Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Un algoritmo de aproximación para encontrar un árbol de Steiner Mínimo en aristas al interior de un polígono simple

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Lic. en Matemáticas Aplicadas y Computación

PRESENTA:

Hernán Tellechea Garduño

DIRECTOR DE TESIS:

Sebastián Bejos Mendoza

Marzo, 2016

Santa Cruz Acatlan, Naucalpan, Estado de México

Agradecimientos

A mis padres: Elva Garduño y Gildardo Tellechea. Su apoyo y enseñanzas me han acompañado a lo largo de mi vida. Éste logro académico no hubiera sido posible sin ustedes.

A mis compañeros: Sinuhé Jaime, Ismael Rosalino, Karla Sánchez y Angélica Zavala. Siempre es importante tener con quién compartir experiencias dentro y fuera del aula. Pasamos muchos buenos ratos. Sufrimos en otros. Ustedes me ayudaron a mantener el balance.

Fuera de la escuela siempre tuve a: Emilio Fonseca, Edmundo Pérez, Oscar Almaraz y Alejandro Zarzosa. Sus valiosos consejos sobre la vida me ayudaron a superar muchos de los obstáculos que se me presentaron. Los momentos de esparcimiento fueron muchos. Siempre los recordaré con gusto.

Quiero hacer un agradecimiento especial a Víctor Muñoz, Alejandro Zarzosa y Daniel Cordero por su aportación a este trabajo tanto en ilustraciones, redacción y ayuda con latex.

A mi asesor Sebastián Bejos, por la oportunidad de trabajar en este tema, por su paciencia, comentarios y por todo lo que he aprendido gracias a él.

Al Laboratorio de Algoritmos para la Robótica de la Facultad de Estudios Superiores Acatlán por brindarme el espacio y material para trabajar en esta tesis.

Índice general

Agradecimientos II

Índice III

Abreviaciones V

Introducción VI

1. Algoritmos de visibilidad en el plano **1**
1.1. Visibilidad . . . 1
1.1.1. Noción de visibilidad . . . 1
1.1.2. Polígonos . . . 3
1.1.3. Visibilidad de un punto en un polígono simple . . . 3
1.2. Triangulación . . . 5
1.2.1. El problema de la galería de arte . . . 7
1.3. Tipos de visibilidad . . . 9
1.3.1. Visibilidad débil . . . 11
1.4. Trayectoria mínima en aristas al interior de polígonos simples . . . 16
1.4.1. Trayectorias mínimas en aristas . . . 16
1.4.2. Partición por ventanas de un polígono simple . . . 21

2. Algoritmo exponencial exacto para encontrar el árbol de Steiner mínimo en aristas dentro de un polígono simple **26**
2.1. Algoritmo exacto para encontrar SMT(Z,P) . . . 26
2.2. Análisis de tiempo y espacio . . . 32
2.3. Ejemplo de ejecución del algoritmo . . . 35

3. Algoritmo de aproximación para encontrar un árbol de Steiner mínimo en gráficas **50**
3.1. Árbol de expansión mínima en hipergráficas . . . 51
3.2. Razones de Steiner . . . 52
3.2.1. Cota inferior para p_r . . . 52
3.2.2. Cota Superior para p_r . . . 60
3.3. Algoritmo de aproximación para el problema del árbol de expansión mínima en hipergráficas . . . 68

4. Algoritmo de aproximación para encontrar un árbol de Steiner mínimo en aristas en polígonos simples **77**
4.1. Construcción del algoritmo 78
4.1.1. Gráfica de distancia $H_2(Z, P)$ 78
4.1.2. Árbol de Steiner r-acotado $H_r(Z, P)$ 79
4.1.3. Métrica 80
4.2. Algoritmo de aproximación para $SMT(Z, P)$ 80
4.3. Ejecución del algoritmo de proxumación 82

Conclusiones y trabajo futuro **102**

Bibliografía **104**

Abreviaciones

$V(p)$	Visibilidad de un punto p al interior de un polígono.
$V(pq)$	Visibilidad de un segmento de recta pq al interior de un polígono.
$T(P)$	Triangulación de un polígono P.
$G(T(P))$	Gráfica dual de la triangulación de P.
DFS	(**D**epth **F**irst **S**earch) Búsqueda en profundidad.
$ST(s,t)$	(**S**hortest **P**ath) Trayectoria euclidiana más corta entre los puntos s y t dentro de un polígono.
$SPT(s)$	(**S**hortest **P**ath **T**ree) Árbol de trayectorias más corta que conecta a s con todos los vértices del polígono donde está contenido s.
$bd(P)$	(**b**or**d**er) Borde o límite de P.
$MLP(s,t)$	(**M**inimum **L**ink **P**ath) Trayectoria mínima en aristas entre los puntos s y t al interior de un polígono.
$MLD(s,t)$	(**M**inimum **L**ink **D**istance) Número de aristas en $MLP(s,t)$.
$P[c;x]$	Partition por ventanas desde c, donde c es uno de los dos puntos finales del segmento de recta x.
$WT(x)$	(**W**indows **T**ree) Árbol de ventanas generado a partir del punto x.
$SMT(Z,P)$	(**S**teiner **M**inimum **T**ree) Árbol de Steiner mínimo en aristas al interior de un polígono simple P a partir del conjunto de terminales Z.
$smt(Z,P)$	Número de aristas en $SMT(Z,P)$.
$\binom{C}{n}$	Todos los subconjuntos de tamaño n del conjunto C.
$H_r(N)$	Hipergráfica r-acotada ponderada.
MST	(**M**inimum **S**panning **T**ree) Árbol de expansión mínima.
mst	Número de aristas en MST.

Introducción

El problema del árbol de Steiner fue nombrado así en honor a Jakob Steiner (1796-1863), quien resolvió y popularizó el problema de unir tres villas por un sistema de caminos que tuvieran la mínima longitud total. El problema, en su versión Euclidiana, consiste en unir un conjunto de puntos que están en el plano a través de segmentos de recta tal que la suma de las longitudes de los segmentos sea el mínimo. Para esto, es posible crear puntos auxiliares que ayudan a minimizar la longitud total, estos puntos son llamados puntos de Steiner (o vértices de Steiner).

El problema del árbol de Steiner en gráficas consiste en, dada una gráfica (conjunto de vértices y aristas) y un conjunto de vértices (conjunto de terminales) contenidos en la gráfica ya mencionada, encontrar el árbol mínimo en aristas que conecte al conjunto de terminales, esto es, encontrar un árbol que conecte al conjunto de terminales y tenga la cantidad mínima de aristas necesaria para ello. Nótese que en este problema no es posible agregar vértices auxiliares, lo que se hace en este caso es utilizar vértices que ya pertenecen a la gráfica.

Calcular trayectorias óptimas bajo algún criterio, en un dominio geométrico, es un problema fundamental de la geometría computacional. Por ejemplo, un problema muy conocido en esta área es calcular la trayectoria con la menor distancia Euclidiana entre dos puntos en un polígono simple. Otro problema interesante y muy utilizado en este trabajo es encontrar la trayectoria mínima en número de segmentos de recta entre dos puntos dentro de un polígono simple.

En este trabajo se continúa el estudio de una generalización en la distancia mínima en segmentos de recta entre dos puntos en un polígono simple. En lugar de buscar

conectar únicamente dos puntos, ahora se desea conectar m puntos al interior de un polígono simple, de igual manera, minimizando el número de segmentos de recta a utilizar. Se busca entonces encontrar un árbol mínimo en aristas que conecte a los m puntos y esté contenido totalmente en el polígono simple. Esta generalización ha sido estudiada en [1] y [2] y sus principales resultados son:

- El problema de encontrar un árbol de Steiner mínimo en aristas al interior de un polígono simple es un problema $NP - Completo$.
- Se encontró una cota combinatoria que relaciona el número de vértices de Steiner y el número de vértices de un polígono simple.
- Un par de algoritmos exponenciales para calcular un árbol de Steiner mínimo en aristas al interior de un polígono simple.
- Un algoritmo de aproximación para este problema con una razón de aproximación de 2.

En el capítulo uno veremos los algoritmos fundamentales para el desarrollo de la teoría del resto del trabajo. En el capítulo dos se muestra un algoritmo exponencial para calcular el árbol de Steiner mínimo en aristas dado un polígono P y un conjunto de terminales Z dentro de P plateado en [1] pero con modificaciones importantes y se realiza su análisis de tiempo y espacio. En el capítulo tres se habla de un algoritmo 1.69-aproximación para el problema del árbol de Steiner mínimo en gráficas. En el capítulo cuatro se describe un algoritmo de 1.69-aproximación para el problema del árbol de Steiner mínimo en aristas al interior de un polígono simple.

En el capítulo dos se agregó la subrutina $BFS - Ajuste$ al algoritmo exponencial exacto [1, pág. 46], la cual sirve para corregir la funcionalidad del algoritmo mencionado para algunos casos de entrada específicos. Lo que hace esta subrutina es: cada vez que se agrega un nuevo vértice terminal a nuestro árbol solución, se hace una busqueda en amplitud sobre el árbol para ajustar el tamaño de los polígonos que representan a cada vértice de Steiner, ya que, para poder conectar un vértice

terminal a un vértice de Steiner, siempre será necesario hacer una intersección de polígonos; la cual puede reducir el tamaño del polígono de visión del vértice de Steiner. Si se reduce el tamaño de un polígono de visión, es posible que los polígonos de visión de otros vértices de Steiner conectados al vértice de Steiner mencionado tambien reduzcan su tamaño, por lo que es necesaraio revisar su vecindad de vértices de Steiner para verificar si es necesaria la reducción de área de visibilidad.

Capítulo 1

Algoritmos de visibilidad en el plano

El objetivo de este capítulo es familiarizar al lector con los conceptos que se manejan a lo largo de este trabajo. Se mostrará información necesaria que respalda conceptos de los siguientes capítulos.

1.1. Visibilidad

1.1.1. Noción de visibilidad

La visibilidad es un fenómeno natural de la vida diaria. Vemos objetos a nuestro alrededor y entonces decidimos nuestros movimientos de acuerdo con esto. Ver un objeto significa identificar porciones del objeto que son visibles desde la posición actual de un observador. Es posible que el objeto entero no sea visible desde la posición de un observador. El observador también puede identificar formas y tamaños de porciones de un objeto. Las porciones visibles de un objeto pueden cambiar de forma si el observador se mueve de un lugar a otro. Por otra parte, el observador puede ver varios objetos en diferentes direcciones desde su posición

actual; las porciones visibles de esos objetos forman la escena alrededor del observador. Construir tal escena es muy común para un observador humano ya que su sistema visual puede ejecutar esta tarea sin esfuerzo.

Supóngase que un robot quiere llegar de un punto inicial a una posición objetivo sin colisionar con un objeto o algún obstáculo alrededor de él. El robot construye la escena alrededor de él desde la posición actual y entonces guía su movimiento en el espacio libre entre él y los objetos alrededor de él. Las posiciones de los objetos y el robot pueden ser representadas por la computadora del robot como coordenadas x, y, y z, por lo tanto, la escena que consiste en porciones visibles de esos objetos pueden ser calculadas para la posición actual del robot. Incluso calcular la visibilidad de un solo objeto con la presencia de otros objetos no es una tarea sencilla. Por otra parte, calcular una escena por cada punto en el camino del robot es una tarea que consume bastante tiempo incluso para una computadora de alta velocidad de procesamiento.

El problema anteriormente mencionado puede ser reducido a su problema correspondiente en dos dimensiones. Si un robot que tiene contacto constante con el suelo es proyectado en el suelo, su sombra bidimensional puede modelarse como un polígono. Estas mismas proyecciones pueden ser hechas para los obstáculos. Este proceso resulta ser un mapa que consiste de polígonos en dos dimensiones. El polígono del robot puede ser utilizado para navegar este mapa sin tocar los obstáculos poligonales. Entonces se puede calcular una ruta libre de colisiones desde la posición original a la posición objetivo. Mientras se navega, se calcula la visibilidad de porciones de los obstáculos para construir la escena alrededor de la posición actual del robot. Aunque la representación bidimensional ha reducido la complejidad del problema de planear la ruta del robot, diseñar un algoritmo eficiente sigue siendo una tarea retadora.

1.1.2. Polígonos

Un polígono P se define como una región cerrada R en el plano acotado por un conjunto finito de segmentos de linea (llamadas aristas de P) tal que existe un camino entre cualesquiera dos puntos dentro de R los cuales no intersecan ninguna arista de P. Todo punto final de una arista de P es llamado vértice de P, el cual es un punto en el plano. Ya que P es una región cerrada y acotada, la frontera de P consiste en ciclos de aristas de P donde dos aristas consecutivas comparten un vértice. La región R es llamada la región interna de P. Similarmente, la región del plano que excluye todos los puntos de R es llamado región externa o exterior de P. Un vértice de P es llamado convexo si el ángulo interior del vértice formado por dos aristas es a lo más π; en caso contrario, es llamado reflejo.

Como se definió arriba, un polígono simple P es una región en el plano acotado por un ciclo de aristas tal que cualquier par no consecutivo de aristas no intersecan. En este trabajo se considera a P como una lista de vértices. Cada vértice tiene los campos x y y, que son las coordenadas de vértice. También se supone que el orden de los vértices está dado en contra de las manecillas del reloj. De esta forma, si se recorre la lista de vértices que definen a P, sabemos que el interior del polígono siempre está del lado izquierdo.

1.1.3. Visibilidad de un punto en un polígono simple

Dos puntos p y q en P se dicen visibles si el segmento de recta que los une no contiene ningún punto exterior de P. Esto significa que el segmento de recta $\overline{pq}$ está contenido totalmente dentro de P. Esta definición permite que el segmento de recta pase sobre un vértice reflejo o permite rozar una arista de P (ver figua 1.1).

