante inserciones y eliminaciones según los puntos de S' entren o salgan de $H_l^+(p_1)$.

[2]H. Edelsbrunner, E. P. Muecke. *Simulation of simplicity.* ACM Trans. Graphics 9 (1990) 66-104.

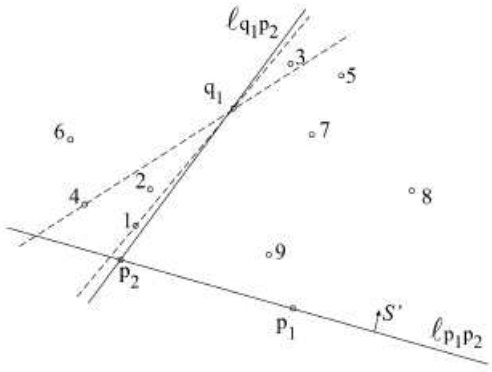


Figura 2.13: Ejemplo del orden en que son visitados los puntos. La línea sólida $l_{q_1p_2}$ marca el comienzo del barrido. *[DiazBanez2006].*

Los puntos de interés para el algoritmo son los puntos p' y q' de P más cercanos, respectivamente, a m y l. Una vez se determinan p' y q', dan origen a 3 posibles tipos de eventos (Vea la figura 2.14):

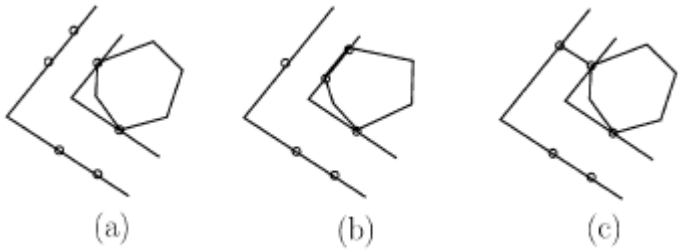


Figura 2.14: Eventos a considerar. *[DiazBanez2006].*

1. l pasa por un nuevo punto q_2 (inicialmente $q_2 = p_2$). En este caso obtenemos un corredor candidato de tipo $(21, 21)$. En la figura, inciso a.

2. El punto más cercano a l del conjunto P actual cambia. Al ocurrir esto l es paralela a un lado de $CH(P)$ y se obtiene un candidato de tipo $(21, 12)$. En la figura, inciso b.

3. El punto q' de P más cercano a l es tal que el segmento qq' es perpendicular a l. Esto produce un candidato de tipo $(21, 11)$. En la figura, inciso c.

En cuanto a la complejidad temporal del algoritmo tenemos que el tiempo necesario para llevara cabo el orden radial de los puntos es $O(n \log n)$. Al almacenar $CH(P)$ en una estructura que permita la busqueda binaria es posible computar todos los puntos más cercanos de $CH(P)$ a l mientras esta última

rota en tiempo total $O(n \log n)$. $CH(P)$ puede ser actualizada dinámicamente a un costo amortizado de $O(\log n)$ por inserción o eliminación. De forma que el tiempo total para la rotación y procesamiento de eventos es $O(n \log n)$. Puesto que hay $O(n^3)$ tripletas (p_1, p_2, q_1), el tiempo total para determinar candidatos de tipo $(21, 21), (21, 11)$ y $(21, 12)$ es $O(n^4 \log n)$.

Caso $(11, 11)$**.**

Sean p_1, p_2 y q_1 tres puntos de S. Denotemos l_m a la línea que pasa por p_1 perpendicular a $l_{p_1p_2}$ y suponga que el corredor de una esquina buscado tiene a p_1 en una pierna exterior y a q_1 en la otra. Como en el caso anterior, obtendremos todos los candidatos mediante una rotación radial de la línea que pasa por q_1. Para ello consideraremos l_m fija y rotaremos a $l = l_q1$ manteniendo a p_2 como el punto del $P = S^{++}_{lm}(p_1, q_1)$ actual más cercano a l_m. Vea la figura 2.15. Así, rotaremos l pero únicamente consideraremos eventos de tipo (c). Note que podemos comenzar la rotación con $l = l_{q_1p_1}$. La complejidad temporal de este caso es la misma que en el caso anterior.

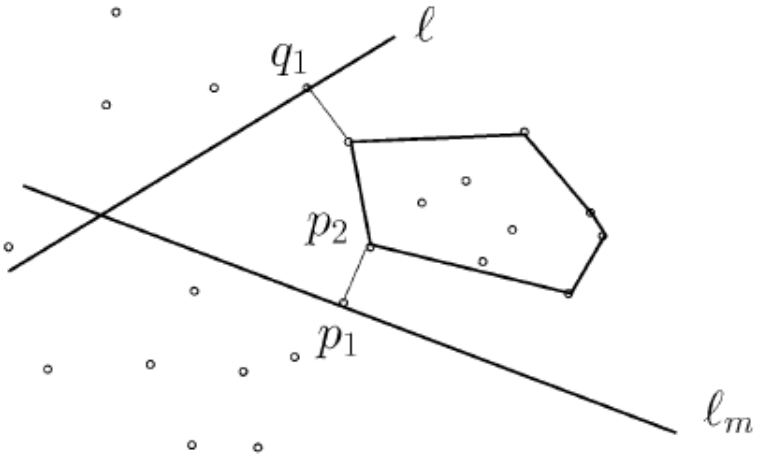


Figura 2.15: Caso (11,11). *[DiazBanez2006].*

Casos $(12, 12)$ **y** $(12, 11)$**.**

Este caso se procesa como el anterior, pero manteniendo 2 cierres convexos $P = S^{++}_{lm}(p_1, q_1)$ y $P' = S^{+-}_{lm}(p_1, q_1)$. Vea la figura 2.16. El punto p' de $S^{+-}_{lm}(p_1, q_1)$ más cercano a m determina una pierna del corredor candidato. La otra pierna se determina rotando la línea l alrededor de q_1 en los eventos de tipo (b) y (c), correspondientes respectivamente a los casos $(12, 12)$ y $(12, 11)$ (incisos (b) y (c) de la figura, respectivamente). Nuevamente p_1, p_2 y q_1 son fijos y $CH(P)$ y $CH(P')$ son actualizadas por el algoritmo, para una complejidad temporal similar a los 2 casos anteriores.

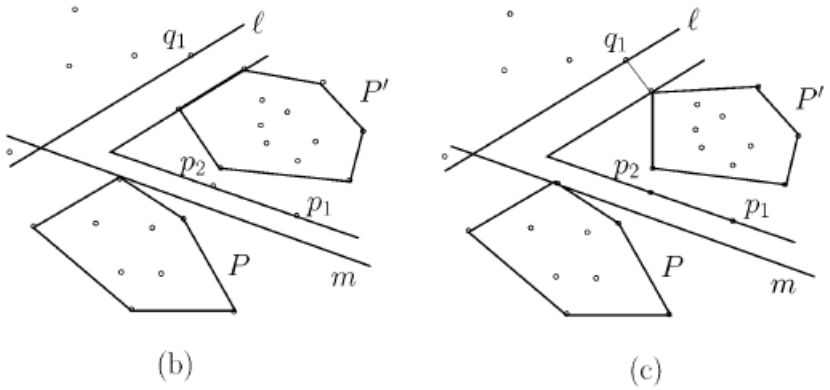


Figura 2.16: Casos (12,12) y (12,11). *[DiazBanez2006].*

Con lo cual tenemos el

Teorema 10. *Sea S un conjunto de n puntos. Un corredor cromático de una esquina de ancho máximo en S puede determinarse en tiempo $O(n^4 \log n)$.*

2.2.2. Mejoras de Das, Mukhopadhyay y Nandy

Sea $P = \{p_1, \ldots, p_n\}$ un conjunto de n puntos en el plano. Un *corredor* $C = (l, l')$ es la región abierta del plano acotada por 2 líneas paralelas l y l'. El ancho del corredor es la distancia entre las 2 líneas.

Un L-corredor es la concatenación de dos eslabones perpendiculares. Un *eslabón* $L = (l, l')$ es un trapezoide no acotado que no contiene puntos de P. Los lados paralelos del trapezoide están formados por las semirrectas l y l' que parten de los puntos p y p', de los 2 lados restantes uno está formado por el segmento $s(L) = [p, p']$, el cuarto lado es no acotado. Las semirrectas l y l' se denominan *piernas* del corredor, $s(L)$ se denomina la *base* de L. El ancho del eslabón L es la distancia perpendicular de l y l'. Un *corredor de una esquina* $C = (L_1, L_2)$ es la unión de 2 eslabones $L_1 = (l'_1, l''_1)$ y $L_2 = (l'_2, l''_2)$ con interior disjunto y que comparten la misma base, es decir $s(L_1) = s(L_2)$. De manera que un corredor de una esquina está acotado por dos cadenas paralelas, cada una de las cuales está formada por dos semirrectas o piernas. Las piernas l'_1 y l'_2 (respectivamente l''_1 y l''_2) definen la frontera externa (respectivamente interna) del corredor. La frontera externa (respectivamente interna) es convexa (respectivamente concava) con respecto al interior del corredor. El *ancho* $w(C)$ del corredor C es el más peque o de los anchos de sus dos eslabones. El *ángulo* $A(C)$ de C, $0 < A(C) \leq 2\pi$, es el ángulo determinado por l'_1 y l'_2 (o, equivalentemente, por l''_1 y l''_2). Un corredor de una esquina es vacío si no contiene a ningún punto de P y divide al plano en 2 regiones no acotadas, cada una de las cuales contiene al menos un punto de P.

Sea C un corredor que pasa a través del cierre convexo $CH(P)$ de P. Se dice

que el corredor es localmente de ancho máximo si cada una de las rectas que forman su frontera contiene al menos un punto de P y no es posible aumentar el ancho del corredor mediante rotaciones de las rectas sobre alguno de dichos puntos (manteniendo, por supuesto, ambas rectas paralelas). Los corredores de ancho máximo son caracterizados por el siguiente

Teorema 11. *Sea C un corredor de ancho máximo que atraviesa $CH(P)$ y sean las líneas l' y l'' las fronteras del corredor. Entonces sucede alguna de las siguientes:*

1. *Existen puntos $p_i, p_k \in P$ tales que l' pasa por p_i, l'' pasa por p_j y l' y l'' son ambas perpendiculares a la recta que pasa por p_i y p_j.*

2. *Alguna de las 2 líneas, por ejemplo l' pasa por dos puntos $p_i, p_j \in P$ y p'' pasa por otro punto $p_k \in P$.*

Denominaremos corredor de tipo 1 (respectivamente de tipo 2) a un corredor que satisface la primera condición (respectivamente la segunda condición). El número de corredores de ancho máximo y vacíos de tipo 1 (respectivamente de tipo 2) es $O(n)$ (respectivamente $O(n^2)$)[3].

Un *corredor cerrado por un extremo* es un eslabón $L = (l', l'')$ en el que cada una de las 2 piernas l' y l'' contiene al menos un punto de P. Si $p' \in l'$ y $p'' \in l''$ son dichos puntos entonces L es de ancho máximo entre todos los corredores cuyas fronteras pasan por dichos 2 puntos. Se dice que L está acotado por arriba (respectivamente por abajo) si el interior de L se encuentra a la derecha (respectivamente a la izquierda) del segmento dirigido $\overrightarrow{pq}$. El segmento pq es la base de L.

Los corredores cerrados por un extremo pueden también ser clasificados en tipo 1 y 2 según las líneas l' y l'' tengan 1) cada una un punto de P o bien 2) una contiene 2 puntos de P y la otra contiene solo un punto de P. Se tiene además el siguiente

Lema 13. *Si $p_i, p_j, p_k \in P$ son los 3 puntos que definen a un corredor cerrado por un extremo de tipo 2, tales que $p_i, p_j \in l'$ y $p_k \in l''$ entonces el triángulo $\triangle p_i p_k p_j$ es acutángulo.*

Un par de semirrectas paralelas l', l'' definen un número infinito de corredores cerrados por un extremo (dependiendo de la selección de la base), pero se consideran topológicamente similares en el sentido de que todos ellos están limitados por las semirrectas l', l''. De esta manera consideraremos que topológicamente solo existen 2 corredores cerrados por un extremo limitados por l', l'', uno de ellos está acotado por arriba y el otro por abajo. De manera que un par de puntos $p_i, p_k \in P$ definen únicamente 2 corredores cerrados por un extremo de tipo 1, y una tripleta $p_i, p_j, p_k \in P$ define a lo más 2 corredores cerrados por un extremo de tipo 2. A diferencia de los corredores vacíos de ancho máximo,

[3]R. Janardan, F.P. Preparata, Widest-corridor problem, Nordic Journal of Computing 1 (1994) 231-245.

el número de corredores cerrados de tipo 1 (respectivamente de tipo 2) para el conjunto P es $O(n^2)$ (respectivamente $O(n^3)$).

Un *corredor de una esquina* es la unión de 2 corredores cerrados por un extremo (ver la figura 2.17. Un corredor de una esquina es *localmente de ancho máximo* si cada una de sus piernas contiene al menos un punto de P y su ancho es máximo entre todos los corredores de una esquina definidos por dichos 4 puntos.

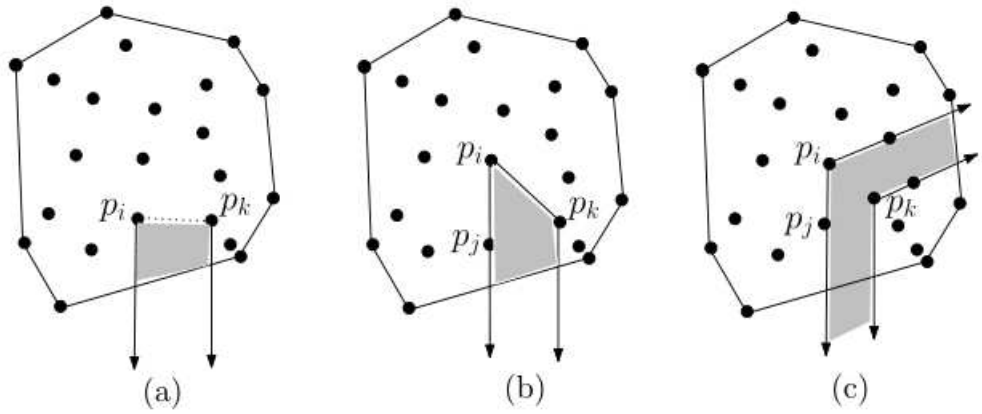


Figura 2.17: Ejemplos de corredores cerrados por un extremo a) de tipo 1 b) de tipo 2 y c) un corredor de una esquina. *[Das2009].*

El algoritmo explorará todas las parejas (p_i, p_j) que pueden determinar un corredor cerrado por un extremo de tipo 1 y todas las tripletas (p_i, p_j, p_K) que pueden determinar un corredor cerrado por un extremo de tipo 2. Para ello utilizara un arreglo D de tama o n. La i-esima entrada D_i contiene los puntos $D\backslash\{p_i\}$ ordenados en sentido contrario al movimiento de las manecillas del reloj. En el primer capítulo se sintetizaron los resultados que muestran que esto es posible en tiempo y espacio $O(n^2)$.

Para cada para de puntos (p_i, p_k) puede dar origen a lo más a 2 corredores cerrados por un extremo de tipo 1. El algoritmo determinará si estos dos corredores son vacios trazando el segmento $s = [p_i, p_k]$. A continuación se trazan dos semirrectas l' y l'' que pasan respectivamente por p_i y p_k y que son perpendiculares a s. Estas determinan un corredor cerrado por un extremo R. Dos de tales corredores son posibles, uno abajo y otro arriba de s. Mediante el uso de una estructura para conteo de rangos triangulares (que utiliza espacio $O(n^2)$) es posible determinar si dichos corredores son vacíos en tiempo $O(\log n)$. Denotaremos C^A_{ik} y C^B_{ik} al corredor determinado por p_i y p_k y acotado por s respectivamente por arriba y por abajo.

