Poligonales heterocromáticas monótonas de mínima longitud

J. M. Díaz-Báñez [*] G. Hernández [†] D. Oliveros [‡] A. Ramírez-Vigueras [§]
J. A. Sellarès [¶] J. Urrutia [‖] I. Ventura [**]

Abstract

Dados n puntos en el plano coloreados con k colores distintos, estudiamos cómo obtener una cadena poligonal de mínima longitud que visita los k colores. Este problema, conocido como *Trip Planning Problem*, es NP-duro por ser una generalización del problema del viajante, *TSP*. Imponemos dos restricciones sobre la poligonal que permiten resolver el problema polinomialmente, a saber, buscamos cadenas poligonales que sean monótonas con respecto a alguna dirección y que visiten los k colores en un orden de visita prefijado.

1 Introducción

Uno de los temas más estudiados en Geometría Computacional es el diseño eficiente de caminos más cortos. En muchos casos, el camino más corto debe visitar un conjunto de objetos [9]. Quizás el ejemplo más famoso sea el problema del viajante (*Traveling Salesman Problem*, **TSP**), donde la solución visita un conjunto de puntos dados en el plano. Es bien sabido que este problema pertenece a la clase de problemas NP-duros [10, 5]. Una variante de este problema es el problema **TPP** (*Trip Planning Problem*) [8] cuyo enunciado es: Dados n puntos en el plano, clasificados según distintas categorías, un punto de partida a y otro de destino b, encontrar el camino más corto que conecta a y b y pasa por al menos un punto de cada categoría. El problema *Trip Planning Problem* es también de naturaleza NP-duro por ser una generalización del **TSP**. Por otra parte, el problema de la búsqueda de subestructuras heterocromáticas minimales en grafos o gráficas ha sido estudiado con creciente interés en la última década en el campo de la Geometría Combinatoria [11, 7, 1]

El problema **TPP** está inspirado en diversas aplicaciones: *Localización-rutas* (una persona viaja de un lugar a otro de la ciudad y quiere parar en un supermercado, un banco, un puesto de periódicos y bar de tapas); *sistemas de navegación avanzada* (planificación de tareas en Google o mapas de navegación); o *redes de computación* (tenemos una red de computadores y un conjunto de tareas, de forma que cada tarea puede ser ejecutada solamente por un conjunto especificado de nodos de la red y queremos saber el camino más corto que visita un nodo de cada categoría).

En este artículo planteamos una versión simplificada del problema **TPP** imponiendo que la ruta sea *ordenada* y *monótona en alguna dirección*. Más adelante se definiran formalmente estas restricciones. La propiedad de monotonía ha sido ampliamente usada en problema de transportes y modelación

[*]Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain, dbanez@us.es. Subvencionado por el proyecto MTM2006-03909.

[†]Facultad de Informática, Universidad Politécnica de Madrid , Spain, gregorio@fi.upm.es. Subvencionado por el proyecto CAM P-DPI-000235-0505.

[‡]Instituto de Matemáticas, Universidad Nacional Autónoma de México, dolivero@matem.unam.mx.

[§]Posgrado en Ciencia e Ingenieria de la Computación, Universidad Nacional Autónoma de México, adriana.rv@gmail.com. Subvencionado por CONACYT de México

[¶]Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, sellares@ima.udg.es. Subvencionado por el proyecto TIN2004-08065-C02-02 del MEC.

[‖]Instituto de Matemáticas, Universidad Nacional, Autónoma de México, urrutia@matem.unam.mx. Subvencionado por los proyectos MTM2006-03909 y CONACYT de México, Proyecto 37540-A.

[**]Departamento de Matemáticas, Universidad de Huelva, Spain, iventura@us.es. Subvencionado por el proyecto MTM2006-03909.

porque simplifica la estructura a buscar [3] (volver hacia atrás es, de alguna manera, más costoso) y, por otro parte, la búsqueda de piezas monótonas nos da información de la 'monotonía' implícita en un conjunto de datos espaciales [12].