Determinar la región de visibilidad de un objeto geométrico dentro de un polígono es un problema muy conocido en la geometría computacional. El polígono de visibilidad $V(q)$ de un punto q en un polígono simple P es el conjunto de todos los puntos de P que son visibles desde q. Por definición, cualquier $V(q)$ es un polígono

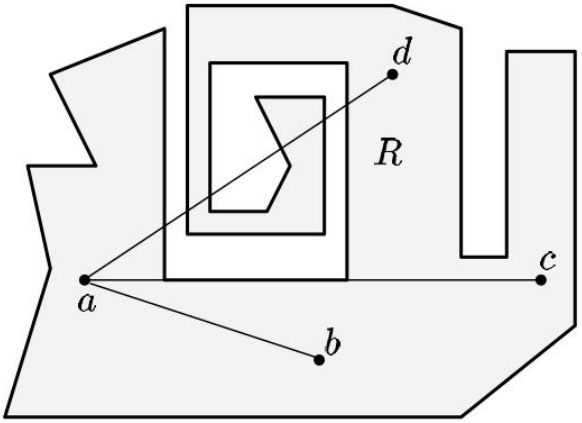


FIGURA 1.1: En este polígono simple, b y c son visibles desde a, pero d no lo es.

en forma de estrella. Un polígono P tiene forma de estrella si existe un punto z tal que para cualquier punto p dentro de P el segmento $\overline{zp}$ está totalmente contenido dentro de P. El conjunto de todos los puntos con esta propiedad es llamado ***kernel***.

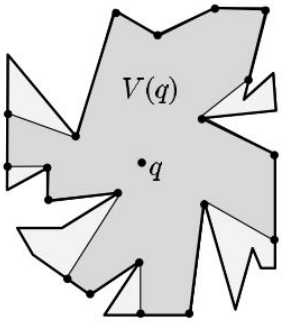


FIGURA 1.2: Polígono de visibilidad del punto q.

El polígono de visibilidad $V(q)$ de un punto q en un polígono simple fue primeramente resuelto por Davis y Benedikt [3] con un algoritmo que tarda $O(n)$. Después ElGindy y Davis [4] y Lee en [5] presentaron un algoritmo lineal que calcula $V(q)$.

Lema 1.1. *El polígono de visibilidad $V(q)$ de un punto q en un polígono simple puede ser calculado en tiempo $O(n)$.*

1.2. Triangulación

Una triangulación $T(P)$ de un polígono P es una partición de P en triángulos hecho por diagonales (ver figura 1.3), donde un segmento de recta que junta dos vértices de P mutuamente visibles es llamado *diagonal* de P. Se puede ver que una triangulación de P no es única ya que muchos subconjuntos de diagonales pueden dar una triangulación del mismo polígono. El problema de calcular el polígono de visibilidad $V(q)$ de un punto q está relacionado con el problema de eliminar lineas escondidas [6] y esto es parte del proceso de *rendereo* en gráficas computacionales.

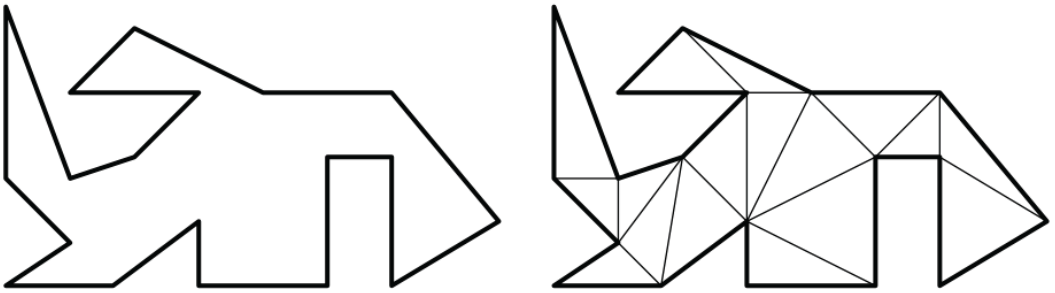


FIGURA 1.3: Triangulación de un polígono simple.

Lema 1.2. *Todo polígono simple puede ser triangulado y cada triangulación de un polígono simple con n vértices consiste en exactamente $n-2$ triángulos.*

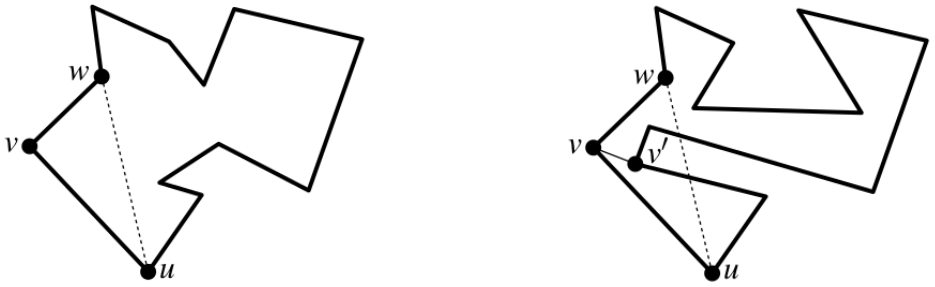
Demostración

Probaremos este teorema por inducción sobre n. Cuando $n = 3$ el mismo polígono es un triángulo, lo que hace que el teorema sea trivialmente cierto. Sea $n > 3$, y suponemos que el teorema es cierto para toda $m < n$. Sea P un polígono con n vértices. Primero probamos la existencia de una diagonal en P. Sea v el vértice más a la izquierda de P (en caso de empate, tomamos el de más abajo). Sean u y w los dos vecinos de v en la frontera de P. Si el segmento $\overline{uw}$ está totalmente contenido en P, encontramos una diagonal. Si no, hay uno o más vértices dentro del triangulo definido por u, v y w o están sobre la diagonal $\overline{uw}$. De los vértices que están a la izquierda de u y w, sea v' el vértice más lejano del segmento $\overline{uw}$. El segmento que conecta v y v' no puede intersectar una arista de P, ya que si existiera dicho vértice, uno de los vértices de esa arista estaría más a la izquierda

de $\overline{uw}$ que v', lo que contradice que v' sea el vértice más a la izquierda de $\overline{uw}$. Por lo tanto, $\overline{vv'}$ es una diagonal.

Así que las diagonales existen. Cualquier diagonal corta a P en dos subpolígonos simples P_1 y P_2. Sea m_1 el número de vértices de P_1 y m_2 el número de vértices de P_2. Ambos m_1 y m_2 tienen que ser menores a n. Así que, por inducción, P_1 y P_2 pueden ser triangulados. Por lo tanto, P puede ser triangulado.

Sólo queda probar que cualquier triangulación de P consiste en $n-2$ triángulos. Para esto, considérese una diagonal arbitraria en cualquier triangulación $T(P)$. Esta diagonal corta a P en dos subpolígonos con m_1 y m_2 vértices, respectivamente. Cada vértice de P aparece exactamente en uno de los subpolígonos, excepto por los vértices que definen la diagonal, quienes aparecen en ambos subpolígonos. Por lo tanto, $m_1+m_2 = n+2$. Por inducción, cualquier triangulación de P_i consiste en m_i-2 triángulos, lo que implica que $T(P)$ consiste en $(m_1-2)+(m_2-2) = n-2$ triángulos. □



(A) $\overline{uw}$ está totalmente contenida en P (B) $\overline{uw}$ no está totalmente contenida en P

FIGURA 1.4

1.2.1. El problema de la galería de arte

Eres dueño de una galería de arte y quieres colocar cámaras de tal manera que toda la galería sea a prueba de robos.

¿Donde pondrías las cámaras de tal manera que toda la galería sea visible?
¿Cuantas cámaras tendrías que comprar?

Para definir el problema de la galería de arte de forma más precisa, primero se debe formalizar la noción de galería. Una galería es un espacio tridimensional, pero el plano del piso nos da la información suficiente para saber donde poner las cámaras. Por lo tanto, se modela una galería como una región poligonal en el plano. Estas regiones están restringidas a ser polígonos simples. Las regiones con hoyos no son contempladas. La posición de una cámara corresponde a un punto dentro del polígono. Una cámara puede ver todos los puntos a los cuales se puede conectar con un segmento de recta que está totalmente contenido en el polígono. ¿Cuantas cámaras necesitas para cuidar toda la galería? Esto depende del polígono a tratar: entre más complejo sea el polígono, más cámaras serán requeridas. Se expresa la cota del número de cámaras necesarias en términos de n, el número de vértices del polígono. Pero incluso cuando dos polígonos tienen el mismo número de vértices, uno puede ser más difícil de vigilar que el otro.

El lema 1.2 implica que cualquier polígono con n vértices puede ser vigilado con $n-2$ cámaras, ya que cada triángulo puede ser vigilado por una cámara, pero esto puede ser una medida excesiva. Una cámara puesta en una diagonal, por ejemplo, puede vigilar dos triángulos, así que poniendo una cámara en una diagonal bien escogida reducimos el número de cámaras a $n/2$. Colocando cámaras en vértices parece ser incluso mejor, porque un vértice puede ser incidente a muchos triángulos, y una cámara en ese vértice los vigila a todos. Esto sugiere el siguiente enfoque: Sea $T(P)$ una triangulación de P. Se selecciona un subconjunto de vértices de P tal que cualquier triángulo en $T(P)$ tiene al menos un vértice seleccionado, y se

coloca una cámara en esos vértices. Para encontrar ese subconjunto asignamos un color a cada vértice de P: blanco, gris o negro. En una 3-coloración de un polígono triangulado, todos los triángulos tienen un vértice blanco, gris y negro. Por lo tanto, si colocamos las cámaras en, por ejemplo, los vértices de color gris, garantizamos la vigilancia del polígono completo. Si escogemos la clase de color más pequeña para poner las cámaras, podemos vigilar P usando a lo más $\lfloor \frac{n}{3} \rfloor$ cámaras.

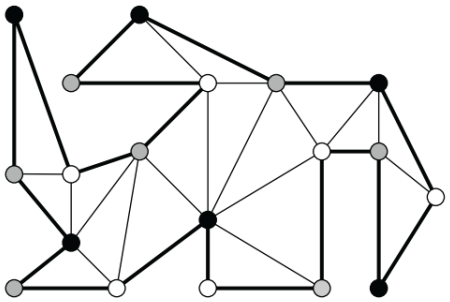
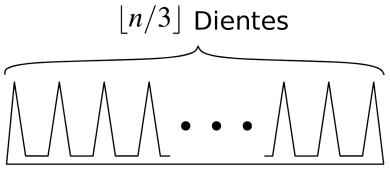


FIGURA 1.5: Una 3-coloración de una triangulación de un polígono simple.

¿Pero siempre hay una 3-coloración? La respuesta es sí. Para ver esto, se introduce la llamada gráfica dual de $T(P)$. Está gráfica $G(T(P))$ tiene un vértice por cada triángulo en $T(P)$. Se denota al triángulo correspondiente a un vértice v por $t(v)$. Hay una arista entre v y u si $t(v)$ y $t(u)$ comparten una diagonal. Una arista en $G(T(P))$ corresponde a una diagonal en $T(P)$. Ya que una diagonal parte P en dos, remover una arista de $G(T(P))$ parte la gráfica en dos. Por lo tanto, $G(T(P))$ es un árbol. Y, por lo tanto, la gráfica puede ser coloreada por un algoritmo simple de recorrido de gráficas, tal como una búsqueda en profundidad (DFS) [7, pág. 603].

Se concluye que un polígono simple triangulado siempre tiene una 3-coloración. Con base en esto, cualquier polígono simple puede ser vigilado con $\lfloor \frac{n}{3} \rfloor$ cámaras. Pero quizá se puede hacer algo mejor que eso. Después de todo, una cámara puesta en un vértice podría vigilar más que los triángulos incidentes. Desafortunadamente, para cualquier n existe un polígono simple que requiere $\lfloor \frac{n}{3} \rfloor$ cámaras. Un ejemplo es un polígono en forma de peine con una base horizontal larga de $\lfloor \frac{n}{3} \rfloor$ dientes

donde cada uno está hecho de dos aristas. Los dientes son conectados por aristas horizontales. La construcción del peine se hace de tal manera que una cámara no pueda ver dos dientes simultáneamente. Así que no podemos ver una estrategia que pueda producir menos de $\lfloor \frac{n}{3} \rfloor$ cámaras. En otras palabras, el enfoque de la 3-coloración es óptimo en el peor caso.
Acabamos de probar el teorema de la galería de arte, un resultado clásico de la geometría combinatoria.



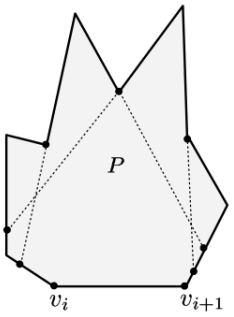
Teorema 1.3. *(Teorema de la galería de arte) Para todo polígono simple con n vértices, $\lfloor \frac{n}{3} \rfloor$ cámaras son ocasionalmente necesarias y siempre suficientes para tener cada punto del polígono siempre visible desde al menos una de las cámaras.*

1.3. Tipos de visibilidad

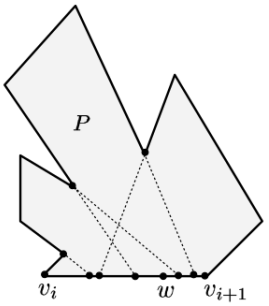
La noción de visibilidad débil de un polígono desde un segmento de recta fue introducido por Avis y Toussaint [8] en el contexto del problema de la galería de arte. Ellos consideraron la variación del problema cuando sólo puede haber un guardia y este tiene permitido moverse sobre una arista del polígono. Ellos definieron la visibilidad de una arista v_iv_{i+1} en un polígono simple P de tres maneras diferentes.