Los corredores de tipo 1 con p_i en una de sus piernas serán almacenados en el arreglo D_i a adiendo $n-1$ enteros extra. El entero $D_i[k]$ será 0 si tanto C^A_{ik} como C^B_{ik} no existen, 1 si solo el primero existe, 2 si solo el segundo existe y 3

si ambos corredores cerrados por un extremo existen. Así el tiempo y espacio requerido para generar todos los corredores cerrados por un extremo de tipo 1 es $O(n^2 \log n)$ y $O(n^2)$ respectivamente.

Una tripleta de puntos p_i, p_j, p_k puede originar a lo más 2 corredores cerrados por un extremo de tipo 2. Sean l' y l'' dos rectas paralelas tales que $p_i, p_j \in l'$ y $p_k \in l''$. Asumamos, sin perder generalidad, que ambas rectas son verticales y p_i está arriba de p_j. Dependiendo de si p_k está a la derecha o izquierda de l' y de si el segmento $[p_i.p_k]$ acota al corredor por arriba o por abajo, pueden existir cuatro posibles corredores con los 3 puntos en las semirrectas que lo forman, a los que denotaremos $C^{AL}_{(ij)k}$,$C^{AR}_{(ij)k}$,$C^{BL}_{(ij)k}$,$C^{BR}_{(ij)k}$. Los superíndices A y B denotan si el segmento acota el corredor por arriba o por abajo (above y below en inglés) y los superíndices L,R denotan si p_k está a la izquierda o derecha de l' (left o right en inglés).

Para cada punto p_i el algoritmo considerará todas las líneas que pasan por p_i y otro punto $p_j \in P \backslash \{p_i\}$. Para cada una de estas líneas l' el algoritmo identificará todos los puntos p_k que pueden originar la otra línea l'' de un corredor cerrado por un extremo de tipo 2 vacío. El algoritmo crea n arreglos C_i, cada uno de los cuales almacena los corredores de tipo 2 cuya pierna l' pasa por p_i. Cada entrada de dicho arreglo corresponde al punto p_j que determina l' junto a p_i. Dicha entrada consiste de 4 arreglos C^{AR}_{ij},C^{AL}_{ij},C^{BR}_{ij},C^{BL}_{ij}. Cada uno de dichos arreglos contiene a todos los puntos p_k que pueden determinar la otra recta l'' que limita el corredor de tipo 2.

Los autores describen el procedimiento para generar el conjunto de corredores C^{AR}_{ij}. El procedimiento para los conjuntos C^{AL}_{ij}, C^{BL}_{ij} y C^{BR}_{ij} es semejante.

Sea P_R el conjunto de puntos a la derecha de la línea $l' = (p_i, p_j)$. Sean l^r_i y l^r_j las semirrectas perpendiculares a l' que pasan respectivamente por p_i y p_j y hacia la derecha de l'. Si la semirrecta l'' de un corredor en C^{AR}_{ij} está determinada por un punto p_k entonces 1) p_k está en el corredor cerrado por un extremo R_{ij} determinado por l^r_i,l^r_j y $[p_i, p_j]$ y 2) el corredor cerrado por un extremo determinado por las rectas l', l'' y el segmento $[p_i, p_k]$ es vacío.

Sea S la lista de puntos en P dentro de la franja R_{ij}, ordenados según sus distancias a l'. A partir de S obtenemos una segunda lista $\hat{S}$ con el mayor número posible de puntos en el mismo orden que S y tal que para cada para de puntos $p_k, p_{k'} \in \hat{S}$ si p_k es más cercano a l' que $p_{k'}$ entonces $p_{k'}$ es más cercano a l^r_j que p_k. Es decir, los puntos de $\hat{S}$ forman una escalera en R_{ij}. Vea la figura 2.18. Para cada $p_k \in \hat{S}$ el algoritmo determina si p_i, p_j, p_k determinan un corredor en C^{AR}_{ij} 1) trazando la línea l'' que pasa por k paralela a l' y 2) determinando si el corredor cerrado por un extremo acotado por l', l'' y el segmento $[p_i, p_k]$ es vacío.

El número de puntos en R_{ij} es $O(n)$, lo mismo que $\hat{S}$. Para cada punto en se necesita ejecutar una consulta de rangos triangular para determinar si dicho punto pertenece a C^{AR}_{ij}. De manera que el tiempo para construir es $O(n \log n)$. Puesto que cada pareja (p_i, p_j) determina uno de tales conjuntos se tiene:

Lema 14. *La complejidad temporal para generar todos los corredores cerrados por un extremo de tipo 2 es* $O(n^3 \log n)$.

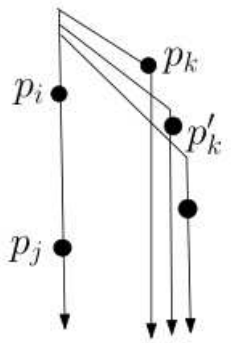


Figura 2.18: Los puntos $\hat{S}$ forman una escalera en R_{ij}. *[Das2009].*

El espacio utilizado en el algoritmo descrito es $O(n^3)$. Considerando cada punto p_i por separado y reutilizando C_i es posible reducir dicho espacio a $O(n^2)$.

Para generar los corredores de una esquina a partir de los corredores de tipo 2 el algoritmo itera sobre todos los puntos p_i. Una recta que pasa por dicho punto es girada en sentido contrario a las manecillas del reloj a partir de la posición vertical. Al incidir dicha línea sobre un punto p_j se determina la línea $l' = (p_i, p_j)$. Sean P_L y P_R los puntos a la derecha e izquierda de l'. Describiremos como computar el corredor de una esquina de mayor ancho utilizando los corredores cerrados por un extremo C_{ij}^{AR}. Para ello se computarán los eslabones vacíos en P_R que pueden unirse con corredores de tipo 2 en C_{ij}^{AR}. El algoritmo necesita mantener los arreglos A_R y A_L de los duales respectivos de P_R y P_L. Los cuatro conjuntos son actualizados al continuar las iteraciones en p_j.

Considere un corredor de tipo 2, $C_{(ij)k}^{AR} \in C_{ij}^{AR}$. El algoritmo hallará un eslabón entre los puntos en P_R que pueda ser unido a $C_{(ij)k}^{AR}$ para obtener un corredor de una esquina de ancho máximo. Observe que cada vértice de A_R corresponde a un corredor vacío entre los puntos de P. El ancho de todos los corredores correspondientes a vértices de A_R puede ser computado en tiempo $O(n^2 \log n)$. Sin embargo no todos estos corredores pueden ser unidos a $C_{(ij)k}^{AR}$ para obtener un corredor de una esquina vacío. Para que esto sea posible necesitamos el cierre convexo H_{ijk} de todos los puntos sobre el segmento $[p_i, p_k]$ dentro de la franja definida por l' y l''. Si un corredor en P_R pasa por abajo de H_{ijk} entonces puede ser unido a $C_{(ij)k}^{AR}$ para formar un corredor de una esquina vacío. Vea la figura 2.19a. Denominaremos *válidos* a dichos corredores.

Para obtener dichos corredores, considere el dual de la cadena inferior de H_{ijk}. Se trata de una cadena monótona $H_{ijk}^{L} = \{h_1, \ldots, h_k\}$. $h_i, i = 1, \ldots, k$ son las aristas de dicha cadena. El corredor vacío correspondiente a un vértice de A_R es un corredor válido si se encuentra arriba de H_{ijk}^{L} (Vea la figura 2.19b. Entre dichos corredores el algoritmo identificará al de ancho máximo. Para ello asociaremos a cada línea $\alpha \in A_R$ un árbol binario balanceado. Las hojas del árbol corresponden a los vértices de A_R que se encuentran sobre α y, por tanto,

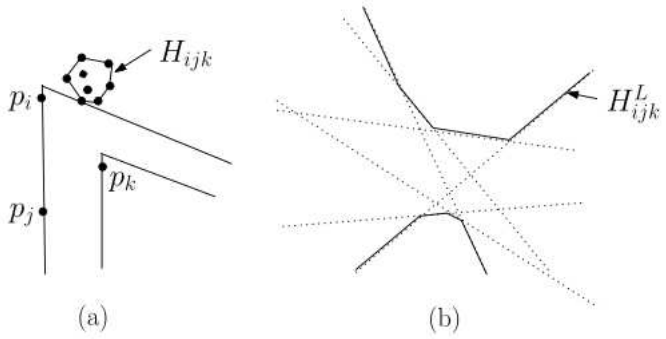


Figura 2.19: Un corredor válido en P_R pasa bajo H_{ijk}. *[Das2009].*

corresponden a corredores en P_R. En cada nodo interno del árbol se almacena el ancho máximo entre los corredores en el subárbol correspondiente. Este árbol puede obtenerse en tiempo $O(n)$ para cada $\alpha \in A_R$. Para cada α los vértices válidos sobre dicha línea se pueden obtener determinando las intersecciones de α con la cadena H^L_{ijk}, los cuales pueden ser 2,1 o ninguno. Dichos puntos de intersección pueden obtenerse mediante busqueda binaria. Una vez conocidos dichos puntos de intersección se puede obtener el ancho del vértice con ancho máximo en el segmento determinado por ellos recorriendo el árbol asociado desde las hojas hacia la raíz, lo que requiere tiempo $O(\log n)$. De manera que el corredor válido de ancho máximo para $C^{AR}_{(ij)k}$ puede calcularse en tiempo $O(n \log n)$.

Sean entonces p_i y p_j dados, los cuales determinan l', la cual asumiremos vertical, sin perdida de generalidad. Los puntos que pueden definir a la otra pierna l'' Se encuentran ordenados en $\hat{S}$. El punto más alto en define el corredor de tipo 2 más angosto, y, conforme la altura de los puntos en disminuye, el ancho del corredor de tipo 2 aumenta. Así que se realiza una busqueda binaria en : se escoge el punto medio $p_k \in$. Este define un corredor de tipo 2, $C^{AR}_{(ij)k}$. A continuación el algoritmo computa el dual H^L_{ijk} de la cadena inferior del cierre convexo H_{ijk} y se computa el corredor válido más ancho C entre los puntos de P_R. Si el ancho de C empata con $C^{AR}_{(ij)k}$ entonces se actualiza el ancho del mejor corredor de una esquino visto hasta el momento. En otro caso se continua la busqueda binaria en hacia arriba o abajo, dependiendo si el ancho de C es mayor o menor que el de $C^{AR}_{(ij)k}$. La busqueda continúa hasta que se visitan dos elementos consecutivos de , entre los cuales se encuentra el corredor de una esquina de mayor ancho para p_i y p_j. Esto lleva al:

Lema 15. *El corredor de una esquina de ancho máximo para p_i y p_j dados puede computarse en tiempo $O(n \log^2 n)$.*

Conforme las iteraciones en p_j continuan es necesario actualizar las P_R,P_L,A_R y A_L. Esto puede realizarse en tiempo $O(n \log n)$[4]. La actualización de las es-

[4]S.C. Nandy, T. Harayama, T. Asano. *Dynamically maintaining the widest k-dense corri-*

tructuras asociadas a los duales de las líneas requiere el mismo tiempo. Una vez actualizadas estas estructuras, el tiempo que lleva computar el corredor de una esquina de ancho máximo es el mismo, $O(n \log^2 n)$. De manera que el proceso de todos los elementos de D_i toma tiempo $O(n^2 \log^2 n)$, y debe repetirse para todos los p_i, lo que lleva al

Teorema 12. *Las complejidades temporal y espacial del problema del corredor de una esquina usando un corredor cerrado por un extremo de tipo 2 son, respectivamente,* $O(n^3 \log^2 n)$ *y* $O(n^2)$.

Para obtener los corredores de una esquina más anchos utilizando los corredores cerrados por un extremo de tipo 1 se utiliza una técnica similar. Sea $p_i \in P$. Sea L una recta vertical que pasa por p_i, y L' una semirecta perpendicular a L que pasa por p_i hacia la derecha de L. El algoritmo computa el arreglo A_R de puntos a la derecha de L. Las lineas L y L' son rotadas en sentido contrario al de las manecillas del reloj hasta que incidan en algún punto de P. Sea p_k dicho punto:

- Si L incidió en p_k se actualiza A_R insertando o eliminando la linea correspondiente, lo cual requiere tiempo $O(n \log n)$.
- Si L' incidió en p_k entonces buscamos un corredor válido de tipo 1, C_{ij}^A (respectivamente C_{ij}^B) en D_i. En caso de que exista se computa el cierre convexo H_{ik} de los puntos arriba del segmento $[p_i, p_k]$ y dentro de la franja determinada por las lineas L y L'', donde L'' es paralela a L' y pasa por p_k. A continuación el algoritmo computa el corredor válido más ancho respecto al dual de la cadena inferior H_{ik}^L. Como se describió en el anterior caso, esto requiere tiempo $O(n \log n)$.

La rotación de L y L' continua sobre todos los puntos de $P \backslash \{p_i\}$. De tal forma que el proceso de cada p_i toma tiempo $O(n^2 \log n)$. Lo cual da como resultado el

Lema 16. *Las complejidades temporal y espacial del problema del corredor vacío de una esquina utilizando corredores cerrados por un extremo de tipo 1 son, respectivamente,* $O(n^3 \log n)$ *y* $O(n^2)$.

Y, junto con el resultado para corredores de tipo 2, lleva al

Teorema 13. *El problema del corredor vacío de una esquina puede ser resuelto en tiempo* $O(n^3 \log^2 n)$ *y espacio* $O(n^2)$.

2.3. Corredores k-densos: Algoritmos de Janardan y Preparata

Sea S un conjunto de n puntos en el plano. Un *corredor* a través de S es la región abierta del plano acotada por dos rectas paralelas que intersecan el

dor. Theoretical Computer Science 255 (2001) 627 639.

cierre convexo $CH(S)$ de S. El *ancho* del corredor es la distancia entre las lineas que limitan el corredor. El corredor es *k-denso* si contiene k puntos de S, $0 \leq k \leq n-2$. El problema del corredor k-denso consiste en encontrar el corredor de ancho máximo.

A lo largo del artículo se asume que no hay 3 puntos de S que sean colineales y que no hay 4 puntos del conjunto que formen un trapezoide. Además se asume que no existe en S un corredor k-denso vertical, pues tal corredor puede encontrarse en tiempo $O(n \log n)$ ordenando los puntos según su coordenada x.

Se tiene el siguiente:

Teorema 14. *Sea C^* el corredor más ancho a través de S limitado por las líneas l' y l''. Entonces ocurre alguna de las siguientes condiciones:*

(a) Una de las líneas, p.ej. l' pasa por dos puntos $p_i, p_j \in S$ y l'' pasa por otro punto $p_h \in S$.

(b) Existen dos puntos $p_i, p_j \in S$ tales que l' pasa por p_i, l'' pasa por p_j, y l' y l'' son perpendiculares a la linea que pasa por p_i y p_j.

Denominaremos a un corredor de tipo (a) o (b) según cumpla la primera o segunda condición del teorema.

Sea F la transformación dual. Sea $H = \{l_i = F(p_i) | p_i \in S\}$ el conjunto de lineas que son los duales de los puntos de S, y A el arreglo determinado por H. Sea $v_{ij} = l_i \cap l_j$ un vértice de A y $x(v_{ij})$ la coordenada x de dicho vértice.

Si C es un corredor de tipo (a) entonces $F(l') = v_{ij}$ y $F(l'')$ es el punto donde la línea vertical que pasa por v_{ij} interseca a l_h. Si C es un corredor de tipo (b) entonces $F(l')$ y $F(l'')$ son puntos en l_i y l_j respectivamente, y ambos puntos tienen coordenada x igual a $-1/x(v_{ij}$. Además si C es de tipo (a) (respectivamente de tipo (b)) y l es cualquier línea paralela a l' y l'' y que se encuentra en el interior de C, entonces $x(F(l)) = x(v_{ij})$ (respectivamente $x(F(l)) = -1/x(v_{ij})$ y $y(F(l')) < y(F(l)) < y(F(l''))$.