Si volvemos al problema del recorrido más corto que conecta dos puntos en una ciudad pasando por varios lugares específicos, podemos considerar a su vez, que el orden de visita esté fijado de antemano, de tal suerte que por ejemplo, estamos interesados en visitar primero el puesto de periódicos, después el banco, a continuación el supermercado y, finalmente ir a tomar una cerveza. Llamaremos *ordenada* a una poligonal que visita las distintas categorías según un orden prefijado de antemano.

Comenzaremos con el caso simple en el que se busca una cadena poligonal heterocromática de longitud mínima cuando la dirección de monotonía es un parámetro fijado a priori. Proponemos un algoritmo eficente para este caso que se usará después para resolver el problema general en el que la dirección de monotonía es también una variable a determinar.

La formalización de los problemas es como sigue. Sean n puntos en el plano coloreados según k colores distintos. Denotamos por n_i los puntos del color i, $i = 1, \ldots, k$ de tal forma que $n_1 + n_2 + \ldots + n_k = n$. En todo lo que sigue suponemos que nos dan el orden de visita, que denotaremos como $(1, 2, \ldots, k)$, renombrando los colores si fuera necesario. Dada una dirección determinada por un ángulo $\theta \in [0, \pi)$, diremos que una cadena poligonal C es monótona en esa dirección si toda recta de dirección $\theta + \frac{\pi}{2}$ intersecta a C en un conjunto conexo. Por abuso del lenguaje llamaramos dirección al angulo θ. Denotaremos como $C = (p_1, p_2, \ldots, p_k)$ una cadena poligonal de vértices $p_1, p_2, \ldots, p_k$ ordenada según el orden $(1, 2, \ldots, k)$. La longitud de C se define como $l(C) = \sum_{i=1}^{k} d(p_i, p_{i+1})$. En lo que sigue supondremos que $d(\cdot)$ denota la distancia Euclídea aunque el método que propondremos puede generalizarse a otras métricas. Los dos problemas a tratar en este trabajo son:

(**PMOO**) *Poligonal Monótona Orientada y Ordenada de Longitud Mínima: Dados n puntos en el plano clasificados según k colores distintos y una dirección $\theta \in [0, \pi)$, encontrar la poligonal monótona en la dirección θ de longitud mínima que contiene al menos un punto de cada color en un orden de visita prefijado $(1, 2, \ldots, k)$.*

(**PMO**) *Poligonal Monótona Ordenada de Longitud Mínima: Dados n puntos en el plano clasificados según k colores distintos, encontrar una dirección $\theta \in [0, \pi)$ de forma que la poligonal monótona más corta en la dirección θ sea la menor de entre todas las poligonales monotonas posibles que visitan al menos un punto de cada color en un orden de visita prefijado.*

Las condiciones de orden y monotonía permiten que haya casos en los que estos problemas no tengan solución. Por ejemplo, dado un conjunto de puntos como en la Figura 1, no existe una poligonal monótona en ninguna dirección que visita los cuatro colores en el orden $(1, 2, 3, 4)$.

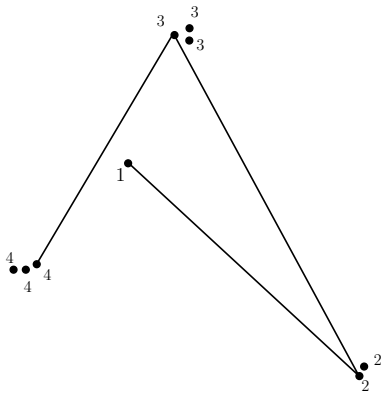

Figura 1: No existe una poligonal heterocromática en el orden $(1, 2, 3, 4)$ que sea monótona en ninguna dirección.

En las secciones que siguen resolvemos estos problemas usando un método combinado de programación dinámica y localización de puntos en Diagramas de Voronoi.