- Se dice que P es *completamente visible* desde v_iv_{i+1} si cada punto de $z \in P$ y cualquier punto $w \in v_iv_{i+1}$ son visibles (ver figura 1.6a).
- Se dice que P es *fuertemente visible* desde v_iv_{i+1} si existe un punto $w \in v_iv_{i+1}$ tal que para cada $z \in P$, w y z son visibles (ver figura 1.6b).

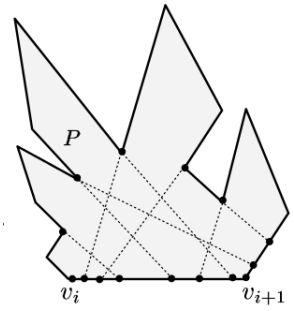
- Se dice que P es *débilmente visible* desde v_iv_{i+1} si para cada punto de $z \in P$ existe un punto $w \in v_iv_{i+1}$ (dependiendo de z), tal que w y z son visibles (ver figura 1.6c).



(A) Visibilidad completa



(B) Visibilidad fuerte



(C) Visibilidad débil

FIGURA 1.6

Volviendo al problema de la galería de arte, si P es completamente visible desde v_iv_{i+1}, un guardia puede posicionarse en cualquier punto de v_iv_{i+1}. En otras palabras, P y el polígono de visibilidad $V(w)$ de P desde cualquier punto $w \in v_iv_{i+1}$ son lo mismo. Si P es fuertemente visible desde v_iv_{i+1}, existe por lo menos un punto w en v_iv_{i+1} donde el guardia puede ver todo el polígono P, esto es, $P = V(w)$. Finalmente, si P es débilmente visible desde v_iv_{i+1}, es necesario que el guardia camine sobre todo el segmento v_iv_{i+1} para poder ver a todo P.

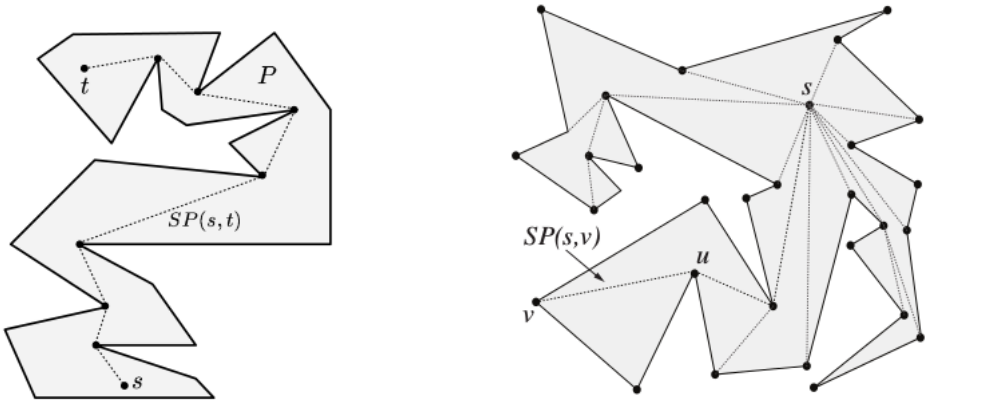
En este trabajo sólo se hablará del problema de calcular la visibilidad débil de un segmento de recta pq dentro de un polígono simple P.

1.3.1. Visibilidad débil

En esta sección se presentará el algoritmo de tiempo $O(n)$ de Guibas [9] para calcular el polígono de visibilidad débil $V(pq)$ de un polígono simple P de n vértices desde un segmento de recta pq al interior de P.

Antes de analizar este algoritmo, se hablará sobre la trayectoria euclidiana más corta entre dos puntos y el árbol de trayectorias euclidianas más cortas entre un punto y todo el polígono.

La trayectoria euclidiana más corta entre dos puntos s y t en P (denotada como $SP(s,t)$) es la trayectoria que conecta a s y t tal que esta trayectoria está totalmente contenida dentro de P y la longitud de la trayectoria es más corta que cualquier otra trayectoria que conecta a s y t (ver figura 1.7a). Así mismo, el árbol de trayectorias euclidianas más cortas de un punto s dentro de un polígono P, consiste en la unión de las trayectorias euclidianas más cortas entre el punto s y todos los vértices de P (ver figura 1.7b).



(A) trayectoria euclidiana más corta (B) Árbol de trayectorias euclidianas más cortas

FIGURA 1.7

Lema 1.4. *Sea P un polígono simple de n vértices y sean p y q dos puntos dentro de P. La trayectoria más corta $SP(p,q)$ puede ser calculada en tiempo $O(n)$ [9].*

Lema 1.5. *Sea P un polígono simple de n vértices y sea s un punto dentro de P. El árbol de trayectorias más cortas $SPT(s)$ puede ser calculado en tiempo $O(n)$ [10].*

Volviendo al tema de la visibilidad débil. El algoritmo que se presenta está dado para calcular el polígono de visibilidad débil en aristas convexas[1]. Pero a continuación se describe como, utilizando el mismo algoritmo, se puede calcular el polígono de visibilidad débil de una arista no convexa.

Se denota por $bd(P)$ a la frontera o límite de P y por $\overrightarrow{ab}$ al rayo[2] que comienza en el punto a y pasa por el punto b. Sea u el punto más cercano a p entre los puntos de intersección de $\overrightarrow{qp}$ y $bd(P)$ (Véase la figura 1.8) y sea v_iv_{i+1} la arista a la que u pertenece. De igual manera, sea w el punto más cercano a q entre los puntos de intersección de $\overrightarrow{pq}$ y $bd(P)$, y sea v_kv_{k+1} la arista a la que pertenece w. Ahora se divide a P en dos polígonos P_1 y P_2 con uw. Donde $P_1 = (v_i, u, p, q, w, v_{k+1}, \ldots, v_{i-1}, v_i)$ y $P_2 = (v_{i+1}, \ldots, v_k, w, q, p, u, v_{i+1})$. Es evidente que $V(pq)$ es la unión de los polígonos de visibilidad de P_1 y P_2 desde pq, y entonces el problema se reduce a calcular los polígonos de visibilidad débil de P_1 y P_2 desde pq (Calcular $V_{P_1}(pq)$ y $V_{P_2}(pq)$). Como pq se ha convertido en una arista convexa de ambos P_1 y P_2, el procedimiento para calcular el polígono de visibilidad débil desde pq para P_1 y P_2 , es el mismo que el que se presenta en el Algoritmo 1.3.1.

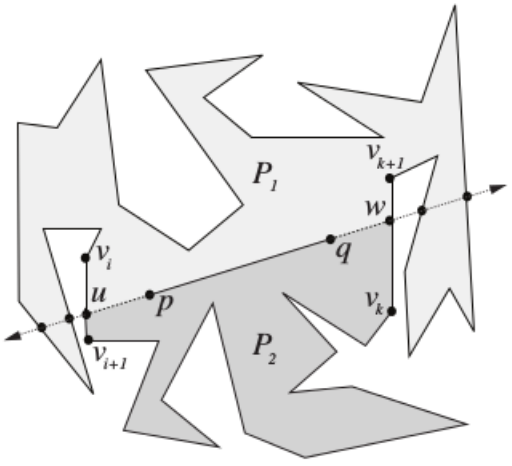
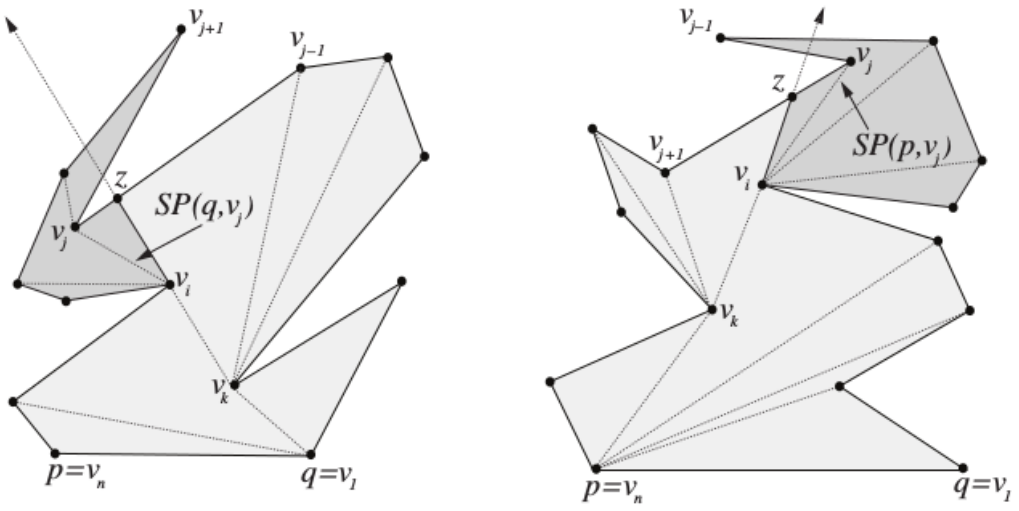


FIGURA 1.8: Cuando pq no es una arista poligonal convexa, dividimos P con uw en dos subpolígonos P_1 y P_2

[1]Esto significa que los ángulos internos de sus dos vértices no son mayores a π.

[2]Un rayo es una línea que comienza en un punto y va hacia el infinito en una dirección en particular.

Dada la discusión anterior, ahora se puede simplificar las cosas y suponer que pq es una arista convexa del polígono dado P. Así que se presentará un algoritmo para calcular $V(pq)$ desde una arista convexa pq de P. Se considera que los vértices de P están etiquetados en el sentido contrario al de las manecillas del reloj $(v_1, v_2, \ldots, v_n)$, donde $q = v_1$ y $p = v_n$. En los corolarios 1.7 y 1.8, productos del lema 1.6, se muestran las ideas principales presentadas para el algoritmo 1.3.1, el cual calcula el polígono de visibilidad débil $V(pq)$, dado el segmento pq en un polígono simple P. La correctez del algoritmo 1.3.1 sigue directamente del lema 1.6.



(A) $SP(q, v_j)$ da una primer vuelta a la izquierda en v_j

(B) $SP(p, v_j)$ da la primera vuelta a derecha en v_i

FIGURA 1.9

Lema 1.6. *Sea $v_k v_{k+1}$ una arista convexa de un polígono simple P. Un vértice v_i de P es visible desde algún punto de $v_k v_{k+1}$ si y sólo si $SP(v_k, v_i)$ da vueltas a la izquierda en todo vértice de la trayectoria y $SP(v_{k+1}, v_i)$ da vueltas a la derecha en todo vértice de la trayectoria.*

Demostración

(Véase la figura 1.9) Si v_i es visible desde un punto u de $v_k v_{k+1}$, entonces $SP(v_k, v_i)$ no puede intersectar el segmento uv_i . De manera que todo vértice en $SP(v_k, v_i)$ pertenece a $bd(v_i, v_k)$. Por lo tanto, $SP(v_k, v_i)$ sólo puede dar vueltas a la izquierda en todo vértice de la trayectoria. Análogamente, $SP(v_{k+1}, v_i)$ da vuelta a la derecha

en todo vértice de la trayectoria. Para probar el recíproco, sea v_p y v_q el vértice siguiente de v_i en $SP(v_k, v_i)$ y $SP(v_{k+1}, v_i)$ respectivamente. Dado que $SP(v_k, v_i)$ da vueltas a la izquierda en todo vértice de la trayectoria, v_k queda a la derecha de $\overrightarrow{v_i v_p}$. Análogamente, como $SP(v_{k+1}, v_i)$ da vueltas a la derecha en todo vértice de la trayectoria, v_{k+1} queda a la izquierda de $\overrightarrow{v_{k+1} v_i}$ Sea h la porción generada desde la intersección de $v_k v_{k+1}$ con $\overrightarrow{v_i v_p}$ hasta la intersección de $v_k v_{k+1}$ con $\overrightarrow{v_{k+1} v_i}$. De manera que v_k y v_{k+1} quedan en lados opuestos de la porción h. Por lo que v_i es visible desde cualquier punto de $v_k v_{k+1}$ que esté en la porción h. □

Corolario 1.7. *Sea v_i el padre de v_j en $SPT(p)$ tal que $SP(p, v_j)$ da la primer vuelta a la derecha en v_i . Entonces todos los descendientes de v_i en $SPT(p)$ no son visibles desde ningún punto de pq (Véase la Figura 1.9a).*

Corolario 1.8. *Sea v_i el padre de v_j en $SPT(q)$ tal que $SP(q, v_j)$ da la primera vuelta a la derecha en v_i . Entonces todos los descendientes de v_i en $SPT(q)$ no son visibles desde ningún punto de pq (Véase la Figura 1.9b).*

Teorema 1.9. *El polígono de visibilidad $V(pq)$ de un segmento pq, en un polígono simple P de n lados, puede ser calculado en tiempo $O(n)$.*

Demostración

Por el lema 1.5, calcular $SPT(p)$ y $SPT(q)$ toma tiempo $O(n)$. Además, cada vértice de $SPT(p)$ y $SPT(q)$ es visitado una sola vez y las operaciones restantes toman tiempo constante. Por lo que la complejidad en tiempo es de $O(n)$. □

Algoritmo 1.3.1 Polígono de Visibilidad débil $V(pq)$

Entrada: Un polígono P y un segmento de línea pq
Salida: $V(pq)$

1: Calcular $SPT(p)$ en P.
2: Haciendo un recorrido en profundidad sobre $SPT(p)$ revisar la dirección de la vuelta en cada vértice v_i en $SPT(p)$.
3: **para todo** v_i en $SPT(p)$ **hacer**
4: Checar la dirección en la que da vuelta.
5: **si** La trayectoria da vuelta a la derecha en v_i **entonces**
6: Encontrar el descendiente de v_i en $SPT(p)$ con el mayor índice j.
7: Calcular el punto de intersección z de v_jv_{j+1} y $\overrightarrow{v_kv_i}$, donde v_k es el padre de v_i en $SPT(p)$.
8: Quitar la frontera de P desde v_i a z, en sentido opuesto de las manecillas del reloj, insertando el segmento v_iz.
9: **fin si**
10: **fin para**
11: Calcular $SPT_(q)$ en P', donde P' la porción remanente de P.
12: Haciendo un recorrido en Profundidad sobre $SPT(q)$ de P' revisar la dirección de la vuelta en cada vértice v_i en $SPT(q)$ de P'.
13: **para todo** v_i en $SPT(q)$ **hacer**
14: Checar la dirección en la que da vuelta.
15: **si** La trayectoria da vuelta a la izquierda en v_i **entonces**
16: Encontrar el descendiente de v_i en $SPT(p)$ con el menor índice j.
17: Calcular el punto de intersección z de v_jv_{j-1} y $\overrightarrow{v_kv_i}$, donde v_k es el padre de v_i en $SPT(q)$.
18: Quitar la frontera de P' desde v_i a z, en sentido opuesto de las manecillas del reloj, insertando el segmento v_iz.
19: **fin si**
20: **fin para**
21: **devolver** La porción remanente de P' como $V(pq)$.