Entonces, en ambos casos, el dual $F(C)$ de C es el segmento vertical abierto determinado por $F(l')$ y $F(l'')$. C es un corredor k-denso sii $F(C)$ interseca k lineas de H. El ancho de C es $|y(F(l')) - y(F(l''))|/\sqrt{1 + x(F(C))}$. En este caso $x(s)$, donde s es un segmento vertical, denota la coordenada x del mismo.

El algoritmo encuentra C^* explorando todos los vértices $v_{ij} \in A$ y encontrando la $(k+1)$-esima línea arriba del dicho vértice. Si existe dicha línea, se computa el ancho del correspondiente corredor de tipo (a). Un proceso similar se realiza para la $(k+1)$-ésima línea abajo del vértice. Además si en la coordenada $x = -1/x(v_{ij}$ hay k líneas entre l_i y $l - j$ entonces se computa el ancho del corredor de tipo (b) correspondiente. Al finalizar el recorrido se da como salida el ancho del corredor más ancho encontrado.

El algoritmo realiza un barrido en las lineas duales y no mantiene el arreglo completo A en memoria. Se utilizan dos líneas de barrido: una línea principal V que encuentra los corredores de tipo (a), y una línea esclava V' que encuentra los corredores de tipo (b). V se mueve desde $x = -\infty$, pasa por $x = 0$ y de allí a $x = \infty$, mientras que V' se mueve de $x = 0$ a $x = \infty$ y después de $x = -\infty$ a $x = 0$.

Para el barrido principal, V tiene asociado un arreglo D y una cola de prioridad Q implementada mediante un min-heap. D almacena las líneas duales en el orden que intersecan a V de abajo a arriba. Q almacena a v_{fg} con llave $x(v_{fg})$ si l_f y l_g son adyacentes en D y su intersección está a la derecha de V. l_f y l_g tienen, en D, apuntadores a v_{fg} en Q, y v_{fg} tiene, en Q, apuntadores a l_f y l_g en D.

El barrido principal es inicializado insertando en D todas las líneas en orden ascendente de sus coordenadas y con la línea $x = -\infty$.

En cada paso del algoritmo el siguiente evento v_{ij} se obtiene consultando el elemento menor de Q. Suponga que, justo antes de v_{ij}, l_i está arriba de l_j, l_a es la línea inmediatamente arriba de l_i y l_b la línea inmediatamente abajo de l_j. Para procesar v_{ij} se realizan las siguientes acciones:

1. l_i y l_j intercambian posiciones en D.
2. Si v_{ia} está a la derecha de V entonces se elimina dicho vértice de Q. v_{jb} es procesado de la misma manera.
3. Si v_{ib} está a la derecha de V entonces se añade dicho vértice a Q. v_{ja} es procesado de igual manera.
4. Si $D[t] = l_j$ entonces la $(k+1)$-ésima línea arriba de v_{ij} es $D[t+k+1]$ (si $t+k+1 \leq n$), y la $(k+1)$-ésima línea abajo de v_{ij} es $D[t-k-2]$ (si $t-k-2 \geq 1$). En caso de existir, se computan los anchos de los correspondientes corredores.

En el barrido de la línea esclava se utilizan dos estructuras, D' y Q', análogas a las usadas en el barrido principal. Cada línea en D' tiene un apuntador a la línea correspondiente en D y viceversa. La linea esclava se moverá en una primera fase desde $x = 0$ hasta $x = -\infty$ (correspondiente al movimiento de V desde $x = -\infty$ hasta $x = 0$ en el barrido principal). En cada parada de este barrido se realizan pasos análogos a 1-3 del barrido principal.

A continuación suponga que V está procesando el evento v_{ij}. Entonces V' avanza al evento u inmediatamente a v_{ij}, esto es, u es el elemento más a la derecha tal que $x(u) \leq -1/x(v_{ij})$, y se realizan los pasos 1-3 en todos los eventos, incluyendo a u (esto se debe a que el estado de D' y Q' justo después de procesar u es el mismo que en $x = -1/x(v_{ij})$). A continuación si $l_i = D'[b]$ y $l_j = D'[c]$ y $|b-c|-1 = k$ entonces hay k líneas entre l_i y l_j en la coordenada $x = -1/x(v_{ij})$ y entonces debe calcularse el ancho del corredor de tipo (b) correspondiente.

Cuando V alcanza $x = 0$ se debe reinicializar V' en $x = -\infty$ y ambos barridos continuan.

El algoritmo utiliza espacio $O(n)$, y el tiempo de ejecución está dominado por el tiempo $O(\log n)$ requerido para actualizar las colas de prioridad en los $O(n^2)$ vértices a procesar, lo que lleva al:

Teorema 15. *El corredor k-denso más ancho ($0 \leq k \leq n-2$) a través de n puntos puede encontrarse en tiempo $O(n^2 \log n)$ usando espacio $O(n)$.*

Capítulo 3

Antecedentes cromáticos

En esta sección haremos un breve repaso sobre diversos resultados que involucran objetos cromáticos mínimos: corredor cromático orientado y no orientado, rectángulo cromático orientado y no orientado y l-corredor. Como se indicó en la introducción en estos casos se busca minimizar el tamaño de los corredores y rectángulos, además de que estos deben contener al menos un punto de cada color.

3.1. Corredor cromático con orientación fija

Dado un conjunto S de puntos coloreados. Hay n puntos y m colores. El problema consiste en encontrar el corredor cromático orientado vérticalmente de ancho mínimo. El problema es resuelto en [Das2009]. Es descrito en dicho artículo como parte del algoritmo para encontrar el corredor cromático de orientación arbitraria.

Un corredor C se define como la región acotada por dós lineas paralelas l y l' que intersecan el cierre convexo de S. El ancho del corredor es la distancia entre l y l'. Dicho corredor es *cromático* si contiene puntos de todos los colores. Los puntos se encuentran en *posición general*, definida para este problema como: a) no exiten 3 puntos colineales en S y b) las pendientes de cualesquiera 2 rectas determinadas cada una por 2 puntos del conjunto son distintas.

El primer paso del algoritmo es proyectar los puntos en el eje x y ordenarlos. El algoritmo utiliza dos apuntadores p_1 y p_2 para marcar las fronteras del corredor candidato. Se auxilia de un arreglo c de tamaño m, el cual indica cuantos puntos de cada color m hay en el corredor determinado por p_1 y p_2. Además otra variable auxiliar z indica el número de ceros en el arreglo m, y otra variable opt indica el ancho del corredor mínimo encontrado hasta el momento, inicialmente $opt = \infty$.

1. Inicialmente ambos apuntadores se encuentran en el punto con menor coordenada x. Se suma 1 a la entrada correspondiente al color de dicho punto en c.

2. p_2 se mueve hacia la derecha por los puntos de S según su coordenada x. En cada punto:

   a) Se aumenta en 1 la correspondiente entrada en c. Si dicha entrada paso de 0 a 1, z se decrementa en 1. Si el valor de z pasó de 1 a 0 se ha hallado un candidato:

      1) El ancho del corredor candidato es $x(p_2) - x(p_1)$. Si este valor es menor que *opt*, actualizar dicho valor.
      2) Mover p_1 al siguiente punto a la derecha
      3) Decrementar en 1 el valor de la entrada de c correspondiente al color del anterior punto p_1. Aumentar z si es necesario.
      4) Mientras no ocurra un cambio en z en el paso anterior, repetir los pasos 1-4 anteriores.

El algoritmo utiliza tiempo $O(n)$ una vez que los puntos han sido ordenados según su coordenada x, por lo que el tiempo de ejecución total es $O(n \log n)$.

3.2. Corredor cromático de orientación arbitraria

En este problema la orientación del corredor ya no está restringida a ser vertical.

3.2.1. Algoritmo de Abellanas et al.

En la sección final de [Abellanas2001] se muestra la primera solución a este problema.

La primera observación es que la solución debe tener 3 puntos de 3 colores distintos en la frontera, pues si solo hay 2 puntos en la frontera o un color se repite entre los 3 puntos, entonces es posible disminuir el ancho del corredor mediante rotaciones.

A continuación se esboza una solución por fuerza bruta: considere las $O(n^2)$ lineas determinadas por cualquiera 2 puntos de S, y ordénelas según su pendiente (lo cual toma $O(n^2 \log n)$). Seleccione una cualquiera de ellas y proyecte todos los puntos en una perpendicular a dicha linea. Ordenando los puntos en dicha perpendicular es posible hallar una solución como en el problema anterior mediante un recorrido lineal, para un tiempo de ejecución total de $O(n \log n)$. Ahora, al cambiar cada vez de dirección, el orden de los puntos proyectados solo se modifica en 2 puntos, de manera que la actualización del ordenamiento toma $O(1)$, más $O(n)$ para el recorrido lineal que encuentra la solución en dicha dirección. De forma que el tiempo de ejecución total es $O(n^3)$.

Una mejor solución es posible utilizando inversiones y envolventes exteriores. Para ello considere un punto b de un color cualquiera. Para otro punto r de otro color, dos lineas paralelas que pasan respectivamente por b y por r tienen la siguiente propiedad: la proyección ortogonal de b sobre la linea que pasa por r (y por tanto un punto tal que el segmento entre dicho punto y b es el ancho del

corredor determinado por las lineas que pasan por b y r) describe un círculo que tiene al segmento br por diámetro.

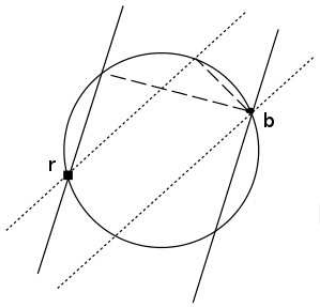


Figura 3.1: Los puntos b y r, dos lineas paralelas por ambos puntos y la proyección ortogonal de b en la linea que pasa por r (linea segmentada).

Ahora mantenemos b fijo y consideramos los círculos determinados por todos los puntos r de un color determinado (y distinto del color de b). Esto determina un arreglo de círculos que pasan por b. La envolvente inferior de este arreglo de circulos indica, para cualquier dirección, el ancho mínimo del corredor que tiene a b en un extremo e incluye únicamente a un punto del otro color. Si ahora invertimos todos los circulos respecto al punto b obtenemos un arreglo de lineas. Las distancias respecto a b se invierten también, de forma que ahora la envolvente exterior del arreglo de lineas nos da el ancho del corredor descrito anteriormente. Puesto que 2 lineas cualquiera se intersecan únicamente una vez, la complejidad de la envolvente es $O(n_i)$ y puede ser computada en $O(n_i \log n_i)$, p.ej. por un algoritmo "divide y venceras", ya que el paso *merge* toma tiempo lineal puesto que sabemos que el punto b se encuentra en el kernel de los poligonos a mezclar).

Para cada sitio, las envolventes exteriores de todos los colores pueden calcularse en $O(n \log n)$. La envolvente inferior de estas envolventes exteriores de cada color determina un corredor cromático. Dicha envolvente inferior tiene complejidad $O(n\alpha(k))$ y puede computarse en $O(n\alpha(k) \log k)$[1]. Dicha envolvente inferior debe computarse y compararse para cada punto. Con ello se obtiene el

Teorema 16. *Dado un conjunto de puntos coloreados, un corredor cromático no orientado de ancho mínimo puede computarse en tiempo* $O(n^2\alpha(k) \log k)$.

3.2.2. Mejoras de Das, Goswami y Nandy

El algoritmo para el corredor cromático de orientación arbitraria descrito en [Das2009] encuentra todos los corredores candidatos en el espacio dual. Para ello se analiza primero el caso en que el corredor cromático mínimo tiene orientación

[1]Usando resultados de: Micha Sharir, P. K. Agarwal *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, N.Y. 1995.

vertical utilizando algoritmo para el corredor cromático orientado. Así en lo subsecuente puede asumirse que el corredor de ancho mínimo no es vertical.

Sean l y l' las rectas que limitan a un corredor cromático C. En el espacio dual, C corresponde a un *segmento cromático*, el cual es un segmento vertical (sobre la recta vertical en el espacio dual cuya coordenada x es la pendiente de l y l') limitado por l^* y l'^*.

Sabemos que una de las rectas l o l' que limitan al corredor contiene 2 puntos de distinto color. El algoritmo barre con una recta vertical las intersecciones de los duales, p^* y p'^*, de cualesquiera dos puntos p y p'. En cada una de estas intersecciones se encuentra el segmento cromático arriba y abajo de la intersección de p^* y p'^*. Para ello se mantiene el estado de la linea de barrido en un arreglo B de tamaño n. Cada entrada en B representa la intersección del dual de un punto con la linea de barrido. Cada entrada en el el arreglo B tiene la siguiente información extra:

prev apuntador a la siguiente linea del mismo color abajo en el arreglo B.

next apuntador a la siguiente linea del mismo color arriba en el arreglo B.

cs apuntador a la primera linea arriba en el arreglo B tal que el segmento entre esta linea y el segmento en la posición cs es cromático.

En la primera parada del barrido se inicializan los anteriores apuntadores para cada elemento de B, lo cual puede lograrse en tiempo $O(n)$ usando la misma técnica que en el corredor orientado. Posteriormente en cada parada del barrido la actualización de estado B puede realizarse en tiempo constante. Los detalles son sencillos, el lector interesado puede encontrarlos en el artículo original.

De esta manera se obtiene una mejora respecto al resultado basado en envolventes e inversiones:

Teorema 17. *Existe un algoritmo de tiempo $O(n^2 \log n)$ para problema del corredor cromático no orientado.*

3.3. Rectángulo cromático

El problema consiste en, dado un conjunto de puntos S, donde cada punto $p \in S$ tiene un color p_{col}, encontrar el rectángulo cuyos lados son paralelos a los ejes coordenados y de menor area o perímetro que contiene a puntos de todos los colores.

3.3.1. Algoritmo de Abellanas et al.

En [Abellanas2001] se dan 3 algoritmos para resolver el problema del rectángulo cromático.

Sea $p \in S$. p_x denotará la coordenada x del punto, p_y la coordenada y. Para simplificar la exposición se asume que 2 puntos cualquiera de S no determinan una recta horizontal o vertical. Para hacer un primer acercamiento a una solución se define:

Definición 4. *Un rectángulo con lados paralelos a los ejes coordenados es* no encogible *sii contiene sitios de todos los k colores y no contiene propiamente a otro rectángulo de lados paralelos a los ejes que contiene puntos de todos los colores.*

Entonces un rectángulo no encogible debe tocar un punto de S con cada uno de sus lados, de manera que hay entre 2 y 4 puntos en su frontera, de los cuales cualesquiera 2 no son del mismo color. Además no hay en el interior del rectángulo otros puntos de los colores de los puntos en la frontera.

Cada uno de los algoritmos explora todos los rectángulos no encogibles y compara su area o perímetro.

Algoritmo 1

1. Para cada posible esquina inferior izquierda (determinada por 1 o 2 puntos) se exploran todos los rectángulos no encogibles.
2. Inicialmente el limite superior del rectángulo está en $y = +\infty$.
3. El borde derecho avanza hacia la derecha punto por punto desde la posición inicial en el borde izquierdo. Se detiene una vez que el rectángulo contiene todos los colores.
4. Entonces se hace descender el borde superior punto por punto mientras el rectángulo siga siendo cromático. Una vez que el borde superior se detiene, se ha hallado un rectángulo no encogible.
5. Para buscar el siguiente rectángulo no encogible, se hace avanzar el borde derecho hasta el siguiente punto del mismo color que el punto en el borde superior, y se itera desde el paso anterior.

Es claro que, para una esquina inferior izquierda dada, las iteraciones para hallar los rectángulos no encogibles consumen $O(n)$ tiempo si se tiene a los puntos ordenados respecto a las coordenadas x y respecto a las coordenadas y. Además el borde izquierdo se encuentra entre los primeros $n - k + 1$ puntos (ordenados respecto a x) y el borde inferior entre los primeros $n - k + 1$ sitios (ordenados respecto a y). De manera que el algoritmo toma tiempo $O(n(n-k)^2$ en total.