2 Poligonal Monótona Orientada Ordenada

Sin pérdida de generalidad, vamos a suponer en esta sección que la dirección de monotonía es la del eje OX, llamando x-monótona a una poligonal que sea monótona en esa dirección. De esta forma, suponemos que los puntos están ordenados según abscisa creciente y denotaremos por p_j^i al punto j-ésimo (según abcisa creciente) del color i. El algoritmo que proponemos está basado en las siguientes propiedades.

Lema 2.1. *Si $(p_j^1, p_l^2, \ldots, p_m^{k-1}, p_s^k)$ es la cadena poligonal heterocromática x-monótona ordenada de menor longitud según el orden de visita $(1, 2, \ldots, k-1, k)$, entonces p_s^k es el punto de color k más cercano a p_m^{k-1} de los situados en el semiplano derecho definido por una recta vertical que pasa por p_m^{k-1}.*

Este lema indica que el último vértice de la poligonal óptima es el punto más cercano al penúltimo vértice de ésta, pero esto no tiene porqué ocurrir para un vértice intermedio, véase como ejemplo la Figura 2.

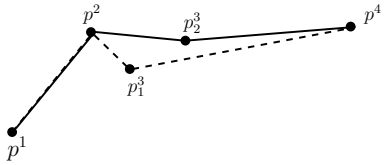

Figura 2: p_2^3 es un vértice de la poligonal óptima y no es el punto de color 3 más cercano a p^2.

Lema 2.2. *Si $(p_j^1, \ldots, p_l^{i-1}, p_r^i, \ldots, p_s^k)$ es la cadena poligonal x-monótona ordenada de menor longitud que visita k colores según el orden de visita $(1, 2, \ldots, k-1, k)$, entonces la poligonal x-monótona ordenada de menor longitud que visita $i \leq k$ colores y termina en p_r^i es precisamente $(p_j^1, \ldots, p_l^{i-1}, p_r^i)$.*

Demostración. Supongamos que existe otra poligonal, $(q_j^1, \ldots, q_l^{i-1}, p_r^i)$, que visita i colores, termina en p_r^i y es de menor longitud que $(p_j^1, \ldots, p_l^{i-1}, p_r^i)$. Entonces, la longitud de la poligonal heterocromática $(q_j^1, \ldots, q_l^{i-1}, p_r^i, \ldots, p_s^k)$ es menor que la de $(p_j^1, \ldots, p_l^{i-1}, p_r^i, \ldots, p_s^k)$, lo que contradice la optimalidad de la cadena $(p_j^1, \ldots, p_l^{i-1}, p_r^i, \ldots, p_s^k)$. □

Las propiedades anteriores sugieren un método de programación dinámica. Nuestro método, como veremos a continuación, se basa en la combinación de programación dinámica y localización de puntos en diagramas de Voronoi aditivamente ponderados.

Sean $\{p_1^i, \ldots, p_{n_i}^i\}$ los puntos de color i, $i = 1, \ldots, k$ ordenados según abscisa creciente. El método que proponemos consta de $k-1$ etapas que denotaremos por $ETAPA(i-1, i)$, $i = 2, \ldots, k$. En cada una de ellas, sólo se consideran los puntos de los colores $i-1$ e i. A continuación, describimos el algoritmo con detalle en tres etapas, una inicial, otra genérica y una de salida de la solución.

- $ETAPA(1, 2)$:

  Si todos los puntos de color 2 están a la izquierda de todos los de color 1, el algoritmo finalizaría aquí pues el problema no tiene solución. En otro caso, realizamos un barrido con una recta vertical que visita de izquierda a derecha los puntos de color 2, calculando en cada parada el punto más cercano de color 1 situado a su izquierda, esto es, en el semiplano izquierdo definido por una recta vertical que pasa por él. Esto se realiza de la forma siguiente:

  Si p_1^2 es el primer punto de color 2, calculamos el diagrama de Voronoi ordinario de los puntos de color 1 a la izquierda de p_1^2 y construimos la estructura de datos necesaria para localizar eficientemente p_1^2 en esta estructura. Si p_s^1 el punto más cercano a p_1^2 (ver Figura 3), asociamos a p_1^2 la etiqueta l_{s1}^{12}, que indica la longitud de la poligonal que tiene como vértices a p_s^1 y p_1^2. Hacemos notar aquí que l_{s1}^{12} es la longitud mínima de una cadena que visita los colores 1 y 2 y acaba en p_1^2.