1.4. Trayectoria mínima en aristas al interior de polígonos simples

Se considera un robot-punto que se mueve en caminos de línea recta dentro de un polígono P. Cada vez que tiene que cambiar de dirección del camino, se detiene y rota hasta que consiga la dirección adecuada. En el proceso, hace muchas vueltas hasta llegar a la posición objetivo. Si los movimientos en línea recta son "baratos", pero las vueltas son "caras", reducir el número de líneas rectas en el camino minimiza el costo de las vueltas pero aumenta la distancia de la ruta. Esto motiva el estudio de las trayectorias en aristas dentro de la región de un polígono P. Una trayectoria en aristas entre dos puntos s y t de un polígono P es un camino que conecta a s y t por una cadena de segmentos de línea (llamadas aristas). Una trayectoria mínima en aristas $MLP(s,t)$ es una trayectoria que conecta a s y a t teniendo el mínimo número de aristas necesarias para esa tarea. Hay que observar que puede haber muchas trayectorias mínimas en aristas entre dos puntos s y t. La distancia mínima en aristas $MLD(s,t)$ es el número de aristas en la trayectoria mínima en aristas entre s y t dentro de P. En general, $MLP(s,t)$ no es única y normalmente existe un número infinito de trayectorias mínimas en aristas entre dos puntos. El problema de calcular la trayectoria mínima en aristas entre dos puntos dentro de un polígono simple fue estudiado por primera vez por ElGindy en [11] y por Suri en [12]. En este capítulo se describirá un algoritmo que calcula una trayectoria mínima en aristas entre s y t en P.

1.4.1. Trayectorias mínimas en aristas

En esta sección se presentará un algoritmo que en tiempo lineal pueda calcular la trayectoria mínima en arista entre dos puntos s y t en el interior de un polígono simple P. Se considera que el polígono de entrada P tiene los vértices etiquetados como $v_1, v_2, \ldots, v_n$ en orden contrario a las manecillas del reloj. Se denotará por $MLP(s,t)$ a la trayectoria más corta entre s y t en P.

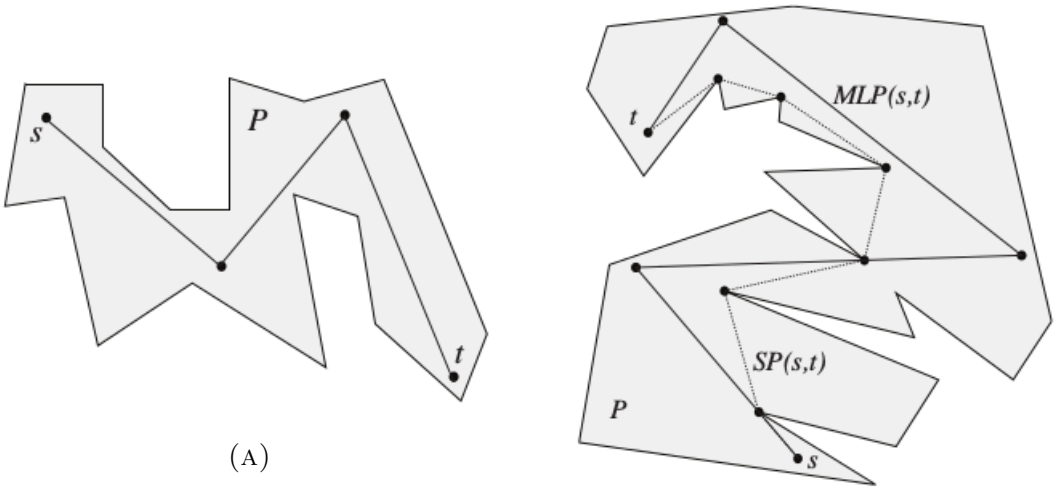


FIGURA 1.10: En (a) siempre son necesarias al menos tres aristas para conectar a s y t. En (b) se muestra $MLP(s,t)$ comparado con $SP(s,t)$

Es evidente que $V(s)$, el polígono de visibilidad de s en P, es el conjunto de todos los puntos en P que pueden ser alcanzados por una arista desde s (ver figura 1.11). Si $t \in V(s)$, entonces el segmento st es la trayectoria mínima en aristas de s a t. Denotaremos la región $V(s)$ como R_1 y a P como P_0, entonces R_1 es la región a la cual s puede llegar con una sola arista, es decir $MLD(s, R_1) = 1$. El polígono de visibilidad de P_0 desde s es el conjunto R_1 de todos los puntos de P_0 que pueden ser alcanzados desde s por una arista.

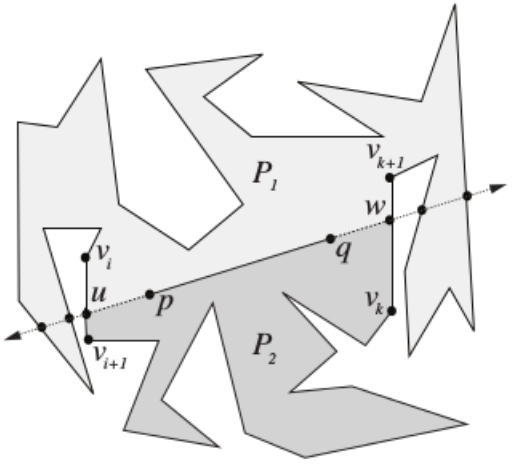


FIGURA 1.11: Todo $MLP(s,t)$ intersecta a u_1z_1, u_2z_2 y u_3z_3, las cuales son aristas construidas de R_1, R_2 y R_3 respectivamente.

Ahora se considera la situación en la que t no puede ser alcanzado desde s por una arista. Como $t \in P_0 - R_1$, toda trayectoria mínima en aristas de s a t debe

intersectar una arista construida de R_1 para poder alcanzar t. De hecho, toda $MLP(s,t)$ intersecta la misma arista no poligonal u_1z_1 de R_1 (ver figura 1.11), dado que P es un región cerrada y delimitada. Sin embargo, este conjunto de trayectorias $MLP(s,t)$ intersectan a u_1z_1 en diferentes puntos.

Lema 1.10. *Existe una trayectoria mínima en aristas en $MLP(s,t)$ tal que el primer punto de giro de $MLP(s,t)$ está en la arista construida u_1z_1.*

Demostración

Consideremos cualquier $MLP(s,t)$. Sea w_1 el punto de intersección entre u_1z_1 y $MLP(s,t)$. Sea $k_1 \in R_1$ el punto de giro en $MLP(s,t)$ y entonces la primera arista $L_1 = sk_1$ puede remplazarse por la arista sw_1 para construir otra $MLP(s,t)$ la cual tendrá como primer punto de giro a w_1. □

Extrayendo R_1 de P_0 se genera un subpolígono de P por cada arista construida de R_1. Para identificar la arista construida u_1z_1 sobre las demás aristas construidas de R_1, hay que identificar el subpolígono P_1 que contiene a t y la arista construida de R_1 que está en la frontera de P_1 es u_1z_1.

Lo que resta por explicar es como localizar el primer punto de giro w_1 sobre u_1z_1. Calculemos el conjunto de puntos $R_2 \subseteq P_1$ que pueden ser alcanzados desde s por dos aristas. R_2 es el conjunto de puntos de P_1 que pueden ser alcanzados por una arista desde algún punto de R_1, pero por el lema 1.10 se puede restringir a los puntos que pueden ser alcanzados por una arista desde u_1z_1. Si recordamos la definición de visibilidad débil de una arista, R_2 es el polígono de visibilidad débil de P_1, desde u_1z_1, es decir $R_2 = V(u_1z_1)$ en P_1. Si $t \in R_2$ tomando cualquier punto w_1 sobre u_1z_1 tal que w_1t sea una arista en el interior de R_2, entonces $MLP(s,t) = (sw_1, w_1t)$. De lo contrario, extrayendo R_2 de P_1, P_1 se divide en subpolígonos y uno de los subpolígonos P_2 contiene a t. Sea u_2z_2 la arista construida de R_2 que está en la frontera de P_2. Continuamos con este proceso de localizar P_i y calcular R_{i+1} desde la arista construida u_iz_i, hasta que t sea visible desde u_mz_m, donde m es la distancia mínima en aristas entre s y t. Resumimos la construcción anterior con el siguiente par de lemas.

Lema 1.11. *Sea u_iv_i la arista no poligonal de R_i tal que después de remover R_i de P_{i-1}, u_iv_i es una arista del subpolígono P_i que contiene a t. Para $1 \leq i < MLD(s,t)$, toda $MLP(s,t)$ intersecta u_iv_i y existe una $MLP(s,t)$ cuyo i-ésimo punto de giro está en u_iv_i.*

Lema 1.12. *Para $1 \leq i < MLD(s,t)$, el conjunto R_{i+1} de todos los puntos en P_i que pueden ser alcanzados desde s por $i+1$ aristas es el polígono de visibilidad débil de P_i desde u_iv_i.*

Basado en los lemas anteriores, se da el siguiente algoritmo para calcular $MLP(s,t)$ en P_0.

Algoritmo 1.4.1 Trayectoria mínima en aristas $MLP(s,t)$

Entrada: Un polígono P y dos puntos $s, t \in P$.
Salida: $MLP(s,t)$

1: Calcular $T(P)$
2: $i \leftarrow 1$
3: Calcular $V(s)$ y llamar R_i a esta región.
4: **si** t es visible desde s **entonces**
5: $MLP(s,t) \leftarrow st$
6: **devolver** $MLP(s,t)$
7: **si no**
8: **repetir**
9: Identificar el subpolígono de P_{i-1} que contiene a t y llamarlo P_i.
10: Nombrar como $u_{i-1}v_{i-1}$ a la arista no poligonal de R_i que forma parte de la frontera de P_i.
11: **si** $i > 1$ **entonces**
12: Asignemos como w_{i-1} al punto de intersección entre $u_{i-1}v_{i-1}$ y v_iu_i.
13: **fin si**
14: Calcular $V(u_iv_i)$ en P_i y llamemoslo R_{i+1}.
15: $i \leftarrow i+1$.
16: **hasta que** t sea visible desde cualquier punto de u_iv_i, es decir, hasta que $t \in V(u_iv_i)$.
17: **fin si**
18: Localizar un punto $w_i \in V(t)$ sobre u_iv_i.
19: $MLP(s,t) \leftarrow (sw_1, w_1w_2, \ldots, w_it)$.
20: **devolver** $MLP(s,t)$.

Demostrar que el algoritmo es correcto se sigue de los Lemas 1.10, 1.11 y 1.12. Se analizará la complejidad del Algoritmo 1.4.1. Utilizando el algoritmo de Lee en [5]

podemos calcular el polígono de visibilidad $V(s)$ en el Paso 3 y $V(t)$ en el Paso 18 en tiempo $O(n)$.

Identificar el subpolígono de P_{i-1} que contiene a t en el Paso 9, también se puede hacer en tiempo $O(n)$ como se describe a continuación. Sea uv la arista no poligonal de p_{i-1} que pertenece a R_i. Dado que $R_i = V(s)$ cuando $i = 1$ o $R_i = V(u_{i-1}v_{i-1})$, entonces sabemos que uno de los puntos finales de la arista no poligonal es un vértice de P. Sin pérdida de generalidad, se considera que u es el vértice v_j de P. Sea v_kv_{k+1} la arista de P tal que $z \in v_kv_{k+1}$. Sea v_p un vértice visible desde t, por lo que $v_p \in V(t)$. Si uz intersecta tv_p, el punto de intersección se convierte en w_i. Si uz no intersecta tv_p, comparando j, k, p podemos determinar la porción de la frontera de P_{i-1} que forma parte de la frontera de P_i. Una vez que P_i es reconocido, la arista no poligonal de R_i asociada a P_i se convierte en u_iz_i. Por lo que gastando tiempo $O(1)$ para cada arista no poligonal de R_i, podemos localizar u_iz_i. Como sólo puede haber una arista construida por cada vértice cóncavo de P, el Paso 9 toma tiempo $O(n)$.

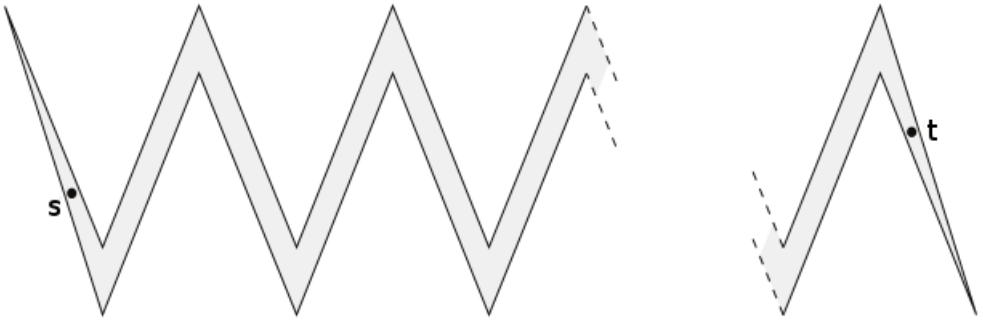


FIGURA 1.12: Un polígono donde $MLD(s,t) = \frac{n}{2}$

Existen polígonos en los cuales pueden existir puntos s y t para los cuales $MLD(s,t) = n/2$.