A continuación el artículo presenta el siguiente

Lema 17. *Existen $O((n-k)^2)$ rectángulos no encogibles.*

El lector interesado puede consultar la prueba del mismo en el artículo original.

Algoritmo 2

El siguiente algoritmo en [Abellanas2001] mejora el tiempo de ejecución a $O(nk(n-k))$. Para cada par de puntos a y b con $a_y < b_y$ el algoritmo explora todos los rectángulos no encogibles con a en la frontera inferior y b en la

superior. Sean l y r los puntos en los bordes izquierdo y derecho. Para que el rectángulo determinado por a, b, l y r sea no encogible, debe tener las siguientes características:

1. a y b deben estar en el rectángulo.
2. El interior del rectángulo no debe contener sitios de los colores a_{col} y b_{col}.
3. Los colores de l y r no deben aparecer en el interior del rectángulo.

Para definir formalmente estas características se define, para cada color c

$$L_c(a,b) = \max_{p \in S, p_{col}=c} \{p_x | a_y < p_y < b_y \text{ y } p_x < a_x\}$$
$$R_c(a,b) = \min_{p \in S, p_{col}=c} \{p_x | a_y < p_y < b_y \text{ y } p_x > a_x\}$$

Es decir, $L_c(a,b)$ es la máxima coordenada x de los puntos de color c en la franja horizontal determinada por a y b y a la izquierda de a_x. $R_c(a,b)$ es el mínimo análogo a la derecha de a_x. El valor de $L_c(a,b)$ (respectivamente $R_c(a,b)$) es $-\infty$ (respectivamente $+\infty$) si no hay puntos en la zona respectiva.

Con estás definiciones las 3 condiciones anteriores pueden enunciarse como:

1.
$$l_x \leq \text{mín}(a_x, b_x)$$
$$r_x \geq \text{máx}(a_x, b_x)$$

2.
$$l_x > \text{máx}(L_{a_{col}}(a,b), L_{b_{col}}(a,b))$$
$$r_x < \text{mín}(R_{a_{col}}(a,b), R_{a_{col}}(a,b))$$

3.
$$l_x = L_{l_{col}}(a,b) \text{ si } l \neq a, b$$
$$r_x = R_{r_{col}}(a,b) \text{ si } r \neq a, b$$

De manera que las condiciones (1) y (2) brindan intervalos para la coordenada x de los bordes izquierdo y derecho. Además la condición 3 proporciona el siguiente

Lema 18. *Sean $a, b \in S$. Existen a lo más $k-2$ rectángulos no encogibles cuyos bordes inferior y superior pasan, respectivamente, por a y b.*

Para a fijo es posible obtener facilmente los valores $L_c(a,b)$ y $R_c(a,b)$ si b itera sobre los puntos en orden según la coordenada y. Esto es realizado en el algoritmo 2. Nuevamente referimos al lector interesado al artículo original. Con base a ello se formula el

Lema 19. *Los candidatos explorados por el algoritmo 2 son todos los rectángulos no encogibles. El tiempo de ejecución del algoritmo es $O(nk(n-k))$.*

Algoritmo 3

El tercer algoritmo propuesto se basa en una definición de elementos maximales.

Definición 5. *Un* elemento maximal *de un conjunto T de puntos en el plano es un punto $p \in T$ tal que no existe $q \in T$ con $p_x > q_x$ y $p_y < q_y$.*

Bajo esta definición los elementos maximales lo son *hacia arriba y a la izquierda.*

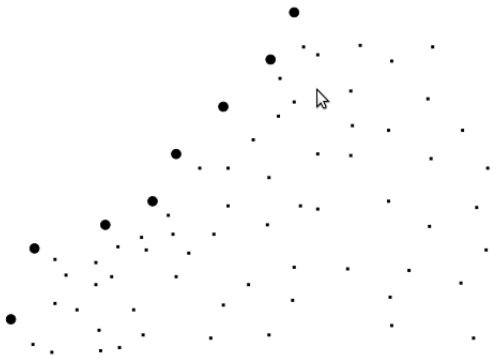


Figura 3.2: Elementos maximales en un conjunto de puntos.

Dados los puntos a y b como en el algoritmo anterior, considere los puntos $L_c(a,b)$ y $R_c(a,b)$ para todos los colores c. Transformaremos estos valores en puntos en el plano:

$$T_c(a,b) = (L_c(a,b), R_c(a,b)) \text{ para } c \neq a_{col}, b_{col}$$

$T(a,b)$ denotará al conjunto de todos los puntos $T_c(a,b)$. El siguiente paso es relacionar los elementos maximales en $T(a,b)$ con un rectángulo cromático, a través del

Lema 20. *Sean a, b tales que la franja horizontal entre ambos es cromática. $T_c(a,b)$, para algún color c, es un elemento maximal de $T(a,b)$ sii el rectángulo con $a, b, L_c(a,b)$ y $R_c(a,b)$ en los bordes, respectivamente, inferior, superior, izquierdo y derecho contiene puntos de todos los colores, exceptuando posiblemente a b_{col}.*

Esta relación entre elementos maximales y rectángulos cromáticos es refinada en los siguientes dos lemas

Lema 21. *Considere un rectángulo no encogible con puntos a, b, l y r en sus bordes, respectivamente, inferior, superior, izquierdo y derecho, y tales que $a \neq l, r$ y $b \neq l, r$. Entonces $T_{l_{col}}(a,b)$ y $T_{r_{col}}(a,b)$ son dos elementos maximales sucesivos en $T(a,b)$.*

Lema 22. *Sea $a, b \in S$ y dos colores $c, c' \neq a_{col}, b_{col}$, y tales que $T_c(a,b)$ y $T_{c'}(a,b)$ son dos elementos maximales sucesivos de $T(a,b)$, y la franja horizontal entre a y b contiene puntos de todos los colores. Entonces el rectangulo con bordes en a, b, l, r con $l_x = L_{c'}(a,b)$ y $r_x = R_c(a,b)$ es no encogible si adicionalmente se satisfacen las condiciones (1) y (2).*

Con la ayuda de estos lemas es posible modificar el algoritmo 2 mediante el uso de un árbol dinámico, utilizando un método descrito por Overmars y van Leeuwen[2]. Con ello se prueba el

Lema 23. *Dados n sitios y k colores el problema del rectángulo cromático orientado puede resolverse en tiempo $O(n(n-k)\log^2 k)$.*

3.4. Rectángulo cromático no orientado

Dado un conjunto de puntos S de m colores, el problema consiste en encontrar el rectángulo generador de area mínima. A diferencia del problema anterior, los lados del rectángulo no tienen que ser paralelos a los ejes coordenados. En [Das2009] se dan dos soluciones al problema.

Un rectángulo cromático es *minimal* si no existe otro rectángulo cromático de menor area que contenga el mismo conjunto de puntos.

Dados 4 puntos coloreados $p_a, p_b, p_c, p_d \in S$, estos pueden generar o bien ningún rectángulo cromático minimal, o un conjunto infinito de rectángulos minimales. En este último caso existe uno rectángulo cromático de menor area, el cual será denominado *primal* (PCSR por sus siglas en inglés). Los rectángulos cromáticos primales satisfacen:

Teorema 18. *Sea R un rectángulo cromático primal. Entonces uno de sus lados incide en 2 puntos de S de distintos colores, y no existen puntos de estos 2 colores en ningún otro lugar en el interior o la frontera del rectángulo.*

El algoritmo explora todos los pares de puntos p_a, p_b de colores, respectivamente, α, β, con $\alpha \neq \beta$. Para cada uno de estos pares de puntos se consideran todos los posibles rectángulos cuyo lado inferior está contenido en la linea l_{ab} que une ambos puntos y cuyo lado superior está determinado por otro punto p_c de color $\gamma \neq \alpha, \beta$. Para ello se realizan dos tipos de barrido. El primero itera sobre los vértices bicoloreados del arreglo de duales de los puntos del conjunto, $A(S*)$. Esto determina los puntos p_a y p_b. Para cada evento en este primer barrido se realiza un segundo tipo de barrido, este en el plano primal, con una linea paralela a l_{ab}. Este segundo barrido recorre en orden los puntos hacia arriba primero y luego hacia abajo de l_{ab}.

Una vez determinada l_{ab}, rote plano de forma que l_{ab} sea paralela al eje x, y trace 2 perpendiculares en p_a y p_b. Esto divide el plano en 3 regiones que serán denominadas $LEFT, MID$ y $RIGHT$. El segundo barrido pasará por todos los puntos arriba de l_{ab}.

[2]M.H. Overmars y J. van Leeuwen. *Maintenance of configurations in the plane.* J. Comput. Syst. Sci., 23:166-204, 1981.

A continuación los autores hacen 4 observaciones, las cuales indican algunas condiciones en los que un punto hallado en el segundo barrido p_c de color γ no genera ningún rectángulo cromático primal:

1. $\gamma = \alpha$ o $\gamma = \beta$.
2. Existe un punto p de color γ en MID con $y(p) < y(p_c)$.
3. $p_c \in LEFT$ y existe otro punto $p \in LEFT$ de color γ tal que $x(p) > x(p_c)$ y $y(p) < y(p_c)$.
4. $p_c \in RIGHT$ y existe otro punto $p \in RIGHT$ de color γ tal que $x(p) < x(p_c)$ y $y(p) < y(p_c)$.

La primera observación nos permite ignorar puntos cuyo color ya apararece en la base del rectángulo. En particular si dicho punto p_c está en MID entonces ya no será posible que ningún punto posterior en el segundo barrido genere un rectángulo primal, por lo que pueden ignorarse todos los puntos subsecuentes en este segundo barrido. Si el punto p_c aparece en $LEFT$ o en $RIGHT$ entonces dicho punto induce un intervalo de coordenadas x que no puede incluir puntos del rectángulo primal. El algoritmo mantendrá este intervalo de busqueda válido mediante dos variables, L y R, inicialmente $-\infty$ e ∞, respectivamente. Si durante el segundo barrido un punto aparece en el intervalo $[L, R]$ entonces es procesado, en caso contrario, se ignora. Así, al hallar puntos de colores α o β en $LEFT$ o $RIGHT$, reducimos el intervalo $[L, R]$ si esto es posible. La observación 2 nos permite ignorar todos los siguiente puntos de color γ en el barrido, una vez que dicho color está presente en MID. Las observaciones tres y cuatro nos permiten mantener un intervalo de busqueda válido para cada color i, $[LB(i), RB(i)]$. Durante la ejecución del algoritmo estos valores serán actualizados al hallar puntos de color i en $LEFT$ o $RIGHT$. Estos valores son mantenidos *únicamente* si no hay puntos de color i en MID.

Los puntos en $MID, LEFT$ y $RIGHT$ son mantenidos en 3 estructuras de datos DS_{MID}, DS_{LEFT} y DS_{RIGHT}. DS_{LEFT} y DS_{RIGHT} son 2 árboles AVL. Para cada color i estas estructuras almacenan el valor de $LB(i)$ y $RB(i)$ respectivamente. DS_{MID} es un arreglo de bits de tamaño i. Para cada color i este arreglo indica si el algoritmo ha visto ya un punto de dicho color en MID. Si esto sucede entonces no hay entradas para el valor i en DS_{LEFT} y DS_{RIGHT}, gracias a las observaciones del párrafo anterior. El algoritmo usa también otro arreglo de bits, denominado $COLOR$. $COLOR(i) = 1$ sii el algoritmo ha visto un punto de color i en el intervalo $[LB(i), RB(i)]$ en cualquiera de las 3 zonas. Finalmente se mantiene una variable *color_count* que indica el número de entradas distintas de 0 en $COLOR$.

Durante el segundo tipo de barrido, si se halla un punto p_c que satisface alguna de las 4 observaciones hechas arriba entonces no se genera un rectángulo cromático primal. Si se satisface la observación 1 y $p_c \in MID$, entonces se detiene el segundo tipo de barrido y se prosigue a la siguiente iteración en el primer tipo de barrido. Si se satisface la observación 1 y $p_c \in LEFT$ o

$RIGHT$, entonces se actualiza, respectivamente, L o R. Además se descartan los valores en DS_{LEFT} (respectivamente DS_{RIGHT}) que están fuera del intervalo $[L, R]$. Al descartar estos valores, si no hay puntos de los colores descartados en MID o $RIGHT$ (respectivamente $LEFT$), es necesario modificar también la entrada $COLOR(\gamma)$ (pues no existen puntos de color γ en el intervalo $[L, R]$) y decrementar la variable *color_count*.

Si p_c no cumple ninguna de las 4 condiciones citadas arriba, entonces p_c puede estar en $MID, LEFT$ o $RIGHT$. En todos los casos hay que modificar la entrada $COLOR(\gamma)$ y añadir 1 a *color_count*. Si $p_c \in MID$ hay que eliminar las entradas correspondientes a γ en DS_{LEFT} y DS_{RIGHT}. Si $p_c \in LEFT$ (respectivamente $p_c \in RIGHT$) hay que actualizar la entrada $DS_{LEFT}(\gamma)$ (respectivamente $DS_{RIGHT}(\gamma)$). Si *color_count* $< m$ no hay rectángulos cromáticos primales que reportar. En otro caso existen rectángulos primales con p_a y p_b en la base y p_c en el lado paralelo. Los puntos en DS_{LEFT} menores a $x(p_c)$ pueden ser la frontera izquierda de dichos rectángulos primales. Para hallar la frontera derecha de los rectángulos primales es necesario hallar un color θ_R con las siguientes propiedades:

1. $\theta_R \in DS_{RIGHT}$
2. $\theta_R \notin DS_{LEFT} \cup DS_{MID}$
3. $RB(j)$ es máximo con las propiedades 1 y 2 anteriores.

Para determinar θ_R se recorre DS_{RIGHT} en forma descendente. Si no se halla ningún elemento con dichas propiedades, θ_R es α o β, dependiendo de cual de p_a o p_b se encuentre más a la derecha. El color θ_R se denomina *color crítico* derecho. Si p_c está en $RIGHT$ se define de forma análoga el color crítico izquierdo.

Para reportar los rectángulos cromáticos primales se utilizan 2 apuntadores p_l y p_r. p_l itera sobre DS_{LEFT} en orden ascendente hasta p_c. p_r apunta inicialmente a $DS_{RIGHT}(\theta_R)$. El rectángulo con bordes izquierdo y derecho en p_l y p_r es cromático. Además es primal si $\mu \notin DS_{RIGHT}$ o $RB(\mu) > RB(\theta_R)$, donde μ es el color de p_l. En dicho caso se reporta el rectángulo primal y su area. En caso contrario, la iteración de p_l continúa al siguiente elemento hacia la derecha en DS_{LEFT}. Cuando se reporta un rectángulo primal con el color ν en p_l, ν se vuelve el color crítico derecho. De manera que p_r se mueve a $DS_{RIGHT}(\nu)$, si dicho valor existe. En este caso p_l itera al siguiente elemento hacia la derecha y el algoritmo continua. El reporte de rectángulos primales termina cuando p_r no puede desplazarse (pues $DS_{RIGHT}(\nu)$ no existe) o cuando p_l alcanza el valor $LB(\gamma)$. Un método análogo se utiliza cuando $p_c \in RIGHT$.

A partir del anterior algoritmo es posible enunciar el:

Lema 24. *El procesamiento de un vértice bicoloreado de $A(S^*)$ por el algoritmo anterior requiere tiempo $O(nm)$.*

Y a partir de ello el

Teorema 19. *El tiempo total del algoritmo anterior para el problema del rectángulo cromático no orientado es* $O(n^3m)$.

3.4.1. Mejoras al algoritmo

En la parte final de [Das2009] se realizan mejoras al anterior algoritmo. Para ello los autores observan que al encontrar la linea de barrido a un punto p_c (de color γ) que causa la generación de distintos PCSR el algoritmo (i) obtiene el correspondiente *critical_color* (ii) itera sobre cada elemento μ en DS_{left} (si $p_c \in DS_{left}$) y (iii) verifica si el correspondiente rectángulo cromático es un PCSR o no. La generación de rectángulos cromáticos que no son PCSRs puede evitarse mediante la introducción de dos campos de tipo entero, χ y max_χ, en cada elemento de DS_{left} y DS_{right}, y una nueva estructura de datos denominada *sequece_pair* que será descrita más adelante.