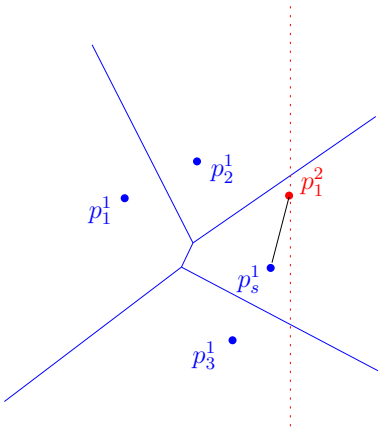

Figura 3: Localización del punto de color 1 más cercano a p_1^2.

Para los siguientes puntos de color 2 que aparecen en el barrido (paradas del barrido), p_j^2, $j = 2, \ldots, n_2$, hacemos lo siguiente:

(a) Si no hay nuevos puntos del color 1 a la izquierda de p_j^2, (Figura 4.a), entonces se obtiene el punto más cercano a p_j^2 usando el diagrama de Voronoi ya construido asociando a p_j^2 la correspondiente etiqueta de longitud.

(b) Si hay nuevos puntos del color 1 a la izquierda de p_j^2 (Figura 4.b) utilizamos el algoritmo de barrido de Fortune [4] para calcular el diagrama de Voronoi dinámicamente y actualizamos dinámicamente la estructura de datos para responder la pregunta del punto más cercano a p_j^2 (p_r^1 en la Figura 4.b). Usamos para ello las estructuras de [6]. A p_j^2 se le asocia la etiqueta l_{rj}^{12} que indica la longitud de la poligonal mínima que visita los colores 1 y 2 y acaba en el punto p_j^2.

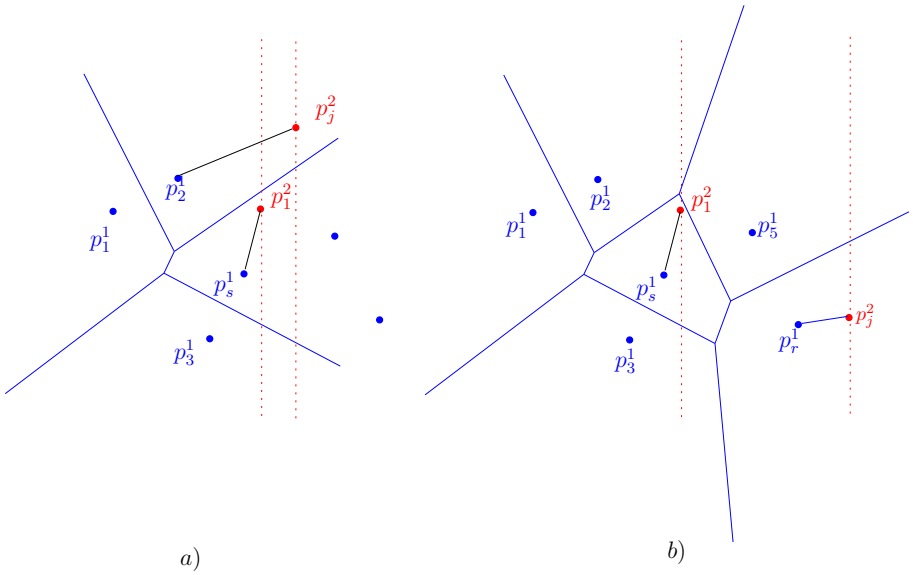

Figura 4: La dirección de monotonía está fijada.

Hacemos notar que en la etapa inicial se eliminan todos los puntos de color 2 con abcisa menor a la abcisa de todos los de color 1. De esta forma, al finalizar el barrido de la lista de color 2, cada punto tendrá asociado un peso y si elegimos el menor peso tendremos la cadena bicromática óptima para el conjunto total de puntos de estos colores.