En la figura 1.12 se muestra un polígono con puntos s y t donde $MLD(s,t) = n/2$. Además, observando la figura 1.12 es fácil ver que para todo polígono y para cualesquiera dos puntos s y t dentro de este, $MLD(s,t) \leq n/2$. Por lo tanto, $i = 1, \ldots, n/2$ y entonces $i = O(n)$ en el Algoritmo 1.4.1.

En la sección 1.3.1 se demostró que calcular el polígono de visibilidad utilizando el Algoritmo 1.4.1 nos toma tiempo $0(n)$. Dado que $i = O(n)$ tendríamos que mandar llamar al Algoritmo 1.3.1 un número lineal de veces, con lo que el Algoritmo 1.4.1 tendría una complejidad de $= O(n^2)$. Para reducir éste tiempo cuadrático sólo hay que modificar un poco el Algoritmo 1.3.1.

La complejidad $O(n)$ del Algoritmo 1.3.1 se debe a que construimos $SPT(u)$ y $SPT(v)$ completos, es decir que recorremos toda la gráfica dual de la triangulación que, como ya vimos anteriormente, es de tamaño $O(n)$. El truco para reducir la complejidad de $O(n^2)$ a $O(n)$ es extender solamente el árbol de trayectorias más cortas hacia los vértices de los triángulos que son al menos parcialmente visibles desde uv, y entonces el tiempo total para calcular $V(uv)$ en el paso 14 es $O(k_e)$, donde k_e es el número de triángulos en $T(P)$ intersectados por $V(uv)$. Es decir, calculamos $SPT(u)$ y $SPT(v)$ tal y como en el Algoritmo 1.3.1, avanzando de una diagonal a otra y revisando igualmente la dirección de la vuelta (izquierda o derecha) en cada vértice de $SPT(u)$ o $SPT(v)$. Entonces, al igual que antes, el tiempo total estará acotado por el tiempo necesario para construir ambos árboles de trayectorias más cortas, pero como ahora únicamente serán hojas de los árboles los vértices de los k_e triángulos, el Paso 14 toma tiempo $O(k_e)$.

En la siguiente sección hablaremos sobre la partición por ventanas, una estructura que nos permitirá demostrar que cualquier triángulo de $T(P)$ intersecta únicamente un número constante de regiones de la partición por ventanas. Lo cual implica que el Algoritmo 1.4.1 corre en tiempo $O(n)$.

1.4.2. Partición por ventanas de un polígono simple

En esta sección se presenta una estructura que será de gran utilidad para analizar el algoritmo 1.4.1. Esta partición por ventanas es simplemente otra forma un poco distinta de ver la teoría desarrollada en la sección anterior. Esta estructura es utilizada en el algoritmo original de Suri en [12] y [13] para calcular trayectorias mínimas en aristas en polígonos simples.

Ésta estructura es resultado de particionar un polígono simple en regiones sobre las cuales la distancia en aristas permanece constante con respecto al punto sobre el cual se construirá la partición. Es decir que se particionará P con respecto a un punto x en regiones $R_1, R_2, \ldots, R_m$ tales que para $1 \leq i \leq m$, todos los puntos de R_i pueden ser alcanzados por i aristas desde x. A esta partición se le llama partición por ventanas. La gráfica dual de una partición por ventanas es un árbol el cual es llamado árbol de ventanas. Sea x un punto o una arista dentro de P, una ventana es definida como una arista del polígono de visibilidad $V(x)$ tal que esta arista no pertenece al borde de P.

Sea c una cuerda y sea x un segmento de línea en P tal que c y x no se intersectan. La cuerda c divide el polígono P en dos subpolígonos, de los cuales sólo uno contiene a x. Se utilizará $P[c; x]$ para denotar el subpolígono que no contiene a x. Estas definiciones se ilustran en la Figura 1.13. El algoritmo 1.4.2 es el algoritmo recursivo original de Suri en [12] para construir la partición por ventanas de un polígono simple P con respecto a x.

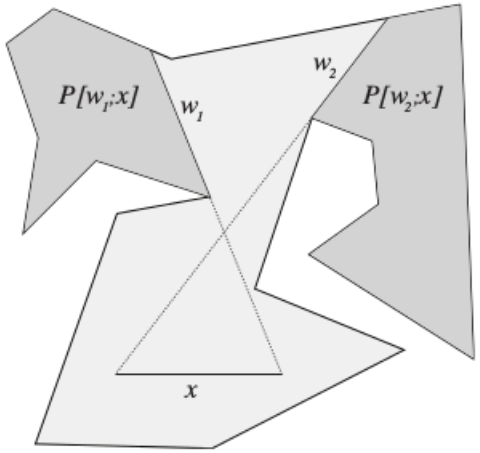


FIGURA 1.13: Se muestran dos ventanas w_1 y w_2 y los subpolígonos $P[w_1; x]$ y $P[w_2; x]$.

Sea w una ventana en la partición de P, decimos que la región $R(w)$ está asociada con (o es generada por) w. Aunque cada ventana de la partición es compartida por dos regiones, se asignará la ventana a la región con menor distancia en aristas

desde la fuente. Es fácil ver que cada región en la partición por ventanas tiene al menos un vértice de P y que el tamaño de la gráfica plana dual sobre la partición por ventanas es $O(n)$.

Tómese en cuenta que en el algoritmo 1.4.2 el valor inicial de j es 1 y el valor inicial de R es $\{\}$.

Algoritmo 1.4.2 Partición por Ventanas $W(x, P, R, j)$

Entrada: Un polígono simple P y un punto o un segmento $x \in P$.
Salida: $R(x)$ = colección de regiones que juntas son una partición de P

1: Calcular el polígono de visibilidad $V(x)$ en P y llamar R_j a esta región.
2: $R \leftarrow R \cup R_j$.
3: Sean $w_1, w_2, \ldots, w_k$ las ventana de $V(x)$.
4: **para todo** $i = 1$ hasta k **hacer**
5: $\quad R \leftarrow R\cup$ Partición por Ventanas $W(w_i, P[w_i, x], R, j + 1)$.
6: **fin para**
7: **devolver** R.

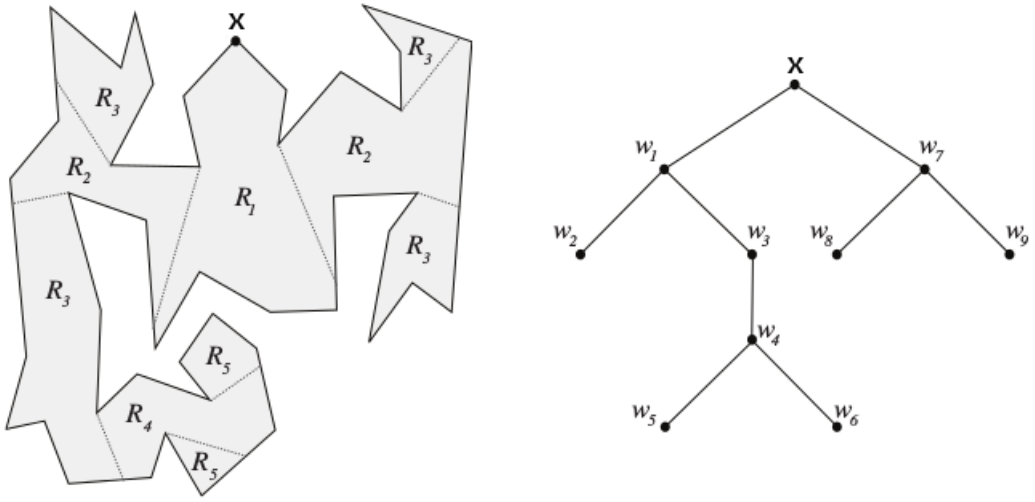


FIGURA 1.14: La partición por ventanas de P y su respectivo árbol de ventanas $WT(x)$

A partir de de la partición por ventanas se puede definir un *Árbol de Ventanas* $WT(x)$. Un árbol de ventana $WT(x)$ es una gráfica plana dual de P con respecto a x definida de la siguiente manera: $WT(x)$ tiene un nodo por cada región de la partición por ventanas y una arista que une a dos nodos si y sólo si las regiones comparten una ventana.

Ya que se definió lo que es una partición por ventanas podemos regresar al análisis de Algoritmo 1.4.1. Observese que en el Algoritmo 1.4.2 cada polígono de visibilidad calculado define una región en la partición por ventanas de P. Por lo tanto, la cota $O(n)$ en tiempo para el Algoritmo 1.4.1 se cumplirá si todo triangulo en $T(P)$ intersecta únicamente un número constante de regiones de la partición por ventanas de P. Se probará esto en el siguiente lema. Ver Figura 1.15.

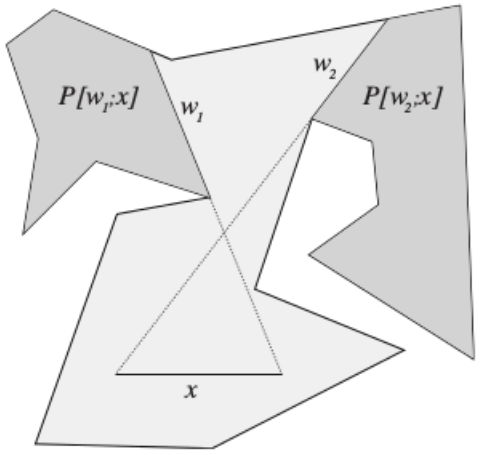


FIGURA 1.15: Intersección de un triángulo de $T(P)$ con tres regiones.

Lema 1.13. *Cualquier triángulo $\triangle \in T(P)$ intersecta a lo más tres regiones de la partición por ventanas de P.*

Demostración

Sea $W(w_i, P[w_i; w_j])$ la primera vez que el triangulo triángulo $\triangle \in T(P)$ es usado, esto es, el polígono de visibilidad de w_i calculado en $P[w_i; w_j]$ intersecta el interior de triángulo $\triangle$. Si está completamente contenido en $V[w_i; w_j](w_i)$, entonces no puede pertenecer a ninguna otra región de la partición, y el lema se cumple. De no ser este el caso, triángulo $\triangle \cap V_P[w_i; w_j](w_i)$ es un conjunto conectado de P y está acotado por a lo más dos ventanas de $V_P[w_i; w_j](w_i)$, se le llamarán w_{i1} y w_{i2}. (Ver la Figura 1.15) Pero entonces, es fácil ver que está completamente contenido en la unión del polígono de visibilidad de w_i en $P[w_i; w_j]$, el polígono de visibilidad de w_{i1} en $P[w_{i1}; w_i]$ y el polígono de visibilidad de w_{i2} en $P[w_{i2}; w_i]]$, por lo que se cumple lo que se quería mostrar. □

En resumen, cada región de una partición por ventanas de P puede ser calculada en tiempo proporcional al número de triángulos en $T(P)$ intersectados por esta, y ningún triángulo es intersectado por más de tres regiones. Esto junto con el desarrollo de la sección anterior prueba el siguiente teorema.

Teorema 1.14. *La trayectoria mínima en aristas entre dos puntos s y t en un polígono simple P de n vértices puede ser calculada en tiempo $O(n)$.*

Corolario 1.15. *Sea P un polígono triangulado de n vértices. Sea x un punto o un segmento de línea en P. La partición por ventanas de P con respecto a x puede ser calculada en tiempo $O(n)$.*

Capítulo 2

Algoritmo exponencial exacto para encontrar el árbol de Steiner mínimo en aristas dentro de un polígono simple

Dado un polígono simple P de n vértices y un conjunto de puntos terminales $Z = \{z_1, z_2, \ldots, z_m\}$ dentro de P. El objetivo es encontrar el árbol de Steiner mínimo en aristas que expanda a Z tal que todo segmento de recta esté totalmente contenido en P. Para llevar a cabo esta tarea, es posible agregar puntos nuevos dentro de P para lograr la interconexión entre puntos terminales. Estos puntos nuevos son llamados vértices de Steiner.

En este capítulo veremos un algoritmo exponencial para resolver este problema y analizaremos su complejidad en tiempo y espacio.

2.1. Algoritmo exacto para encontrar SMT(Z,P)

Definición 2.1. Sea P un polígono simple y $Z = \{z_1, z_2, \ldots, z_m\}$ un conjunto de puntos terminales dentro de P, definimos $SMT(Z, P)$ como el árbol de Steiner

mínimo en aristas que une a todos los puntos de Z y donde toda arista está totalmente contenida en P. De igual manera, definimos $smt(Z, P)$ como la cardinalidad del conjunto las aristas de $SMT(Z, P)$.

El algoritmo para encontrar el árbol de Steiner mínimo en aristas que se muestra en este capítulo, es un algoritmo de programación dinámica tipo *bottom-up* (desde el fondo hacia arriba), lo que significa que el algoritmo empieza resolviendo los casos base y al final se resuelve el problema original. Al resolver el problema para subconjuntos de Z de cardinalidad 2, es decir, para algún $Z^{'} = \{z_i, z_j\} \in Z$ donde $1 \leq i, j \leq w$ y $i \neq j$, es evidente que $SMT(Z^{'}, P) = MLP(z_i, z_j)$.
De ahora en adelante se utilizará $smt(Z)$ y $SMT(Z)$ en vez de $smt(Z, P)$ y $SMT(Z, P)$, respectivamente.

Ahora se retomará y se cambiará un poco la definición de R_i del capítulo 1.4.