Note que la frontera izquierda de un PCSR no puede estar a la derecha de θ_L (el color crítico del lado izquierdo). Similarmente la frontera derecha de un PCSR no puede estar a la izquierda de θ_R. El método para establecer los campos χ de los elementos de DS_{left} es el siguiente. Comience por el elemento más a la izquierda de DS_{left} y proceda hacia la derecha. Los campos χ de los elementos reciben el valor *null* hasta hallar un elemento $LB(\mu) \in DS_{left}$ tal que su correspondiente entrada $RB(\mu) \in DS_{right}$ satisfaga $RB(\mu) > RB(\theta_R)$. Al campo χ de dicho elemento $LB(\mu)$ se asigna el valor $RB(\theta_R)$. Luego se continua asignando a los campos χ de los siguientes elementos de DS_{left} el valor *null* hasta hallar un elemento $LB(\delta) \in DS_{left}$ cuyo correspondiente $RB(\delta) \in DS_{right}$ satisfaga $RB(\delta) > RB(\mu)$. Al campo χ de $LB(\delta)$ se asigna el valor $RB(\mu)$. El proceso continúa hasta llegar a $LB(\theta_L)$. Si el campo χ de $LB(\nu)$ fué el último en asignarse antes de alcanzar a $LB(\theta_L)$ entonces al campo χ de $LB(\theta_L)$ se asigna el valor $RB(\nu)$.

El campo max_χ de cada elemento en $\alpha \in DS_{left}$ (análogamente para DS_{right}) es el máximo (respectivamente mínimo) valor de los campos χ de los elementos en el subárbol de DS_{left} con raíz en α.

Entonces si un color $\mu \in DS_{left}$ cuyo campo χ no es *null* aparece en la frontera izquierda de un rectángulo cromático entonces el rectángulo es un PCSR y su frontera derecha está dada por el campo χ de μ.

A partir de lo anterior, los autores definen la estructura *sequence_pair* izquierda. Esta consiste en una secuencia de duplas (u_i, w_i) para $i = 1, \ldots, k$, donde $u_i \in DS_{left}$ y $w_i \in DS_{right}$. Los valores de u_i y w_i se definen como:

- $u_0 = LB(\theta_L)$. u_0 no pertenece a *sequence_pair*.
- $u_i = LB(\mu)$ y $w_i = RB(\mu)$ donde μ está indicado por el campo χ de u_{i-1}.
- La anterior definición recursiva continúa hasta que $w_i > RB(\theta_R)$.
- Cuando $w_i = RB(\theta_R)$ se tiene que $k = i - 1$ donde k es la longitud de la estructura *sequence_pair*.
- $w_{k+1} = RB(\theta_R)$. w_{k+1} no es un elemento de *sequence_pair*.

La estructura *sequence_pair* derecha se define de forma análoga. A partir de las anteriores definiciones los autores pueden probar el

Lema 25. *Las estructuras sequence_pair izquierda y derecha son iguales.*

y también

Lema 26. *Si existe un PCSR con fronteras inferior y superior dadas por l_{ab} y p respectivamente entonces su lado izquierdo está definido por u_i, donde $(u_i, w_I) \in$ sequence_pair. El lado derecho de dicho PCSR está dado por w_{i+1}, donde (u_{i+1}, w_{i+1}) es el siguiente elemento de sequence_pair.*

Los autores describen además como actualizar la estructura *sequence_pair* durante la ejecución del algoritmo[3]. A partir de ello los autores prueban

Lema 27. *Si se añaden k nuevos elementos a la estructura sequence_pair entonces se generan a lo más $k+1$ nuevos PCSRs*

Lema 28. *La estructura sequence_pair puede ser actualizada en tiempo $O(k \log m)$ donde k es el número de nuevos PCSRs reportados.*

Lema 29. *Al procesar un vértice $v_{ab} \in A(S^*)$ el tiempo total necesitado para actualizar sequence_pair es $O(n \log m)$.*

Lo que lleva a la mejora final:

Teorema 20. *El problema del rectángulo cromático no orientado puede ser resuelto en tiempo $O(n^3 \log m)$.*

3.5. L-corredor cromático

Sea S un conjunto de puntos coloreados en el plano, donde hay k colores distintos. Para un punto $p = (a, b)$ definimos dos regiones:

$$C_p^+ = \{(x, y) | a \leq x, b \leq y\}$$

y

$$C_p^+ = \{(x, y) | a \leq x, b \leq y\}$$

Entonces un L-corredor se define como $C_p^+ \backslash C_q^-$ con $q = (c, d), a \leq c$ y $b \leq d$.

El problema del L-corredor cromático consiste en encontrar un L-corredor que contenga puntos de S de todos los k colores, y tal que minimice el ancho del corredor. Hay diferentes definiciones de ancho para un corredor, lo que origina distintos problemas y distintos algoritmos. Además es también posible dar 2 versiones del problema, dependiendo de si la frontera del corredor debe ser paralela a los ejes coordenados (L-corredor cromático orientado) o no (L-corredor cromático no orientado).

[3]Los detalles son de interés únicamente si se va a realizar una implementación del algoritmo, por lo que el lector interesado puede referirse al artículo original.

3.5.1. Algoritmo de Bautista-Santiago et al.

En [Bautista2009] se resuelven detalle 1 problema y se esbozan las soluciones a otros 2 problemas que involucran L-corredores cromáticos:

L-corredor de ancho mínimo

Un L-corredor $C_{p,q}$ es el conjunto $C_p^+ \backslash C_q^-$ donde $p = (a, b)$ y $q = (c, d)$ definimos $w_x = c - a$ y $w_y = d - b$. En este problema tenemos la restricción $w = w_x = w_y$. En este caso a w se denomina el *ancho* del corredor, y el problema consiste en encontrar un L-corredor de ancho mínimo.

Para cada punto $p = (a, b)$ del plano se definen los rayos horizontal y vertical que parten de p hacia la derecha y hacia arriba, respectivamente $h_p = \{(x, b)|x \geq a\}$ y $v_p = \{(a, y)|y \geq b\}$. De manera que la frontera del corredor $C_{p,q}$ está formada por los rayos h_p, v_p, h_q y v_q. Si $r \in S$ es un punto en la frontera del corredor, decimos que r es *esencial* si no hay otros puntos del mismo color de r en el interior de $C_{p,q}$. Se tiene entonces que $h_p \cup v_p$ contienen al menos un punto esencial, y lo mismo para $h_q \cup v_q$. Es posible siempre obtener un corredor de ancho mínimo con puntos esenciales en h_p y v_p, y con un punto esencial en al menos uno de h_q o v_q.

Para un cuadrante C_p^+ defina $H_p = \{h_1, \ldots, h_k\}$ como el conjunto de los puntos más cercanos a h_p para cada color $i = 1, \ldots, k$. Similarmente defina $V_p = \{v_1, \ldots, v_k\}$ como el conjunto de los puntos más cercanos al rayo v_p para cada color. Definimos una función d_p en C_p^+

$$d_p(r) = \text{mín}\{d(r, h_p), d(r, v_p)\} \text{ para } r \in C_p^+$$

donde la distancia de un punto a un rayo es la longitud del segmento de la perpendicular que une a ambos. Para cada color i sea m_i el punto de color i en C_p^+ que minimiza $d_p(m_i)$, y definamos $M = \{m_1, \ldots, m_k\}$. Entonces es claro que el elemento de M de máxima d_p está en $v_q \cup h_q$.

Definamos las intersecciones de H y V con M:

$$H' = H \cap M$$
$$V' = V \cap M$$

El algoritmo de Bautista-Santiago et al. procesa S realizando dos tipos de barrido. Primero se realiza un barrido vertical en orden descendente según la coordenada y de los puntos. Sea $r = (a, b)$ el punto actual durante el barrido vertical. El algoritmo encontrará el mejor corredor $C_{p,q}$ con $r \in h_p$, visitando todos los puntos $p_i = (x_i, y_i)$ con $x \leq a$ y $y \geq b$ mediante el segundo barrido, el cual procede de derecha a izquierda. Sea p_i el punto actual en el barrido de derecha a izquierda, c el color de p_i, y $p_j = (x_j, y_j)$ el punto de color j en H al momento de que la iteración horizontal llega a p_i. En cada iteración de este segundo barrido hay que realizar, posiblemente, 3 tareas:

1. El punto en la actual iteración pasa a formar parte de H si es más cercano a h_p que el punto actual de dicho color.

2. En cada punto del barrido de derecha a izquierda el algoritmo considera que la frontera v_p del corredor está determinada por el punto iterado (esto es, $p = (x_i, b)$) de manera que en cada iteración del segundo barrido el punto p_i entra a V y por tanto a V' (pues $d_p(p_i) = 0 = d(p_i, v_p)$ en esta iteración).

3. Conforme el barrido procede hacia la izquierda, p_i se aleja cada vez más de v_p, de manera que dejará de estár en V' y el punto de color c en H pasará a formar parte de H'. El momento en que p_i dejará de estar en V' y p_j entrará en H' es el momento en que el barrido vertical pasa por la cordenada $x = x_i - (y_j - b)$. Asociaremos a esta coordenada x un *evento* de color c (la coordenada y del evento es irrelevante). Designaremos E el conjunto de eventos. De manera que cada parada del barrido de derecha a izquierda genera un nuevo evento. Los eventos deben procesarse a la par que los puntos de S en el barrido de izquierda a derecha.

H será representada por el algoritmo como una cola de prioridad implementada mediante un árbol binario balanceado. La llave de cada hoja es su distancia a h_p. Las hojas son mantenidas independientemente en un arreglo indexado por color, de manera que es posible referenciar a la hoja de color i en tiempo constante. Las hojas podrán ser marcadas como activas o inactivas, lo cual señala que un elemento de H pertenece también a M y, por tanto, a H'. Cada nodo del árbol tiene la llave de su descendiente activo con llave de mayor tamaño (esto es, el árbol es, además, un max-heap), de forma que la raíz contiene al elemento de H' más alejado de h_p.

De manera similar V es implementado mediante una cola de prioridad. Las llaves son las coordenadas x de los puntos en V. V comparte las características de la estructura usada para H: sus elementos son marcados como activos e inactivos, V es también un max-heap y sus hojas se mantienen además en un arreglo indexado por color, de forma que se puede acceder a la hoja de color i en tiempo constante.

Finalmente, E se representa mediante una cola de prioridad implementada como un árbol binario balanceado. Las llaves de los elementos son las coordenadas x de los elementos, y el árbol, en forma análoga a las estructuras usadas para V y H, es un max-heap.

Es claro que las operaciones realizadas en cada parada del barrido de izquierda a derecha tiene un costo de $O(\log k)$. Estás deben realizarse para cada punto del barrido, para un tiempo total de $O(n \log k)$. La inicialización de las estructuras H, V y E toma $O(k)$, lo cual es menor que el costo anterior. Finalmente, estas operaciones deben realizarse para cada punto del barrido de arriba a abajo, lo que da el

Teorema 21. *Dado un conjunto S de puntos k-coloreados y en posición general es posible encontrar un L-corredor $C_{p,q}$ k-cromático de ancho mínimo en tiempo $O(n^2 \log k)$.*

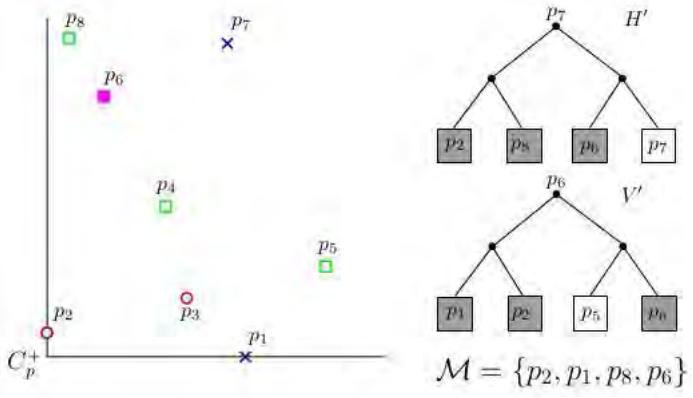


Figura 3.3: Ejemplo de las colas H y V para el algoritmo de Bautista-Santiago et al. [Bautista2009]

L-corredor cromático que minimiza $w_x + w_y$

En este problema el ancho de los eslabones del corredor ya no debe ser el mismo, es decir, se elimina la restricción $w_x = w_y$. Una característica de las soluciones a este problema es que los rayos h_q y v_q deben cada uno contener un punto de S, y ambos puntos son distintos.

La estrategia utilizada es similar a la del problema anterior, pero ahora H y V son, además, conjuntos ordenados respecto a las distancias a h_p y v_p, respectivamente. Las actualizaciones realizadas a cada estructura durante el barrido de izquierda a derecha son las mismas, y tienen la misma complejidad. En cada paso de el barrido de izquierda a derecha se utilizan dichas estructuras para hallar el corredor que minimiza w_x+w_y. Para ello, sean e y e' los puntos por los que pasan h_p y v_p. $e \in V$, $e' \in H$, $e \neq e'$ y ningún elemento que esté antes que e en el orden de V tiene el mismo color que e'. Similarmente, ningún elemento que esté antes de e' en el orden de H tiene el mismo color que e. Estas son condiciones necesarias para e y e', y, para cada e, el correspondiente e' pueden ser determinado en tiempo $O(k)$. Si los elementos de V son procesados en orden es posible calcular para cada e el correspondiente e' en tiempo amortizado $O(k)$. De forma que se tiene el

Teorema 22. *Dado un conjunto S, k-coloreado, es posible determinar un L-corredor cromático orientado que minimiza $w_x + w_y$ en tiempo $O(n^2k)$.*

L-corredor cromático no orientado

En esta versión del problema las fronteras del corredor ya no están sujetas a la restricción de ser paralelas a los ejes coordenados: la orientación del corredor es una variable libre. En forma similar al caso orientado, siempre existe un corredor de ancho mínimo con un punto de S en cada semirrecta de la frontera exterior

(h_p y v_p en los dos anteriores problemas), o bien un punto en la intersección de ambas semirrectas (a dicha intersección se le llama *ápice*).

Sean x y y dos rectas ortogonales orientadas. Un cuadrante es la intersección de los semiplanos izquierdos de x y y (ver la figura 3.4). Observe que si la frontera exterior del cuadrante pasa por 2 puntos, p y q, entonces su ápice puede estar en cualquier punto del semicirculo cuyo diámetro es el segmento pq.

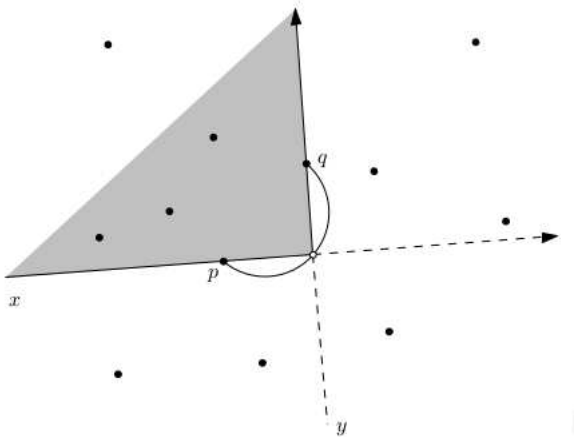


Figura 3.4: Un cuadrante. [Bautista2009]

El algoritmo realizará un barrido con un cuadrante anclado en p y q, lo cual es equivalente a mover el ápice sobre el semicirculo que tiene por diámetro al segmento pq. En cada momento del barrido necesitamos saber, para cada color, el punto dentro del cuadrante más cercano a la frontera exterior del mismo. En el espacio dual el barrido del cuadrante es equivalente a barrer el espacio dual con dos lineas verticales, los cuales parten de x^* y y^*, cuya distancia horizontal es $\pi/2$. Para cualesquiera de dichas 2 rectas se utilizará la envolvente inferior de un arreglo de semirrectas para obtener el punto más cercano de cada color. Para ver esto, observe que durante el barrido con el cuadrante un punto s entra y sale del cuadrante una sola vez. A partir de ello construiremos un arreglo A_i^x de semirrectas: para cada $s \in S$ tomaremos el rayo de s^* compuesto por los puntos de la forma (a, b) tales que, cuando x tiene pendiente a, s está dentro del cuadrante. Cada uno de dichos arreglos tiene $O(n)$ rayos, y el punto en el interior del cuadrante tal que su distancia a x es mínima (cuando x tiene pendiente a) está dado por la envolvente inferior de dichos rayos. De forma similar se construye el arreglo A_i^y para la recta y. Al superponer ambos arreglos, A_i^x y A_i^y, obtenemos un arreglo al que denotaremos A_i, el cual nos da, para cada color i y orientación de las rectas x y y, el punto de color i más cercano a la frontera del cuadrante.