- $ETAPA(i-1,i)$, $i=3,\ldots,k$:

  Para cada color $i, i=3,\ldots,k$ realizamos lo siguiente:

  (a) Si todos los puntos de color i están a la izquierda de todos los de color $i-1$, el problema no tiene solución.

  (b) En otro caso, en esta etapa ya disponemos de una lista de los puntos de color $(i-1)$, de sus correspondientes pesos (que corresponden a la mínima longitud de una cadena que visita $i-1$ colores y acaba en ese punto) y de la lista ordenada según abcisas de los puntos de color i. Sea p_j^i el j-ésimo punto de color i. Calculamos dinámicamente el diagrama de Voronoi de pesos aditivos de los puntos de color $(i-1)$ (aplicando [4]) que tienen abscisa menor que la de p_j^i y actualizamos la estructura de datos para responder a localización de puntos usando [2]. Si p_s^{i-1} es el generador de la región donde se ubica p_j^i, asociamos a p_j^i la etiqueta $l_{r\ldots sj}^{1\ldots(i-1)i}$, que contiene la suma del peso del punto p_s^{i-1} y la distancia entre p_j^i y p_s^{i-1}, es decir, guardamos la longitud de la poligonal más corta que visita i colores y acaba en p_j^i. Notemos que los puntos de color i que están está a la izquierda de todos los de color $i-1$ serían eliminados en un preproceso.

- $SALIDA$: Al finalizar el algoritmo obtenemos a lo sumo n_k etiquetas (este caso se da cuando no existe solución) y nos quedamos con la que tenga menor peso. La solución del problema es la poligonal que pasa por los puntos señalados en la etiqueta y su longitud es el peso de ésta.

Análisis de Complejidad:

Cada etapa requiere fundamentalmente de dos tareas que se procesan dinámicamente durante el barrido: calcular dinámicamente el Diagrama de Voronoi y construir la estructura de datos necesaria para responder la localización de puntos.

En la $ETAPA(1,2)$ calculamos el diagrama de Voronoi ordinario de los puntos de color 1 en tiempo $O(n_1 \log n_1)$ [4]. Para realizar la localización dinámica de puntos, usamos el método propuesto en [6], donde se describe un esquema para construir incrementalmente una subdivisión monótona del plano, donde la partición del plano es un conjunto de polígonos monótonos con respecto al eje Y, es decir que cada cara de dichos polígonos es intersectada a lo más una vez con una línea horizontal en este artículo se mantiene una estructura de datos formada por dos árboles de expansión, los cuales son construidos a partir de los puntos y su grafo o gráfica plana dual, para responder de forma eficiente preguntas de localización de puntos en dicha subdivisión, haciendo una *descomposición centroide* del árbol dual.

Usando las estructuras de [6] respondemos las cuestiones de localización de puntos en tiempo $O(\log n_1 \log\log n_1)$ y se requiere tiempo $O(1)$ amortizado para realizar inserciones de vértices y aristas (insertar una poligonal con k vértices requiere $O(k)$ tiempo amortizado). El espacio utilizado por esta estructura de datos es $O(n)$.

En la $ETAPA(i-1,i)$ $i=3,\ldots,k$, calculamos el diagrama de Voronoi de pesos aditivos basandonos en una transformación geométrica, tal como se menciona en [4] y se realiza en tiempo $O(n_{i-1}\log n_{i-1})$

Para la actualización dinámica de la estructura de datos que se refiere para localización de puntos usamos ahora [2], puesto que el diagrama de Voronoi ponderado no es en general una subdivisión monótona del plano. De esta forma para cada punto de color i se obtiene el de color $i-1$ más cercano, situado a su izquierda, en tiempo $log^2(n_{i-1})$, con un espacio lineal y la inserción de vértices dentro de la estructura es en $O(log\ n)$

Por el análisis anterior, el tiempo total que requiere este método es, $O(n_1\log n_1 + n_2 \log n_1 \log\log n_1) + O(n_2\log n_2 + n_3\log^2 n_2) + \ldots + O(n_{k-1}\log n_{k-1} + n_k \log^2 n_{k-1}) \leq O(n\log^2 n)$.