Definición 2.2. Sea P un polígono simple y $Z = \{z_1, z_2, \ldots, z_m\}$ un conjunto de puntos terminales dentro de P, se define a $R_i(z_k)$, donde $z_k \in Z$, como la región donde z_k puede llegar utilizando i aristas.

Lema 2.3 ([1])**.** *Sea $S_{\{i_1,i_2\},j}$ para algún $\{z_{i_1}, z_{i_2}\} \in \binom{Z}{2}$ la región donde se puede colocar el punto p_j de $SMT(\{z_{i_1}, z_{i_2}\})$, donde $0 \leq j \leq smt(\{z_{i_1}, z_{i_2}\})$. Entonces, $S_{\{i_1,i_2\},j} = R_j(z_{i_1}) \cap R_{l_{\{i_1,i_2\}}-j}(z_{i_2})$, donde $l_{\{i_1,i_2\}} = MLD(z_{i_1}, z_{i_2})$.*

Demostración

Cuando $j = 0$ y $j = smt(\{z_{i_1}, z_{i_2}\})$, la región del terminal está dada por $R_0(z_{i_1}) \cap R_{l_{\{i_1,i_2\}}}(z_{i_2}) = z_{i_1}$ y $R_{l_{\{i_1,i_2\}}}(z_{i_1}) \cap R_0(z_{i_2}) = z_{i_2}$, respectivamente, dado que $R_0(z_{i_1}) = z_{i_1}$ y $R_0(z_{i_2}) = z_{i_2}$. Lo cual es valido por que la región donde puedo colocar a los terminales z_{i_1} y z_{i_2} es precisamente el punto mismo z_{i_1} y z_{i_2}, respectivamente. Luego, si $1 \leq j \leq smt(\{z_{i_1}, z_{i_2}\}) - 1$ entonces p_j es un punto de Steiner y su región donde puede ser colocado está dada por la región en la que puedo llegar en j aristas desde el terminal z_{i_1}, $R_j(z_{i_1})$ intersección con la región en la que puedo llegar en $l_{\{i_1,i_2\}} - j$ aristas desde el terminal z_{i_2}. □

Como se mencionó anteriormente, el algoritmo dado en esta sección es un algoritmo de programación dinámica. Todo algoritmo de programación dinámica es representado por una relación de recurrencia en donde un problema es definido como la unión de ciertos subproblemas. Es necesario escribir esta relación de recurrencia y mostrar que esta es correcta.

Lema 2.4 ([1]). *Sea P un polígono simple y $Z = \{z_1, z_2, \ldots, z_m\}$ un conjunto de puntos dentro de P, el número de aristas del árbol de Steiner mínimo de Z en P está dado por la siguiente recurrencia:*

$$smt(Z) = \begin{cases} MLD(z_{i1}, z_{i2}) & \text{si } |Z| = 2 \\ \min\limits_{\{z_{i_1},\ldots,z_{i_{w-1}}\}\in\binom{Z}{w}} \Big(smt(\{z_{i_1}, \ldots, z_{i_{w-1}}\}) & \\ + \min\limits_{0\le j\le smt(\{z_{i_1},\ldots,z_{i_{w-1}}\})} \Big(MLD(\{z_{i_1}, \ldots, z_{i_w}\} & \\ -\{z_{i_1}, \ldots, z_{i_{w-1}}\}, S_{\{z_{i_1},\ldots,z_{i_{w-1}}\},j}) \Big) \Big) & \text{si } |Z| > 2 \end{cases}$$

donde $S_{\{z_{i_1},\ldots,z_{i_{w-1}}\},j}$ es la región donde se puede colocar el punto p_j de $SMT(\{z_{i_1}, \ldots, z_{i_{w-1}}\})$ para $0 \le j \le smt(\{z_{i_1}, \ldots, z_{i_{w-1}}\})$.

Demostración

Supongamos que tenemos $SMT(Z)$ que está compuesto por $SMT(Z - \{z_k\}) \cup MLP(z_k, v_s)$ donde $z_k \in Z$ y s_i es una región en donde se puede colocar un vértice Steiner en $SMT(Z - \{z_k\})$. Si construimos un árbol de Steiner $ST(Z - \{z_k\}, P)^*$ que también utilice la región v_s y tenga menor cardinalidad que $SMT(Z - \{z_k\})$ o construimos una trayectoria en aristas $LP(z_k, v_s)^*$ más corta que $MLP(z_k, v_s)$, entonces podemos construir un árbol de Steiner menor a $SMT(Z)$, lo que contradice que $SMT(Z)$ es mínimo. □

Antes de hablar de los algoritmos, se dará una definición que describe a $SMT(Z, P)$ como una gráfica.

Definición 2.5. Sea P un polígono simple y $Z = \{z_1, z_2, \ldots, z_m\}$ un conjunto de puntos terminales dentro de P, definimos $SMT(Z, P)$ como la tupla (S, L), donde S es un conjunto donde cada elemento es una región donde se puede poner un vértices de Steiner o un vértice terminal y L es un lista de adyacencia de los elementos de S. L representa que regiones de S son alcanzables entre si en P por una arista.

Supongamos que se tiene un conjunto $Z = \{z_1, z_2, \ldots, z_m\}$ de vértices terminales y un conjunto $Z^* = \{z_1, z_2, \ldots, z_{w-1}\} \subset Z$ con $2 < w < m$. Al momento de agregar un nuevo terminal z_w a un árbol de Steiner mínimo $SMT(Z^*)$, el algoritmo que describe la recurrencia del lema 2.4 selecciona una región de Steiner S_i a la cual se conectará z_w mediante el algoritmo $MLP(z_w, S_i)$. En esta situación pueden ocurrir dos cosas:

- S_i es algún otro vértice terminal, es decir, $S_i = z_k$, donde $z_k \in Z^*$, entonces, para conectar a z_w y z_k es necesario calcular $MLP(z_w, z_k)$.
- S_i es una región de Steiner, por lo que $MLP(z_w, S_i)$ es una trayectoria mínima en aristas entre un punto y una región.

En el segundo caso es fundamental recalcular la región de Steiner S_i, dado que es posible que no todo punto en S_i sea alcanzable desde z_w con $MLD(z_w, S_i)$ aristas. Por lo que $S_i^* = R_{MLD(z_w,S_i)}(z_w) \cap S_i$ tomará el lugar de S_i en $SMT(Z^*)$. Sin embargo, es posible que la vecindad de $S*_i$ no sea visible en su totalidad desde S_i^*, es decir, es posible que $V(S_i^*) \cap S_j \neq V(S_i^*)$ ocurra, donde S_j es alguna región de Steiner adyacente a S_i^* en $SMT(Z^*)$, por lo que es necesario reemplazar S_j por $S_j^* = V(S_i^*) \cap S_j$. Es evidente que se tiene que realizar el mismo proceso para cada región de Steiner de $SMT(Z^*)$. Para lograr esto, el algoritmo 2.1.1 hace una búsqueda en amplitud sobre $SMT(Z^*)$ empezando por S_i^* para asegurar la recalculación de todas las regiones de Steiner.

Algoritmo 2.1.1 BFS-Ajuste

Entrada: Un árbol de Steiner $SMT^* = (S, L)$ y un entero j^* que es el indice de una región de Steiner.

Salida: $SMT(Z_w)$

1: Sea Q una cola de elementos enteros.
2: $Q \leftarrow \emptyset$
3: Sea $Marca$ un arreglo de tamaño $|S|$ de elementos booleanos, todos estos con valor inicial de **cierto** .
4: Sea t un entero.
5: $Marca[j^*] \leftarrow$ **falso**
6: $Marca[j^* + 1] \leftarrow$ **falso**
7: $Q.push(hijo)$
8: **mientras** $|Q| \neq 0$ **hacer**
9: $t \leftarrow Q.front()$
10: $Q.pop()$
11: Sea $L[t]$ el conjunto de regiones vecinas de $S[t]$
12: **para todo** vecino $v \in L[t]$ **hacer**
13: **si** $Marca[v] =$ **cierto entonces**
14: $Marca[v] \leftarrow$ **falso**
15: $S[v] \leftarrow V(S[t]) \cap S[v]$
16: $Q.push(v)$
17: **fin si**
18: **fin para**
19: **fin mientras**
20: **devolver** (S, L)

Algoritmo 2.1.2 Árbol de Steiner Mínimo en aristas

Entrada: Un polígono simple P de n vértices y un conjunto de puntos $Z = \{z_1, z_2, \ldots, z_m\}$ en P.

Salida: $SMT(Z)$

1: **para todo** $\{z_{i_1}, z_{i_2}\} \in \binom{Z}{2}$ **hacer**
2: $smt(\{z_{i_1}, z_{i_2}\}) \leftarrow MLD(z_{i_1}, z_{i_2})$
3: $l_{\{i_1,i_2\}} \leftarrow smt(\{z_{i_1}, z_{i_2}\})$
4: **para** $j \leftarrow 0$ hasta $j \leftarrow smt(\{z_{i_1}, z_{i_2}\})$ **hacer**
5: $S_{\{i_1,i_2\},j} \leftarrow R_j(z_{i_1}) \cap R_{l_{\{i_1,i_2\}}-j}(z_{i_2})$
6: **fin para**
7: **fin para**
8: **para** $w \leftarrow 3$ hasta $w \leftarrow m$ **hacer**
9: **para todo** $Z_w = \{z_{i_1}, \ldots, z_{i_w}\} \in \binom{Z}{w}$ **hacer**
10: $smt(Z_w) \leftarrow \min_{Z_{w-1}=\{z_{i_1},\ldots,z_{i_{w-1}}\}\in\binom{Z_w}{w-1}} \{smt(Z_{w-1}) + \min_{0\le j\le smt(Z_{w-1})} \{MLD(z_{i_w}, S_{Z_{w-1},j})\}\}$
11: Sean Z^*_{w-1} y j^* los mínimos y $z^*_{i_w}$ el punto seleccionado en el paso anterior.
12: $SMT^* \leftarrow SMT(Z_{w-1}) \cup MLP(z^*_{i_w}, S_{Z^*_{w-1},j^*})$
13: $l_{\{i_1,\ldots,i_w\}} \leftarrow MLD(z^*_{i_w}, S_{\{Z^*_{w-1},j^*\}})$
14: **para** $j \leftarrow 0$ hasta $j \leftarrow smt(Z_w)$ **hacer**
15: **si** $0 \le j < j^*$ **entonces**
16: $S_{Z_w,j} \leftarrow S_{Z^*_{w-1},j}$
17: **si no**, **si** $j^* \le j \le j^* + l_{\{i_1,\ldots,i_w\}}$ **entonces**
18: $S_{Z_w,j} \leftarrow R_{j-j^*}(S_{Z^*_{w-1},j^*}) \cap R_{l_{\{i_1,\ldots,i_w\}}-(j-j^*)}(z^*_{w_i})$
19: **si no**
20: $S_{Z_w,j} \leftarrow S_{Z^*_{w-1},j-l_{\{i_1,\ldots,i_w\}}}$
21: **fin si**
22: **fin para**
23: $SMT(Z_w) \leftarrow$ **BFS-Ajuste**(SMT^*, j^*)
24: **fin para**
25: **fin para**
26: **devolver** $SMT(Z)$

2.2. Análisis de tiempo y espacio

Lema 2.6. *Dado un polígono P con n vértices y un conjunto de puntos $Z = \{z_1, z_2, \ldots, z_m\}$ dentro de P.*

$$m - 1 \leq smt(Z) < 2 \cdot \lfloor \tfrac{n}{3} \rfloor + m$$

Demostración

Tomando los vértices de la clase cromática de menor cardinalidad, sobre una 3-coloración de alguna triangulación de P, por el teorema 1.3, sabemos que siempre son suficientes $\lfloor \frac{n}{3} \rfloor$ vértices de Steiner para formar una arista entre los m terminales y estos vértices de Steiner. Dado que por cada vértice de una clase cromática en la 3-coloración de la triangulación, siempre existe otro vértice de la misma clase cromática dentro de un mismo cuadrilátero de P, y todo cuadrilátero es un polígono en forma de estrella, entonces puedo conectar cualesquiera dos de los vértices de Steiner anteriores por medio de a lo más dos aristas. Por lo que a lo más son necesarias $m + 2 \cdot \lfloor \frac{n}{3} \rfloor$ aristas en un árbol mínimo en aristas para conectar al conjunto de punto Z en P. Para el lado izquierdo de la inecuación, basta con suponer que todos los puntos terminales son visibles entre sí. Para crear un árbol que una m puntos, se necesitan exactamente $m - 1$ aristas. □

Lema 2.7. *Dado un árbol de Steiner $SMT^* = (S, L)$ y un entero j^* que es el índice de una región de Steiner, el algoritmo 2.1.1 termina en tiempo $O(n^2 \log n)$.*

Demostración

Ya que el ciclo de la línea 8 es repetido una vez por cada vértice en S, este ciclo hace $O(m)$ iteraciones. Cada vez que un elemento entra a la cola, en la línea 15 se calcula una intersección de polígonos, y por [14, pág. 40], sabemos que esto toma $O(n \log n)$. Por lo tanto, el algoritmo 2.1.1 toma $O(n^2 \log n)$. □

Teorema 2.8. *Dado un polígono simple P de n vértices y un conjunto de terminales $Z = \{z_1, z_2, \ldots, z_m\}$ dentro de P, el algoritmo 2.1.2 calcula $SMT(Z)$ en tiempo $\theta(2^m nm(n+m))$.*

Demostración

El ciclo de la línea 1 hace $\binom{|Z|}{2} \in \theta(n^2)$ iteraciones. En la linea 2 calculamos MLD dentro del ciclo y por 1.14 sabemos que esto toma $O(n)$. Por el lema 2.6 sabemos que el ciclo de la linea 4 toma $O(n)$. Para la instrucción 5 sabemos que por [14, pág. 40] la intersección de polígonos toma $O(n \log n)$ por lo que el procesamiento del ciclo de la linea 1 toma $O(n + n^2 \log n)$