El arreglo A_i tiene complejidad lineal para $i = 1, \ldots, k$ y su envolvente inferior puede calcularse en tiempo $O(n \log n)$[4]. La unión de todas estas envolventes

[4]Usando resultados de M.Sharir, P.K.Agarwal *Davenport-Schinzel sequences and their ge-*

inferiores forma otro arreglo, al que denotamos A. La envolvente superior de A proporciona la solución al problema. El arreglo A tiene un número lineal de segmentos, y su envolvente superior puede calcularse en tiempo $O(n \log n)$. Puesto que dicha envolvente debe calcularse para cada par de puntos de S se obtiene el

Teorema 23. *Dado un conjunto S de puntos en el plano k coloreados en posición general es posible calcular el L-corredor cromático de orientación arbitraria y ancho mínimo en tiempo $O(n^3 \log n)$.*

ometric application. Cambridge University Press, 1995.

Capítulo 4

Corredor cromático de una esquina

En este capítulo mostraremos el trabajo original desarrollado en esta tesis: el *corredor cromático de una esquina*.

4.1. Corredor cromático de una esquina

4.1.1. Definiciones

Sean p y p' dos puntos en $\mathbb{R}^2$, $\vec{v}$ un vector. Considere las semi-rectas

$$l = p + r\vec{v} \qquad r \in \mathbb{R}^+$$
$$l' = p + r\vec{v} \qquad r \in \mathbb{R}^+$$

Junto al segmento pp', l y l' dividen a $\mathbb{R}^2$ en 2 regiones. Un *eslabón* $E = (p, p', l, l')$ (en inglés *link*) es la región convexa del plano acotada por las semirrectas l, l' y el segmento pp', incluyendo ambas lineas y el segmento.

Definimos el ancho de un eslabón como la distancia entre las rectas l y l'.

Sean E, E' dos eslabones. Un *corredor de una esquina* $L = (E, E')$ es la unión de 2 eslabones distintos que comparten los puntos p y p' de los correspondientes eslabones, esto es, $E = (p, p', l, l')$ y $E' = (p, p', l'', l''')$. Si w y w' son los anchos respectivos de los dos eslabones de un corredor de una esquina L, definimos el *ancho de* L como $w + w'$.

El corredor tiene 2 fronteras, ambas formadas por 2 semirrectas que comparten el punto inicial. Una de las fronteras del corredor es convexa respecto al interior del mismo, otra es cóncava. A la frontera convexa la denominaremos *frontera exterior*, a la otra *frontera interior*.

Sea $P = \{p_1, \ldots, p_n\}$ una colección de n puntos en el plano. Decimos que P es *cromática* si cada punto p_i tiene asociado un color $color(p_i)$ de un conjunto

de k posibles colores, esto es:

$$color(p_i) \in \{1, \ldots, k\} \qquad i = 1, \ldots, n$$

Denotaremos $C = \{1, \ldots, k\}$ al conjunto de colores, y

$$C_i = \{p \in P | color(p) = i\}$$

a la *i-esima clase cromática de P.*

Sea L un corredor de una esquina. Se dice que L es *cromático* si contiene puntos de cada uno de los k colores, esto es

$$L \cap C_k \neq \emptyset \qquad k = 1, \ldots, n$$

.

Por el resto de este capítulo asumiremos que los puntos de P se encuentran en posición general:

1. La recta determinada por cualesquiera 2 puntos de P no es vertical.
2. Dadas 3 rectas cualesquiera, cada una de ellas determinadas por 2 puntos de P, su intersección común es vacía.

Dada una colección cromática de puntos en el plano, nuestro objetivo es encontrar un *corredor cromático de una esquina de ancho mínimo.*

4.1.2. Número de puntos que determinan una solución del problema

Por el resto de esta subsección, sea $L = (E, E')$ un corredor cromático de una esquina de ancho mínimo, $E = (p, p', l_1, l_2)$ y $E' = (p, p', l_3, l_4)$ sus eslabones, p y p' los puntos de la definición de eslabón, l_1 y l_2 las rectas que limitan al eslabón E, y l_3 y l_4 las rectas que limitan al eslabón E'.

Lema 30. *Existen 4 puntos distintos $p_i \in P$ con*

$$p_i \in l_i$$

Ademas si $i \neq j$ entonces

$$color(p_i) \neq color(p_j)$$

Demostración. Si alguno de los eslabones, digamos E, no tiene un punto $p \in P$ en una de las rectas l que lo definen, podemos reducir el ancho de dicho eslabón trasladando a l hasta que incida en alguno de los puntos del interior de L. El corredor así obtenido sigue siendo cromático y tiene un ancho menor al de L.

Para probar la segunda parte del lema, asuma que $color(p_i) = color(p_j)$. Si trasladamos l_i hacia el interior de E hasta que incida en un punto p con

$$color(p) \neq color(p_j)$$

obtendremos, nuevamente, un corredor cromático de ancho menor al de L. □

Lema 31. *Sean p_1 y p_2 los dos puntos en las rectas l_1 y l_2 que limitan a uno de los 2 eslabones de L, digamos, E. Entonces se cumple una de las siguientes 2 condiciones:*

1. *O bien E no contiene más puntos que p_1 y p_2 y E es un eslabón degenerado consistente en una semirrecta.*
2. *O bien existe un tercer punto $p_3 \in P$ que satisface*

$$p_3 \in l_1 \cup l_2$$
$$color(p_3) \neq color(p_1)$$
$$color(p_3) \neq color(p_2)$$

Demostración. Suponga que no existe tal punto p_3 en las rectas l_1 y l_2. Considere el segmento p_1p_2. Dos de los cuatro ángulos que forma con las rectas l_1 y l_2 e interior al corredor L son agudos. Denotemos a uno de dichos ángulo β. Si los 4 ángulos que forma el segmento p_1p_2 con las rectas l_1 y l_2 e interiores a L son rectos, entonces β puede ser cualquiera de ellos. Si hacemos girar ambas rectas en el mismo sentido alrededor de p_1 y p_2 respectivamente, de forma que β se reduzca, obtenemos un corredor cromático de ancho menor al de L(figura 4.1). Observe que podemos hacer girar ambas rectas hasta que se cumpla una de dos condiciones:

1. Si no hay otros puntos además de p_1 y p_2 dentro del eslabón E, entonces podemos girar ambas rectas hasta que $\beta = 0$, lo que nos da un eslabón degenerado formado por una semirrecta, o bien
2. Alguna de las rectas l_1 o l_2 incide en un tercer punto p_3 que cumple $color(p_3) \neq color(p_i), i = 1, 2$(pues si $color(p_3) = color(p_i)$ para alguna $i \in \{1, 2\}$ podríamos continuar disminuyendo β y mantener la cromaticidad de L a pesar de que p_3 haya dejado de estar en el interior de L).

□

Por el lema 31 necesitamos 3 puntos de distintos colores en cada una de las 2 rectas que determinan cada eslabón. Antes de proseguir es conveniente hacer la siguiente

Observación 3. *En un corredor de ancho mínimo eslabones E y E' pueden compartir el punto en la esquina interior del corredor, pero no la externa.*

En la figura 4.2 vemos, en lineas continuas, un corredor cuyos eslabones comparten un punto G en su frontera interna. En lineas punteadas vemos como puede ser reducido el ancho de dicho corredor conservando la cromaticidad al trasladar una de las semirrectas que limita a uno de los eslabones. Note que semejante reducción del ancho del corredor es imposible de realizar – mediante traslaciones y rotaciones – si el punto de P que los eslabones comparten se encuentra en la frontera externa.

Finalmente, los 3 puntos en la frontera de cada eslabón tienen una característica importante:

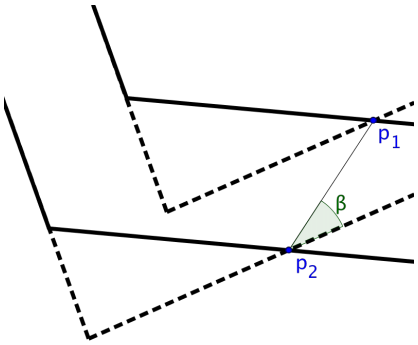


Figura 4.1: En lineas punteadas, el corredor al reducir el ángulo β.

.

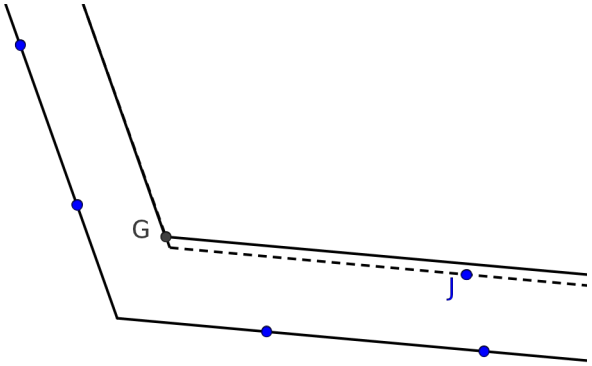


Figura 4.2: En lineas punteadas, el corredor al reducir el ancho de un eslabón.

.

Lema 32. *Sean $p_1, p_2, p_3 \in P$ los 3 puntos en la frontera de uno de los eslabones, digamos E. Entonces el triángulo $\triangle p_1p_2p_3$ es acutángulo*[1] *o rectángulo.*

Demostración. Por reducción al absurdo. Suponga que el triángulo $\triangle p_1p_2p_3$ es obtusángulo. Sean p_1 y p_2 los dos puntos que se encuentran sobre la misma linea l que limita al eslabón E. El segmento p_2p_3 forma con l dos ángulos, uno de los cuales es agudo. Denotemos como α a dicho ángulo y sea l' la otra linea que limita al eslabón E. Si hacemos rotar a l y l' respectivamente alrededor de p_2 y p_3 de manera que reduzcamos a α obtendremos un corredor de menor ancho que L. Vea la figura 4.3. □

Entonces un corredor cromático de una esquina queda determinado por 6 o 5 puntos en las fronteras de sus eslabones. Esto nos sugiere un sencillo algoritmo de búsqueda por fuerza bruta del corredor cromático de una esquina de ancho

[1] Un triángulo es *acutángulo* si todos sus ángulos interiores son agudos.

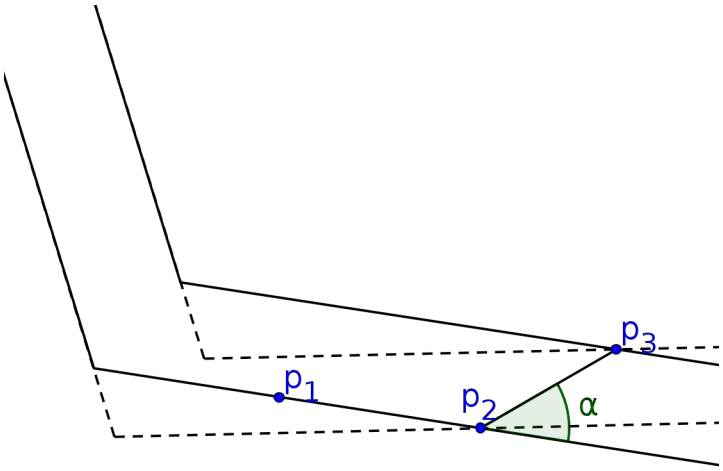


Figura 4.3: En lineas punteadas, el corredor al rotar l y l'.

.

mínimo: en $O(n^6)$ formamos un corredor de una esquina, y después verificamos que contenga puntos de cada color, lo que nos toma tiempo adicional $O(n)$, manteniendo registro del corredor de menor ancho. Eso nos da el siguiente

Teorema 24. *Dado un conjunto de n puntos coloreados $P = \{p_1, \ldots, p_n\}$ es posible obtener, mediante búsqueda por fuerza bruta, un corredor cromático de ancho mínimo en $O(n^7)$.*

4.2. Características de la solución en el espacio dual

Es claro que un corredor no vertical

$$\{(x, y)|y = ax + b, a \neq 0, b_o \leq b \leq b_1\}$$

se convierte en el espacio dual en un segmento vertical

$$\{(x, y)|x = a, -b_1 \leq y \leq -b_0\}$$

De manera que 2 de tales segmentos segmentos verticales representan un par de corredores no verticales, esto es, los dos corredores

$$C_0 = \{(x, y)|y = a_0x + b_0, a_0 \neq 0, b_1 \leq b_0 \leq b_2\}$$
$$C_1 = \{(x, y)|y = a_1x + b_1, a_1 \neq 0, b_3 \leq b_1 \leq b_4\}$$

se dualizan en los segmentos verticales

$$C_0^* = \{(x, y)|x = a_0, -b_2 \leq y \leq -b_1\}$$
$$C_1^* = \{(x, y)|x = a_1, -b_4 \leq y \leq -b_3\}$$

los cuales tienen intersección no vacía siempre que

$$a_0 \neq a_1$$

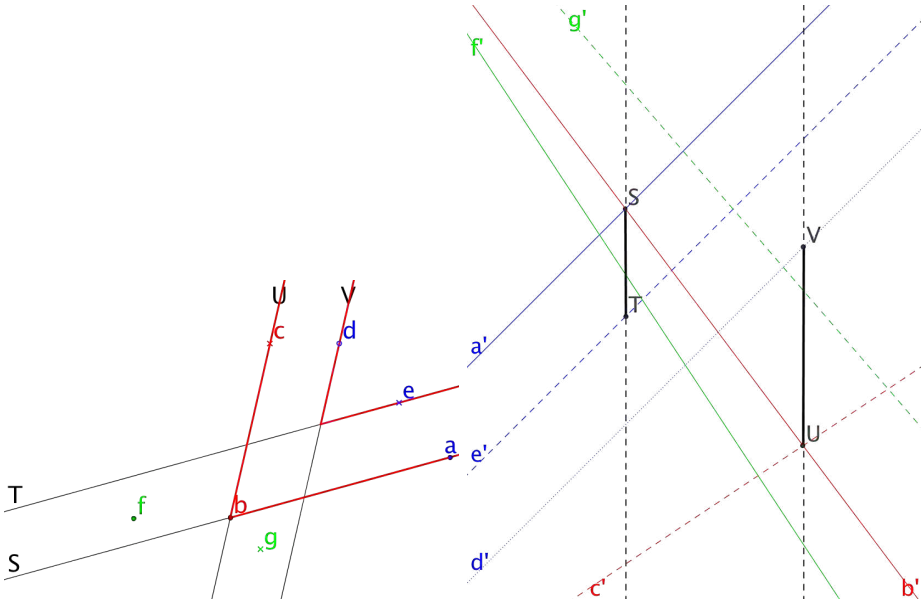


Figura 4.4: Del lado izquierdo el espacio primal. Hay 7 puntos y 7 colores, estos están dados por la combinación de color y forma de punto/linea.