Debido a que las estructuras de datos usadas reunen espacio lineal. Podemos enunciar el siguiente resultado:

Theorem 2.3. *El problema* ***PMO*** *cuando la dirección de monotonía está fijada, se puede resolver en tiempo* $O(n\log^2 n)$ *y espacio* $O(n)$.

3 Poligonal monótona no orientada ordenada

En esta sección abordamos el problema general no orientado, esto es, buscamos la mejor orientación posible que permite construir la cadena poligonal monótona más corta (en esa dirección).

A continuación veremos cómo reducir este problema a un número cuadrático de problemas tipo orientado.

Consideremos una orientación fija $\theta \in [0, \pi)$. Denotamos como $l(\theta)$ la recta que pasa por el origen de coordenadas O y orientación θ y mediante $proy_\theta(p)$ a la proyección ortogonal del punto p sobre la recta $l(\theta)$.

Podemos considerar una caracterización equivalente de monotonía: la cadena poligonal C formada por $p_1, p_2, \ldots, p_n$ es monótona respecto a la dirección de $l(\theta)$ si los puntos $proy_\theta(p_1)$; $proy_\theta(p_2)$; $\ldots$; $proy_\theta(p_n)$ aparecen en $l(\theta)$ en el mismo orden que en C.

Si rotamos la recta $l(\theta)$ alrededor del origen O con θ de 0 a π, la ordenación de las proyecciones $proy_\theta(p_1)$; $proy_\theta(p_2)$; $\ldots$; $proy_\theta(p_n)$ cambia en unos determinados instantes que denominamos *eventos*. En efecto, cuando la recta que pasa por cualesquiera dos puntos p_i y p_j de C es ortogonal a $l(\theta)$, el orden de los puntos $proy_\theta(p_i)$; $proy_\theta(p_j)$ se invierte, véase la Figura 5.

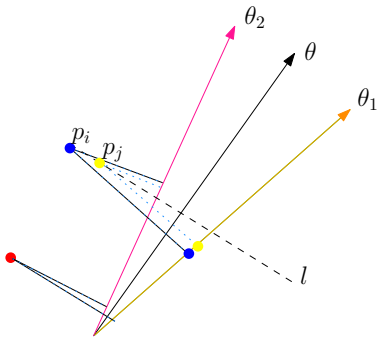

Figura 5: El orden de las proyecciones de los puntos p_i y p_j se invierte.

Analizando el proceso de rotación obtenemos un total de $O(n^2)$ eventos que corresponden a distintas ordenaciones de las proyecciones. Diremos que dos orientaciones θ y θ' son *equivalentes* cuando los órdenes de los puntos $proy_\theta(p_1)$; $proy_\theta(p_2)$; $\ldots$; $proy_\theta(p_n)$ y $proy_{\theta'}(p1)$; $proy_{\theta'}(p_2)$; $\ldots$; $proy_{\theta'}(p_n)$ coinciden. Por tanto, C es monótona respecto a todas las direcciones dadas por orientaciones de una misma clase o no lo es respecto a ninguna.

Para cada una de las $O(n^2)$ clases de equivalencia elegimos una orientación y aplicamos el algoritmo de la sección anterior para buscar una cadena poligonal monótona en esa dirección. Notamos aquí que esta solución no cambia para cualquier otra dirección de la misma clase de equivalencia. Si nos permitimos usar esta técnica para cada clase de equivalencia, obtenemos un algoritmo simple de tiempo $O(n^3 \log^2 n)$. Y para elegir una orientación en cada clase de equivalencia hace falta ordenar las $O(n^2)$ orientaciones que definen las clases. Una vez ordenadas, dos orientaciones consecutivas determinan una clase y es fácil determinar un representante. Por tanto hay un preproceso que requiere tiempo $O(n^2 \log n)$ y espacio $O(n^2)$.
En consecuencia, tenemos el siguiente resultado:

Teorema 3.1. *El problema **PMO** cuando la dirección de monotonía no está fijada, se puede resolver en tiempo $O(n^3 \log^2 n)$ y espacio $O(n^2)$.*

4 Comentarios

En este trabajo consideramos un caso particular del problema *Trip Planning Problem*, donde el orden de visita de las distintas categorías a cubrir está determinado a priori y el camino poligonal buscado

ha de ser monótono en alguna dirección. Se han distinguido dos casos fijando o no la dirección de monotonía para la cadena poligonal. Y si se decide agregar la restricción como el **TPP** de tener un punto de partida a y otro de destino b se tiene el mismo resultado.

Incluimos aquí posibles líneas de seguimiento que ya estamos abordando. En primer lugar, podemos eliminar una de las restricciones aquí consideradas. De esta forma, tendríamos el problema de encontrar un camino poligonal mínimo heterocromático monótono (no ordenado) o un camino poligonal mínimo heterocromático ordenado (no necesariamente monótono). De hecho, para el segundo caso, hemos visto en la Figura 1 que puede no existir una cadena monótona heterocromática para un orden prefijado.

En segundo lugar, existen otras variantes posibles como las siguientes. Consideramos la dirección de monotonía fijada y que cada segmento de la cadena se encuentre en un rango de ángulos dados (conos). En el caso que hemos resuelto en la Sección 2, el ángulo es de 180 grados. Por otro lado, para los casos que ya sabemos resolver (con direcion fijada y sin fijar), puede que no exista solución. Entonces se trataría de buscar el camino mínimo que "vuelve hacia atras" el mínimo número de veces.

Agradecimientos

Los problemas aquí tratados fueron propuestos y resueltos parcialmente durante el *Primer Taller Iberoamericano de Geometría Combinatoria y Computacional*, celebrado en el Centro de Investigación en Matemáticas (CIMAT) de Guanajuato, México los días 11–15 de Diciembre de 2006. Los autores quieren agradecer a los organizadores del evento y a los demás participantes sus comentarios y apoyo.

References

[1] R. A. Brualdi, S. Hollingsworth. Multicolored forests in complete bipartite graphs. *Discrete Math.* 240, 239-245, 2001

[2] S.W. Cheng and R. Janardan. New results on dynamic planar point location. *IEEE*, 1990.

[3] Díaz-Báñez, J.M., Gómez F. and Hurtado, F., Approximation of Point Sets by 1-Corner Polygonal Chains. *INFORMS Journal on Computing*, 12, (2000), 317-323.

[4] S.J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[5] M. R. Garey, D. S. Johnson. Computers and Intractability: A guide to the theory of NP-completeness. *Freeman*. 1979.

[6] M.T. Goodrich and R. Tamassia. Dynamic trees and dynamic point location. *SIAM J. COMPUT.*, Vol. 28, No. 2, pp. 612–636, 1998.

[7] A. Kaneko, M. Kano, K. Suzuki. Three edge-disjoint Multicolored Spanning Trees in Complete Graphs. *preprint*, 2002.

[8] F. Li and D. cheng. On trip Planning Queries in Spatial Databases. *BUCS TR 2004-027*, Boston university, MA, USA.

[9] J.S.B. Mitchell Shortest paths and networks. In Goodman, J.E., O'Rourke, J. eds. Handbook of Discrete and Computational Geometry, CRC Press LLC (1997), 445-466.

[10] C. H. Papadimitriou. The euclidean traveling salesman problem is NP-complete. *Theor. Comput. Sci.* 4. pp. 237-244, 1977.

[11] K. Suzuki. A Necessary and Sufficient Condition for the Existence of a Heterochromatic Spanning Tree in a Graph. *Graphs and Combinatorics*, Volume 22, Number 2, pp. 261-269, June 2006

[12] Y. Yan, D. Lemire and M. Brooks Monotone Pieces Analysis for Qualitative Modeling. *Proceedings ECAI 2004 MONET Workshop on Model-Based System*, Valencia, Spain, 2004.