El algoritmo busca, para cada uno de los posibles $\theta(2^m)$ subconjuntos de Z de cardinalidad 3 o mayor, su árbol de Steiner. Para algún árbol de Steiner de cardinalidad i, se toma uno de los terminales y se calcula la distancia más corta en aristas de ese terminal a alguna región de un árbol de Steiner de cardinalidad $i-1$. Esto se hace para los $\theta(m)$ terminales. Como vimos en la sección 1.4, calcular la distancia más corta en aristas toma $\theta(n)$. Esto se hace para cada uno de los $\theta(n+m)$ vértices del árbol Steiner de cardinalidad $i-1$. □

Teorema 2.9. *Dado un polígono simple P de n vértices y un conjunto de terminales $Z = \{z_1, z_2, \ldots, z_m\}$ dentro de P, el algoritmo 2.1.2 calcula $SMT(Z)$ utilizando $\theta(2^m(n^2+m))$ en espacio.*

Demostración

Este algoritmo genera todos los $\theta(2^m) = \binom{m}{1}, \binom{m}{2}, \ldots, \binom{m}{m}$ árboles de Steiner y, al ser un algoritmo de programación dinámica tipo *Bottom-up*, es necesario que existan todos los árboles de Steiner de tamaño $i-1$ para generar todos los árboles de Steiner de tamaño i, por lo que el número de árboles de Steiner que existen simultáneamente en memoria está dado por:

$$\max_{1 \leq i \leq m} \left(\binom{m}{i-1} + \binom{m}{i} \right)$$

Este número está dado por el **coeficiente binomial central** [15]

$$\binom{2m}{m} = \frac{(2m)!}{(m!)^2} \leq \frac{4^m}{\sqrt{3m+1}}$$

Tomando en cuenta que nuestra combinación en realidad es $\binom{m}{\frac{m}{2}}$, sustituyendo m con $\frac{m}{2}$, obtenemos:

$$\binom{m}{\frac{m}{2}} \leq \frac{2^m}{\sqrt{\frac{3}{2}m+1}}$$

Este número nos indica cuantos árboles de Steiner hay simultáneamente. Ahora solo necesitamos calcular cuanto pesa cada árbol de Steiner. Sabemos que la intersección de dos polígonos de visibilidad tiene a lo más $\theta(n)$ vértices [14, pág. 40] y por el teorema 2.8 sabemos que un árbol de Steiner tiene $\theta(n)$ vértices de Steiner y m vértices terminales. Por lo tanto, cada árbol de Steiner pesa $\theta(n^2+m)$. Con esto podemos concluir que la complejidad en espacio del algoritmo 2.1.2 es de $\theta(2^m(n^2+m))$

□

2.3. Ejemplo de ejecución del algoritmo

En esta sección se dará un ejemplo del funcionamiento del algoritmo 2.1.2 paso a paso. A continuación se muestra el polígono con el que estaremos trabajando:

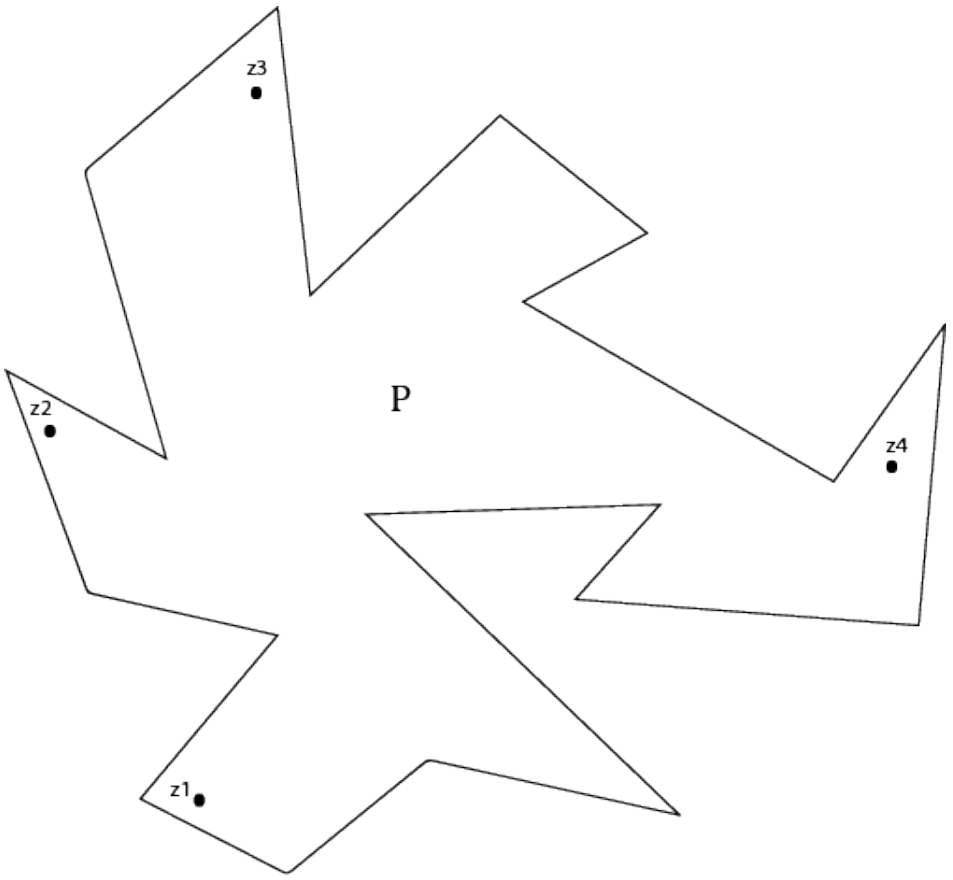
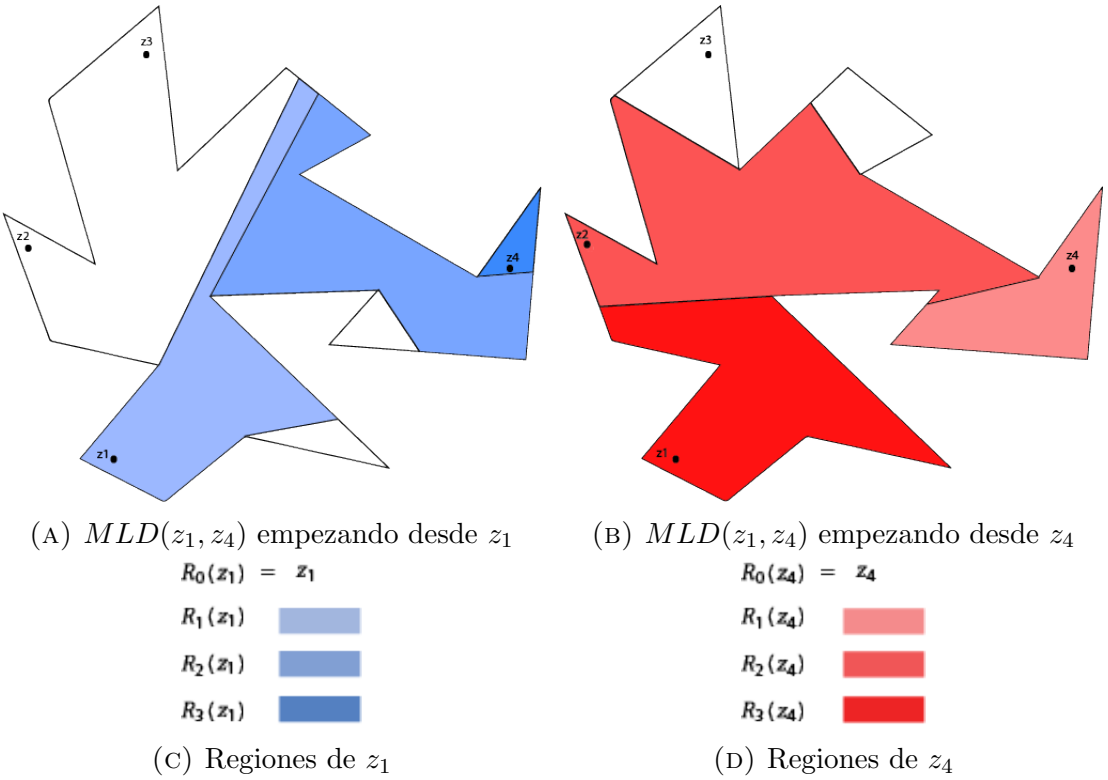


FIGURA 2.1: Polígono inicial

Empezaremos por calcular todos los árboles de Steiner mínimos en aristas de cardinalidad dos. Lo que es lo mismo que calcular todas las trayectorias mínimas en aristas entre todo par de puntos. Para ejemplificar este paso, calcularemos $SMT(z_1, z_4)$, que es equivalente a calcular $MLP(z_1, z_4)$. El objetivo de este paso es encontrar las regiones en donde podemos colocar los vértices de Steiner para conectar z_1 y z_4 mediante una trayectoria mínima en aristas. Para esto, se calcula la distancia mínima en aristas desde z_1 a z_4 y desde z_4 a z_1 (figura 2.2).

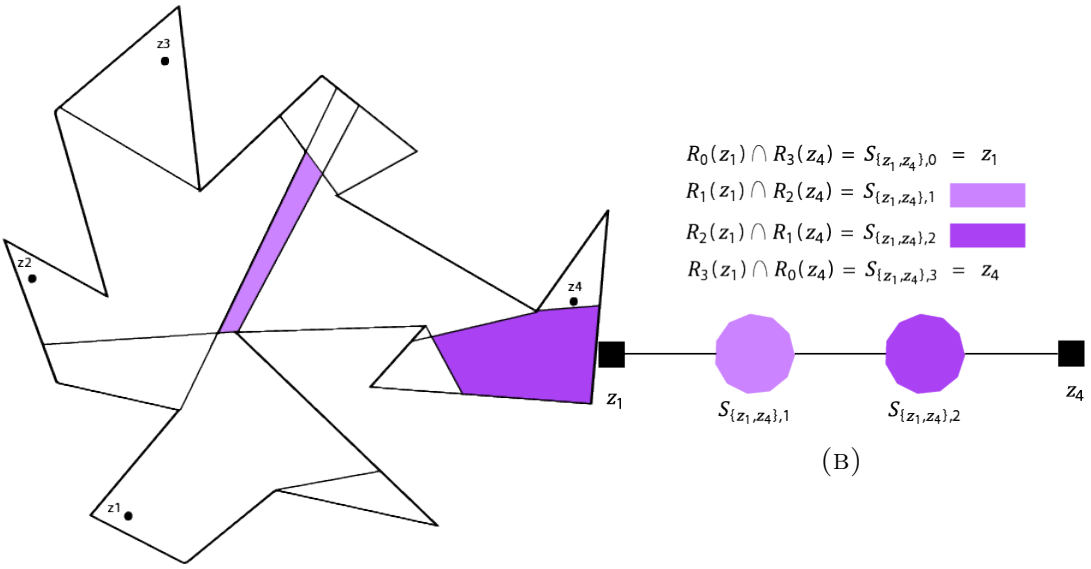


(A) $MLD(z_1, z_4)$ empezando desde z_1 (B) $MLD(z_1, z_4)$ empezando desde z_4

(C) Regiones de z_1 (D) Regiones de z_4

FIGURA 2.2: $MLD(z_1, z_4)$ Empezando desde z_1 (a) y z_4 (b) y almacenando todas las regiones $R_i(z_1)$ (c) y $R_i(z_4)$ (d).

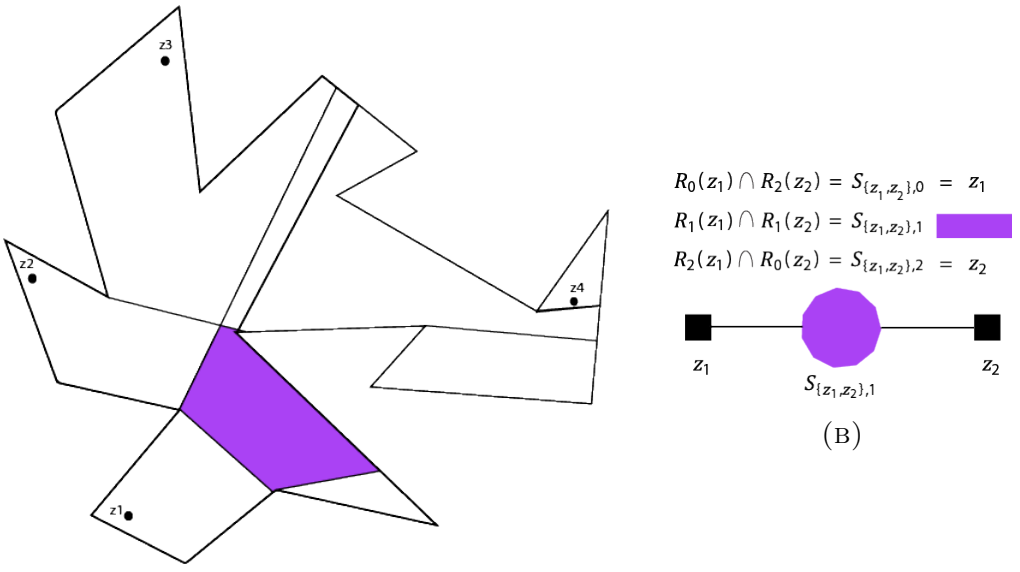
Ahora, para encontrar las regiones donde podemos colocar vértices Steiners, se calcula la intersección de las regiones $R_i(z_1)$ y las regiones $R_i(z_4)$ de la siguiente manera. Sea $j \leftarrow 0$ y $k \leftarrow MLD(z_1, z_4)$, intersectamos la región $R_j(z_1)$ con la región $R_k(z_4)$ y lo almacenamos en $S_{\{z_1,z_4\},j}$, aumentamos j en uno y disminuimos k en uno. Repetimos el proceso anterior hasta que $j = MLD(z_1, z_4)$ y $k = 0$ (figura 2.3).