¿Como distinguir entonces está unión arbitraria de 2 corredores cuya intersección es no vacía (una *cruz*, ver figura 4.4), de un verdadero corredor de una esquina? La respuesta puede observarse en la figura: hay puntos en el lado equivocado de las fronteras exteriores. En el espacio primal, el punto f está del lado equivocado de la recta U (la frontera exterior del corredor, en rojo), y el punto g se encuentra también en el lado equivocado de la recta S (nuevamente, la frontera exterior del corredor). En el espacio dual podemos observar esto también: la intersección de f' con la vertical que pasa por U y V se encuentra del lado contrario de U al que se encuentran las demás intersecciones. También la intersección de g' con la vertical que pasa por S y T se encuentra del lado de T contrario a donde se encuentran el resto de las intersecciones del corredor. Es fácil notar que estas condiciones son necesarias y suficientes para distinguir un corredor de una cruz. Con base en las anteriores observaciones, podemos caracterizar el dual de un corredor cromático de una esquina:

Teorema 25. *Dos segmentos verticales en el espacio dual*

$$C_0^* = \{(x,y)|x = a_0, b_2 \leq y \leq b_1\}$$
$$C_1^* = \{(x,y)|x = a_1, b_4 \leq y \leq b_3\}$$

Corresponden a un corredor cromático de una esquina en el espacio primal si y solo si

1. $a_0 \neq a_1$

2. C_0 *y* C_1 *tienen, juntos, intersecciones con rectas de todos los colores.*

3. *Sucede alguna de las siguientes 4 opciones:*

   a) *Para toda recta* $l = ax - b$ *cuya intersección con* C_0 *es no vacía se tiene que*

   $$aa_1 - b \leq b_3$$

   y para toda recta $l = ax - b$ *cuya intersección con* C_1 *es no vacía se tiene que*

   $$aa_0 - b \leq b_1$$

   b) *Para toda recta* $l = ax - b$ *cuya intersección con* C_0 *es no vacía se tiene que*

   $$b_4 \leq aa_1 - b$$

   y para toda recta $l = ax - b$ *cuya intersección con* C_1 *es no vacía se tiene que*

   $$aa_0 - b \leq b_1$$

   c) *Para toda recta* $l = ax - b$ *cuya intersección con* C_0 *es no vacía se tiene que*

   $$aa_1 - b \leq b_3$$

   y para toda recta $l = ax - b$ *cuya intersección con* C_1 *es no vacía se tiene que*

   $$b_2 \leq aa_0 - b$$

   d) *Para toda recta* $l = ax - b$ *cuya intersección con* C_0 *es no vacía se tiene que*

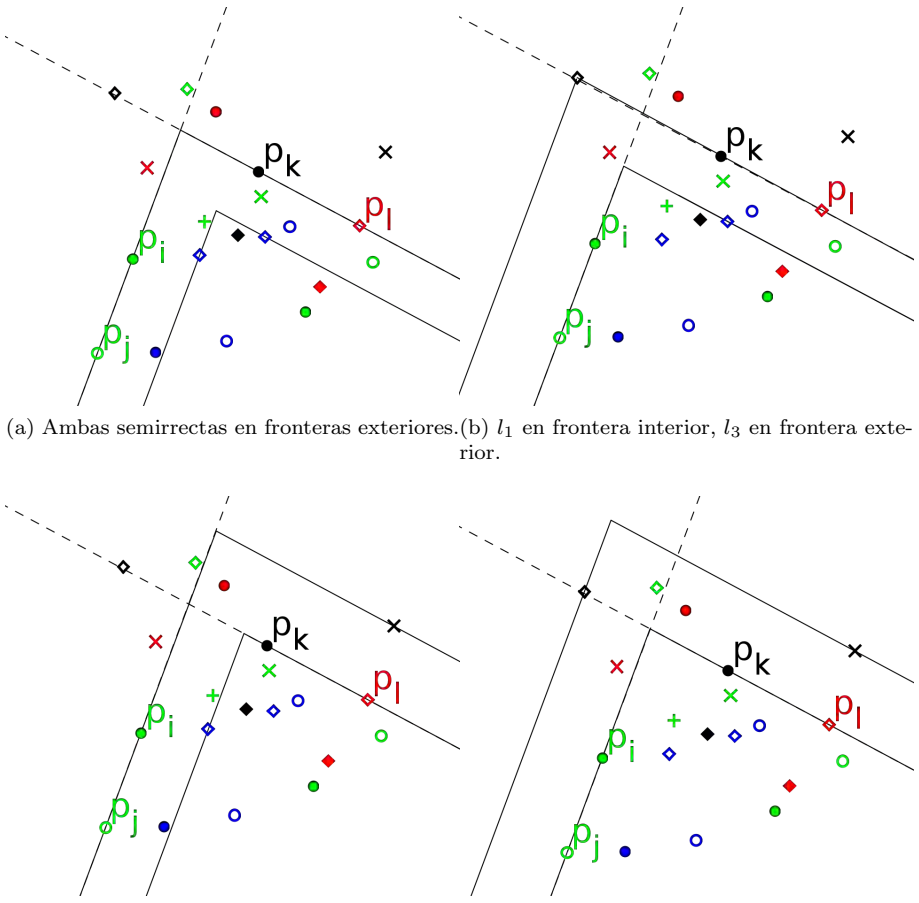
   $$b_4 \leq aa_1 - b$$

   y para toda recta $l = ax - b$ *cuya intersección con* C_1 *es no vacía se tiene que*

   $$b_2 \leq aa_0 - b$$

4.3. El algoritmo

En virtud del lema 2, si $L = (E, E')$ es un corredor cromático de ancho mínimo, donde $E = (p, p', l_1, l_2)$ y $E' = (p, p', l_3, l_4)$ son los dos eslabones que forman al corredor L, sabemos que dos de las semirrectas l_1, l_2, l_3, l_4 tendrán, cada una, 2 puntos de P. Suponga que son l_1 y l_3 las dos rectas que tienen, cada una, dos puntos de P. Denotemos dichos puntos por p_i, p_j, p_k, p_l, con $p_i, p_j \in l_1$ y $p_k, p_l \in l_3$. Entonces hay cuatro posibles tipos de corredor que pueden formarse, dependiendo de si las semirrectas l_1 y l_3 están en la frontera interna o externa del corredor. Vea la figura 4.5.



(a) Ambas semirrectas en fronteras exteriores.

(b) l_1 en frontera interior, l_3 en frontera exterior.

(c) l_1 en frontera exterior, l_3 en frontera interior.

(d) Ambas semirrectas en frontera interior.

Figura 4.5: Los cuatro tipos de corredores.

1. Las semirrectas l_1 y l_3 se encuentran, ambas, en las fronteras exteriores

del corredor. Ver figura 4.5a. Denominaremos a este caso *exterior-exterior*

2. La semirrecta l_1 se encuentra en frontera interior, la semirrecta l_3 en frontera exterior. Ver figura 4.5b. Denomiremos a este caso *interior-exterior*.

3. La semirrecta l_1 se encuentra en frontera exterior, la semirrecta l_3 en frontera interior. Ver figura 4.5c. Denominaremos a este caso *exterior-interior*.

4. Ambas semirrectas se encuentran en la frontera interior. Ver figura 4.5d. Denominaremos a este caso *interior-interior*.

Dados los 4 puntos p_i, p_j, p_k, p_l nuestro algoritmo encuentra un corredor cromático de ancho mínimo en $O(k^2 \log n)$. Los 4 posibles casos para las lineas l_1 y l_3 son resueltos de manera similar.

Corredor tipo *interior-interior*

Sean L_1 y L_3 las lineas que contienen, respectivamente, a las semirrectas l_1 y l_3. Solo uno de los 2 semiplanos abiertos determinados por L_1 interseca al interior de L (si este es no vacio). Denotemos a dicho semiplano por s_1, sea $\bar{s_1}$ el otro semiplano abierto determinado por L_1 y sean s_3 y $\bar{s_3}$ los semiplanos análogos determinados por L_3. Estos semiplanos determinan 3 regiones del plano que pueden contener puntos de P y deben ser consideradas:

1. $s_1 \cap \bar{s}_3$. Denotaremos a dicha región I.

2. $\bar{s}_1 \cap s_3$. Denotaremos a dicha región II.

3. $s_1 \cap s_3$. Denotaremos a dicha región III.

Un ejemplo de las 3 regiones puede verse en la figura 4.6.

El algoritmo analizará un conjunto de corredores cromáticos con los puntos p_i, p_j, p_k y p_l en las 2 fronteras interiores respectivas, entre los cuales se encuentra el corredor cromático de ancho mínimo. Para ello necesitaremos algunas definiciones. Si consideramos cualquiera de los 2 eslabones del corredor, por ejemplo E, y los tres puntos que limitan su frontera p_i, p_j y q, donde $p_i, p_j \in l_1$, entonces, por el lema 32, q satisface que $\triangle p_i p_j q$ es rectángulo o acutángulo. Es claro que dicho punto debe ser el más cercano de su color en la región correspondiente (I o II respectivamente). El algoritmo utilizará 2 arreglos, f_I y f_{II}, que indican, para cada color a, el punto más cercano en I o II:

$$f_I(a) = \begin{cases} p & \text{sii } p \in I \cap C_a \text{ y } d(p, L_1) = \text{mín}\{d(q, L_1) | q \in I \cap C_a\} \\ null & \text{sii } I \cap C_a = \emptyset \end{cases}$$

$$f_{II}(a) = \begin{cases} p & \text{sii } p \in II \cap C_a \text{ y } d(p, L_3) = \text{mín}\{d(q, L_3) | q \in II \cap C_a\} \\ null & \text{sii } II \cap C_a = \emptyset \end{cases}$$

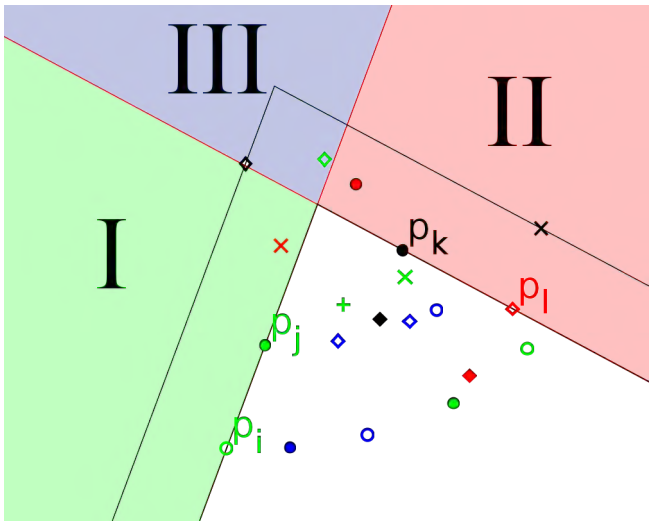


Figura 4.6: Las lineas L_1 y L_3 en rojo. En negro, un ejemplo de corredor *interior-interior*. En sombreado verde, rojo y azul las zonas I, II y III, respectivamente
.

Podemos determinar el arreglo f_I (respectivamente f_{II}) en $O(n)$, recorriendo todos los puntos de S, calculando sus distancias a L_1 (respectivamente L_3) y manteniendo en f_I (respectivamente f_{II}) los elementos minimos de cada color. Note que no todos los puntos en el arreglo f_I (respectivamente f_{II}) pueden determinar la otra frontera de E (respectivamente E'), pues, como señalamos más arriba, la condición necesaria para esto es que dicho punto determine, junto a p_i y p_j (respectivamente p_k y p_l), un triángulo acutángulo o rectángulo.

Un corredor puede contener puntos de color a aun si el ancho de E es menor que $d(f_I(a), L_1)$ y el ancho de E' es menor que $d(f_{II}(a), L_3)$, pues pueden existir puntos p en la región III tales que $color(p) = a$, $d(p, L_1) < d(f_I(a), L_1)$ y $d(p, L_3) < d(f_{II}(a), L_3)$ (vea la figura 4.7). Para determinar la existencia de tales puntos el algoritmo mantiene, para cada color a, los puntos en III de color a en un árbol T^a_{III} de búsqueda de rangos de 2 dimensiones (las distancias a L_1 y L_3), de forma que es posible determinar, para 2 anchos determinados de los eslabones E y E' (y por tanto 2 distancias a las rectas L_1 y L_3), si el corredor con tales anchos de eslabones contiene un punto de color a en la región III en tiempo $O(\log n)$. Note que, para cada color a, solo nos interesan los puntos $p \in III$ que satisfacen $d(p, L_1) < d(f_I(a), L_1)$ y $d(p, L_3) < d(f_{II}(a), L_3)$ pues si el ancho del eslabón E (respectivamente E') es mayor o igual que $d(f_I(a), L_1)$ (respectivamente $d(f_{II}(a), L_3)$) entonces el eslabón contiene a $f_I(a)$ (respectivamente $f_{II}(a)$), lo cual puede determinarse sin necesidad de consultar al árbol T^a_{III} .

Una primera idea para hallar el corredor de ancho mínimo de tipo *interior-interior* sería incluir al menor de $f(I, k)$ y $f(II, k)$ para cada k. Como puede

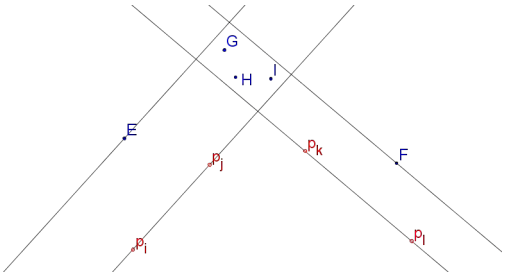


Figura 4.7: E y F son los puntos azules más cercanos en I y II. G, H o I son azules también, y permiten la existenciá de corredores con puntos azules y anchos menores a $f_I(c)$ y $f_{II}(c)$ donde c es el color azul

verse en la figura 4.8 esta idea no lleva a un corredor mínimo.

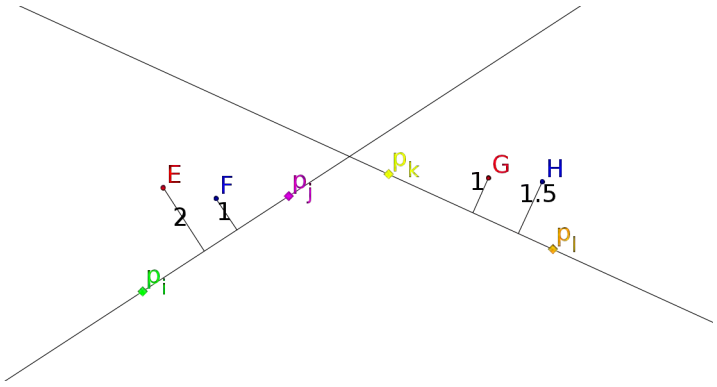


Figura 4.8: El corredor de ancho mínimo usa los puntos G, H. Los segmentos que unen a los puntos con la frontera interior del corredor están etiquetados con su longitud.

De manera que para determinar el corredor cromático de ancho mínimo de tipo *interior-interior* con p_i, p_j, p_k y p_l en las respectivas fronteras, será necesario explorar todos los candidatos a ser el tercer punto en las fronteras de cada eslabón del corredor. Para ello usaremos 2 listas ordenadas. La primera de ellas, L_I, contiene los candidatos a ser la frontera en la región I, es decir, contiene a los $f_I(a)$, ordenados de forma ascendente según su distancia a L_1.

Observe que para que un elemento $f_I(c)$ en f_I sea el otro punto en la frontera de E debe cumplir con las siguientes características:

1. $f_I(c) \neq null$.
2. El triángulo $\triangle f_I(c) p_i p_j$ es rectángulo o equilatero.
3. T^c_{III} es vacío.

Solo la tercera condición requiere ser comentada. Basta observar que si T^c_{III} es no vacío esto quiere decir que existen puntos en la región 3 que son más cercanos a L_1 (o a L_2) que $f_I(c)$ (o que $f_{II}(c)$). De manera que un corredor cuya frontera pasara por $f_I(c)$ o $f_{II}(c)$ puede prescindir tanto de $f_I(c)$ como de $f_{II}(c)$ sin dejar de ser cromático, disminuyendo con ello su ancho.

Los puntos $f_I(c) \in L_I$ que satisfacen las anteriores 3 condiciónes serán llamados *válidos*. El algoritmo itera únicamente sobre los elementos válidos de L_I.

En otra lista ordenada, L_{II} tendremos los puntos análogos de la región II. Los puntos *válidos* de L_{II} se definen en forma análoga.

El algoritmo utiliza dos apuntadores, p_I y p_{II}, los cuales son el otro punto que limita a los eslabones E y E', esto es: p_i, p_j y p_I limitan al eslabón E, y p_k, p_l y p_{II} limitan al eslabón E'. El algoritmo mantiene en todo momento memoria de los puntos que están en la frontera del mejor candidato encontrado hasta el momento, y el ancho del mejor corredor hallado hasta el momento, inicialmente ∞.

1. p_I itera en forma descendente sobre los elementos válidos de L_I sin ascender jamás, comenzando desde el último elemento, el más alejado de la línea L_1.

2. p_{II} inicialmente es p_k (de manera que el eslabón E' es unicamente la semirrecta l_3 que pasa por p_k y p_l), posteriormente itera ascendentemente sobre los elementos de L_{II} sin descender jamás.

3. Si el corredor determinado no es cromático

   a) Iterar p_{II} hasta hallar un corredor cromático. Para determinar si el corredor actual es cromático en cada valor nuevo de p_{II} se hacen 3 consultas por cada color c:

      1) ¿Es el ancho del eslabón E mayor que $d(f_I(c), L_1)$?
      2) ¿Es el ancho del eslabón E' mayor que $d(f_{II}(c), L_2)$?
      3) ¿Existen elementos en T^c_{III} para el rango determinado por los anchos de los eslabones E y E', esto es, en el rango $[0, d(p_I, L_1)] \times [0, d(p_{II}, L_2)]$?

      Una respuesta afirmativa a cualquiera de las 3 preguntas implica que el corredor contiene puntos del color c. Una vez se encuentra un corredor cromático:

      1) Se determina su ancho y, en caso que sea el mejor corredor hallado hasta el momento, se actualizán los valores correspondientes.
      2) p_I desciende 1 punto en la lista L_I. El algoritmo procede a partir del inciso a).

Al concluir las iteraciones sobre p_{II} en a) tenemos un corredor cromático. Esto y el hecho de que p_I itera descendentemente por todos los elementos válidos

de L_I nos garantiza que exploraremos todos los corredores que cumplen las condiciones para ser de ancho mínimo.

En cuanto al tiempo de ejecución del algoritmo, observe que las 3 consultas hechas por cada color toman tiempo $O(\log n)$ en el peor caso. Estas se deben realizar para cada uno de los k colores, lo que da $O(k \log n)$. Finalmente, observemos que p_I asciende siempre y p_{II} desciende siempre, lo cual da $O(k)$ posibles combinaciones. Así, el tiempo de ejecución total del anterior algoritmo es $O(k^2 \log n)$. Lo que nos lleva al

Lema 33. *Para p_i, p_j, p_k y p_l dados es posible determinar el corredor cromático de ancho mínimo de tipo interior-interior en tiempo $O(k^2 \log k)$.*

Corredor tipo *exterior-interior* e *interior-exterior*

Puesto que estos casos son simétricos, trataremos únicamente el primero.

Nuevamente, sean L_1 y L_3 las rectas que contienen, respectivamente, a las semirrectas l_1 y l_3. Sean s_1 y s_3 los semiplanos definidos respectivamente por L_1 y L_3, de forma analoga al caso anterior, y $\bar{s}_1$ y $\bar{s}_3$ sus complementos respectivos. Sea L' la paralela a L_1 que pasa por p_k, y E el eslabón que contiene a l_1 en su frontera exterior. Sean s y $\bar{s}$ los dos semiplanos abiertos determinados por L', donde s es el semiplano que contiene a p_i y p_j. Puesto que p_k y p_l se encuentran en la frontera interna del otro eslabón E'', no todo $s_1 \cap \bar{s}_3$ contiene candidatos a formar la frontera interior del eslabón E'. En la figura 4.9 puede apreciarse que la región del plano que contiene puntos que pueden determinar a la recta l_2 (la cual forma junto a l_1 las fronteras del eslabón E') es precisamente $s_1 \cap \bar{s}_3 \cap s$. De manera que tenemos nuevamente 3 regiones:

1. Denotaremos como I a la región $s_1 \cap \bar{s}_3 \cap s$.
2. Denotaremos II a la región $s_3 \cap \bar{s}$, la cual contiene únicamente puntos de E''.
3. Finalmente denotaremos III a la región $s_1 \cap s_3 \cap s$.

Puede ver un ejemplo en la figura 4.9.

Como en el caso *interior-interior*, necesitamos en primer término los puntos más cercanos de cada color en I y II, los cuales se definen en forma análoga:

$$f_I(k) = \begin{cases} p & \text{sii } p \in I \cap C_k \text{ y } d(p, L_1) = \text{mín}\{d(q, L_1) | q \in I \cap C_k\} \\ null & \text{sii } I \cap C_k = \emptyset \end{cases}$$

$$f_{II}(k) = \begin{cases} p & \text{sii } p \in II \cap C_k \text{ y } d(p, L_3) = \text{mín}\{d(q, L_3) | q \in II \cap C_k\} \\ null & \text{sii } II \cap C_k = \emptyset \end{cases}$$

Como en el caso anterior, un algoritmo glotón no resuelve el problema, lo cual puede mostrarse con una construcción análoga. De tal suerte que la estrategia a utilizar será la misma: utilizaremos 2 listas ordenadas L_I y L_{II} con los elementos válidos de f_I y f_{II}. Los elementos válidos cumplen las mismas condiciones que

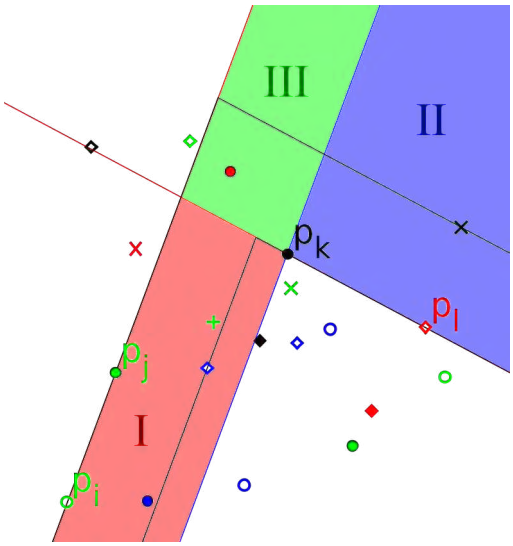


Figura 4.9: Las lineas L_1 y L_3 en rojo. En azul la linea L'. En negro, un ejemplo de corredor *interior-interior*. En sombreado rojo, azul y verde las zonas I, II y III, respectivamente

.

en el caso anterior. Como en el caso anterior, utilizaremos un arbol T^c_{III} por cada color c para determinar si hay puntos de color c en la región III para un rango bidimensional (correspondiente a los anchos de los eslabones E y E'). Una vez contamos con las listas L_I, L_{II} y los árboles T^c_{III} para todos los colores c el algoritmo a utilizar es el mismo que en el caso anterior.

De esta manera es posible encontrar los corredores de tipo *exterior-interior* y *interior-exterior* en tiempo $O(k^2 \log n)$ para p_i, p_j, p_k y p_l dados. Tenemos entonces el

Lema 34. *Para p_i, p_j, p_k y p_l dados es posible determinar el corredor cromático de ancho mínimo de tipo interior-exterior y exterior-interior en tiempo $O(n^2 \log n)$.*

Corredor tipo *exterior-exterior*.

En forma análoga a los casos anteriores sean L_1 y L_3 las rectas que contienen, respectivamente, a las semirrectas l_1 y l_3. Sean s_1 y s_3 los semiplanos definidos respectivamente por L_1 y L_3, de forma analoga a los casos anteriores, y $\bar{s}_1$ y $\bar{s}_3$ sus complementos respectivos. Es claro que todos los candidatos a ser el otro punto en la frontera de los 2 eslabones del corredor se encuentran en $s_1 \cap s_3$ (vea la figura 4.10). Esto hace al corredor de tipo exterior-exterior el más sencillo de los 4 tipos de corredores analizados, lo cual será reflejado, como veremos, en un tiempo de ejecución menor al de los casos anteriores.

Al igual que en los casos anteriores necesitaremos los puntos más cercanos de cada color respecto a cada una de las rectas L_1 y L_3:

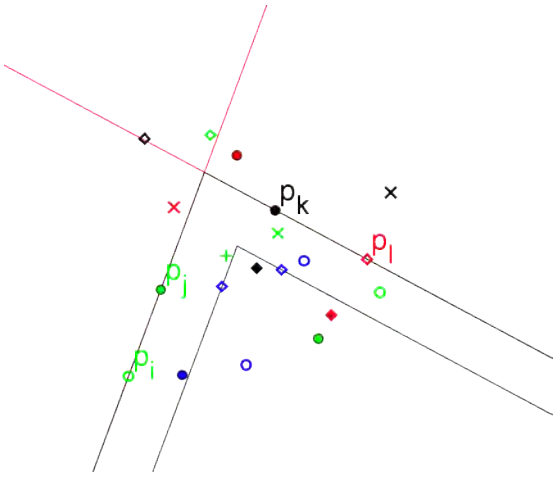


Figura 4.10: Las lineas L_1 y L_3 en rojo. En negro, un ejemplo de corredor *exterior-exterior*.
.

$$f_{L_1}(k) = \begin{cases} p & \text{sii } p \in s_1 \cap s_3 \cap C_k \text{ y } d(p, L_1) = \text{mín}\{d(q, L_1) | q \in s_1 \cap s_3 \cap C_k\} \\ null & \text{sii } s_1 \cap s_3 \cap C_k = \emptyset \end{cases}$$

$$f_{L_3}(k) = \begin{cases} p & \text{sii } p \in s_1 \cap s_3 \cap C_k \text{ y } d(p, L_3) = \text{mín}\{d(q, L_3) | q \in s_1 \cap s_3 \cap C_k\} \\ null & \text{sii } s_1 \cap s_3 \cap C_k = \emptyset \end{cases}$$

Como en los casos anteriores, un algoritmo glotón no resuelve el problema, lo cual puede mostrarse con una construcción análoga. De manera que emplearemos la misma estrategia: utilizaremos 2 listas ordenadas L_I y L_{II} con los elementos válidos de f_{L_1} y f_{L_3}. Los elementos válidos cumplen las mismas condiciones que en el caso anterior. En esta ocasion no es necesario utilizar árboles de rango para cada color , pues todos los puntos se encuentran en la misma intersección de semiplanos determinados por L_1 y L_3 . Una vez que tenemos las listas L_I y L_{II}, el algoritmo a utilizar es el mismo que en el caso anterior, solamente que ahora se omiten las consultas a los árboles de rangos:

1. p_I itera en forma descendente sobre los elementos válidos de L_I sin ascender jamás, comenzando desde el último elemento, el más alejado de la línea L_1.

2. p_{II} inicialmente es p_k (de manera que el eslabón E' es unicamente la semirrecta l_3 que pasa por p_k y p_l), posteriormente itera ascendentemente sobre los elementos de L_{II} sin descender jamás.

3. Si el corredor determinado no es cromático

a) Iterar p_{II} hasta hallar un corredor cromático. Para determinar si el corredor actual es cromático en cada valor nuevo de p_{II} se hacen 2 consultas por cada color c:

1) ¿Es el ancho del eslabón E mayor que $d(f_{L_1}(c), L_1)$?
2) ¿Es el ancho del eslabón E' mayor que $d(f_{L_3}(c), L_2)$?

Una respuesta afirmativa a cualquiera de las 2 preguntas implica que el corredor contiene puntos del color c. Una vez se encuentra un corredor cromático:

1) Se determina su ancho y, en caso que sea el mejor corredor hallado hasta el momento, se actualizán los valores correspondientes.
2) p_I desciende 1 punto en la lista L_I. El algoritmo procede a partir del inciso a).

De manera que ahora solo se realizan 2 consultas por cada color, las cuales toman tiempo $O(1)$. Estas se deben realizar para cada uno de los k colores, lo que da $O(k)$. Finalmente, observemos que p_I asciende siempre y p_{II} desciende siempre, lo cual da $O(k)$ posibles combinaciones. Así, el tiempo de ejecución total del anterior algoritmo es $O(n^2)$, para p_i, p_j, p_k y p_l dados. Así tenemos el

Lema 35. *Para p_i, p_j, p_k y p_l dados es posible determinar el corredor cromático de ancho mínimo de tipo exterior-exterior en tiempo $O(k^2)$.*

Los tres lemas anteriores nos llevan entonces al resultado principal

Teorema 26. *Dado un conjunto de puntos k-coloreados es posible determinar el corredor cromático de una esquina de ancho mínimo en tiempo $O(n^4k^2 \log n)$.*

Conclusión

Mediante el uso de técnicas de geometría computacional fue posible determinar el corredor cromático de ancho mínimo de una esquina en tiempo $O(n^6 \log n)$. Esto constituye una mejora por un factor de $\frac{k^2 \log n}{n^2}$ sobre el algoritmo de busqueda exhaustiva trivial para dicho problema. Con este nuevo resultado se cuenta ahora con algoritmos para las versiones cromáticas y no cromáticas de los problemas:

1. Corredor.
2. l-corredor.
3. Corredor de una esquina.

Además se dió una caracterización de la solución del problema del corredor de una esquina en el espacio dual. Si bien dicha caracterización es sencilla, no existe en la literatura consultada sobre el tema, salvo para el caso del corredor simple.

Adicionalmente se realizó una recapitulación de diversos resultados que involucran corredores y rectángulos en conjuntos de puntos.

Bibliografía

[Abellanas2001] Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop y Vera Sacristán. *Smallest color-spanning objects*. ESA '01 Proceedings of the 9th Annual European Symposium on Algorithms. Pages 278-289. Springer-Verlag London, UK 2001 .

[Bautista2009] C. Bautista, J.Cano, J.M. Díaz-Báñez, H. González-Aguilar, D. Lara, J. Urrutia. *k-Chromatic L-Corridor*. Proc. XIII Encuentros de Geometría Computacional - ECG'09, Zaragoza, Spain, 2009.

[Brodal2003] Gerth Stølting Brodal, Riko Jakob *Dynamic planar convex hull*. Borrador enviado a Elsevier Science. Accesible en internet en `www.brics.dk/~gerth` y `www.brics.dk/~jakob`.

[Cheng1996] Siu-Wing Cheng. *Widest-empty L-shaped corridor*. Information processing letters 58 (1996). 277-283.

[Das2009] Sandip Das, Partha P. Goswami, Subhas C. Nandy. *Smallest color-spanning object revisited*. International Journal of Computational Geometry and Applications. Vol. 19, No. 5, pp. 457-478. 2009.

[Das2009a] Gautam K. Das, Debapriyay Mukhopadhyay, Subhas C. Nandy. *Improved algorithm for the widest empty 1-corner corridor*. Information Processing Letters 109 (2009) 1060-1065.

[DeBerg2008] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars. *Computational Geometry. Algorithms and applications.* 3rd edition. Springer-Verlag 2008.

[DiazBanez2006] J. M. Díaz-Bañez, M. A. López, J. A. Sellarès. *On finding a widest empty 1-corner corridor*. Information Processing Letters 98 (2006) 199-205.

[Goswami2004] Partha P. Goswami, Sandip Das, Subhas C. Nandy. *Triangular range counting query in 2D and its application in finding k nearest neighbors of a line segment*. Computational Geometry 29 (2004) 163-175.

[Janardan1994] Ravi Janardan, Franco P. Preparata. *Widest corridor problems.* Nordic Journal of Computing 1(1994), 231-245

[Lee1985] D. T. Lee, Y. T. Chang. *The power of geometric duality revisited.* Information processing letters 21 (1985) 117-122.