(A) (B)

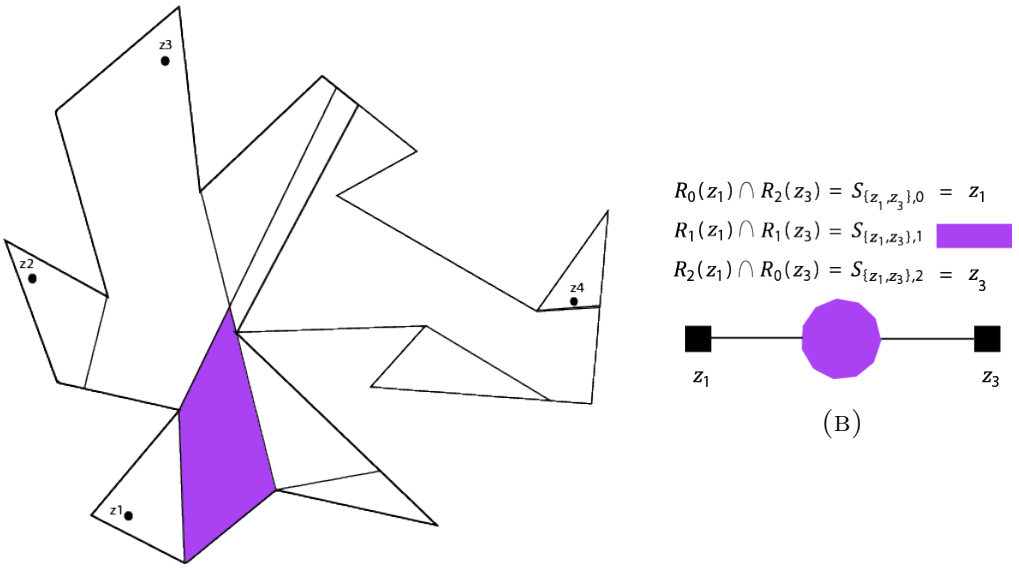
FIGURA 2.3: En (a) se puede apreciar la intersección de regiones descritas por (b). En (b) se muestra un árbol que describe a la lista de adyacencia de la relación entre las regiones encontradas.

Este proceso se hace para todo posible par de puntos: $\{z_1, z_2\}, \{z_1, z_3\}, \{z_1, z_4\}, \{z_2, z_3\}, \{z_2, z_4\}$ y $\{z_3, z_4\}$. A continuación se presenta el resultado del proceso anterior para todos los pares de puntos faltantes.



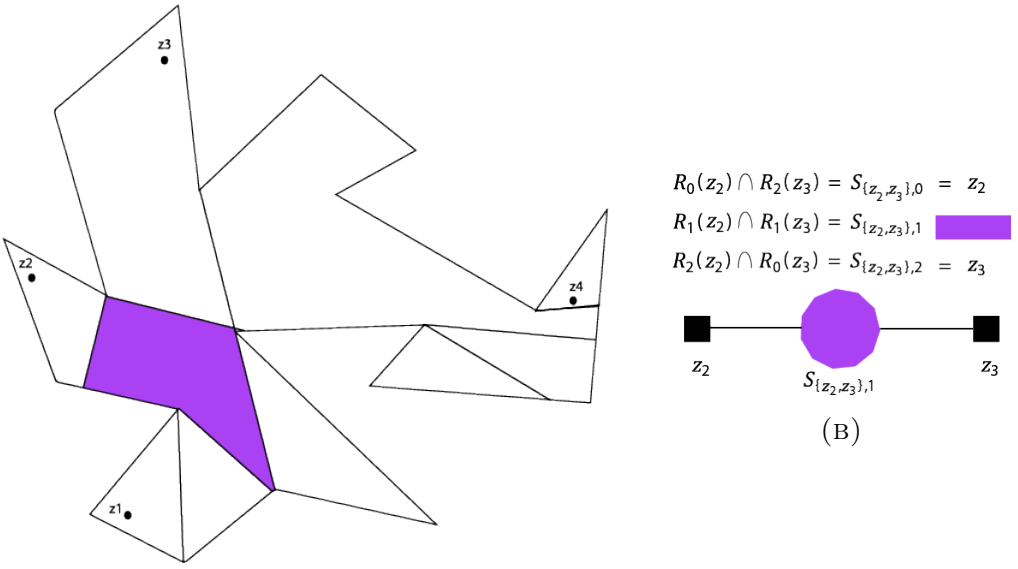
(A) (B)

FIGURA 2.4



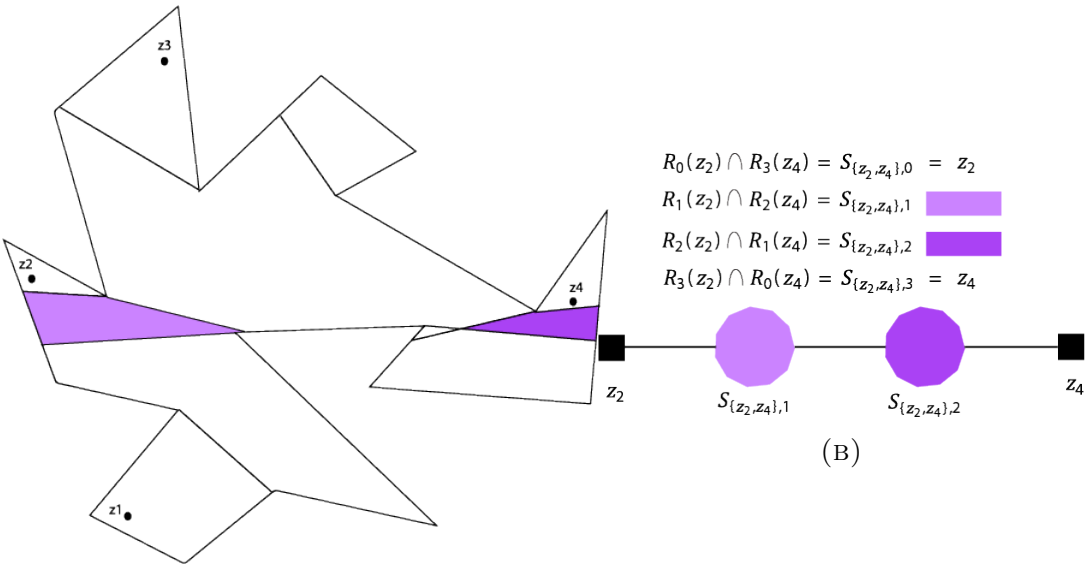
(A) (B)

FIGURA 2.5



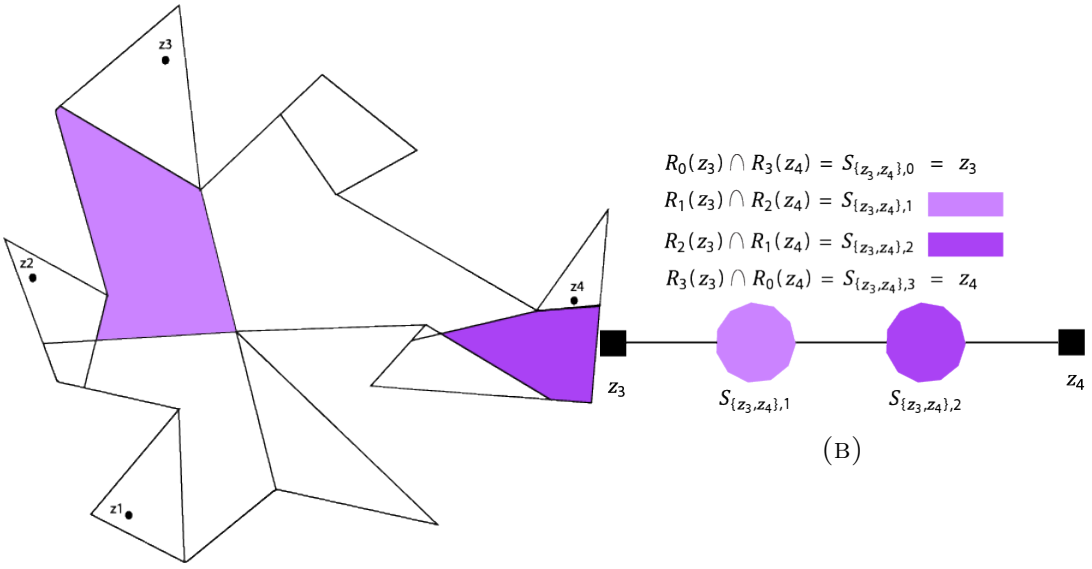
(A) (B)

FIGURA 2.6



(A) (B)

FIGURA 2.7



(A) (B)

FIGURA 2.8

Al terminar estos procesos, las siguientes listas de adyacencia se encuentran contenidas en memoria.

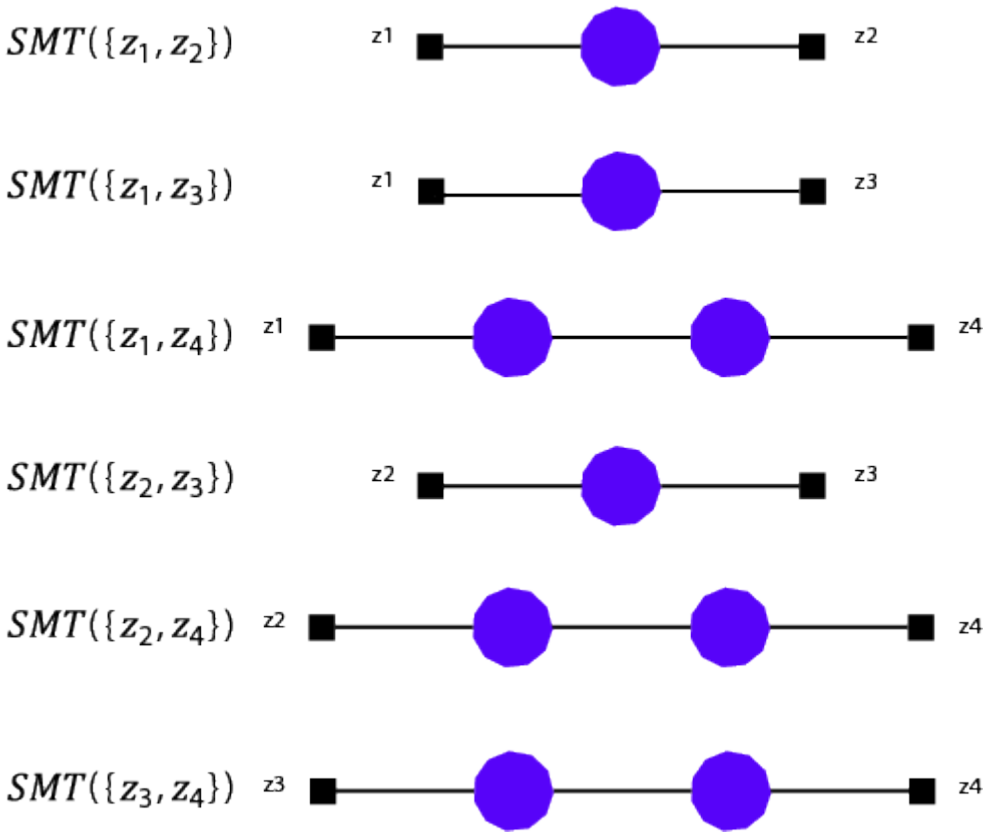


FIGURA 2.9

Una vez que todos los árboles de Steiner en aristas de cardinalidad dos están almacenados en memoria, el siguiente paso es encontrar todos los árboles de Steiner en aristas de cardinalidad tres, es decir, todos los árboles de Steiner en aristas para las tripletas $\{z_1, z_2, z_3\}$, $\{z_1, z_2, z_4\}$, $\{z_1, z_3, z_4\}$ y $\{z_2, z_3, z_4\}$. A continuación calcularemos $SMT(\{z_1, z_2, z_4\})$ para ejemplificar este proceso.

Como se mencionó al principio de este capítulo, nuestro algoritmo toma soluciones de problemas pequeños para resolver problemas más grandes. Para calcular $SMT(\{z_1, z_2, z_4\})$ se insertará z_1, z_2 y z_4 en $SMT(\{z_2, z_4\})$, $SMT(\{z_1, z_4\})$ y $SMT(\{z_1, z_2\})$ respectivamente.

Empezaremos insertando z_4 en $SMT(\{z_1, z_2\})$.

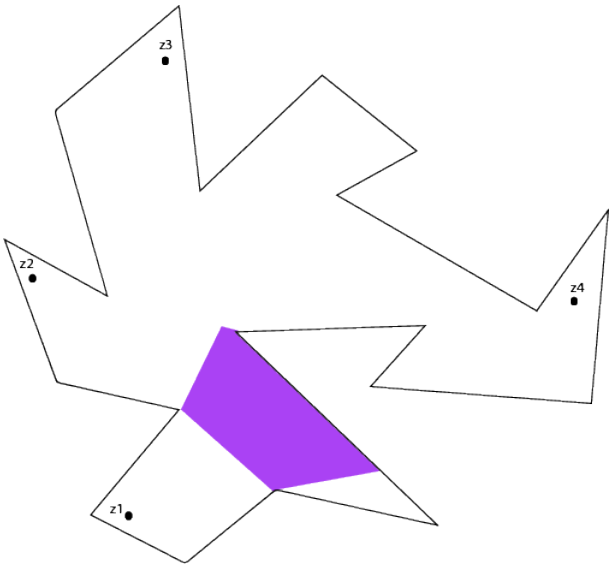


FIGURA 2.10: $SMT(\{z_1, z_2\})$.

Se calcula la distancia mínima en aristas entre z_4 y todas las regiones de $SMT(\{z_1, z_2\})$.

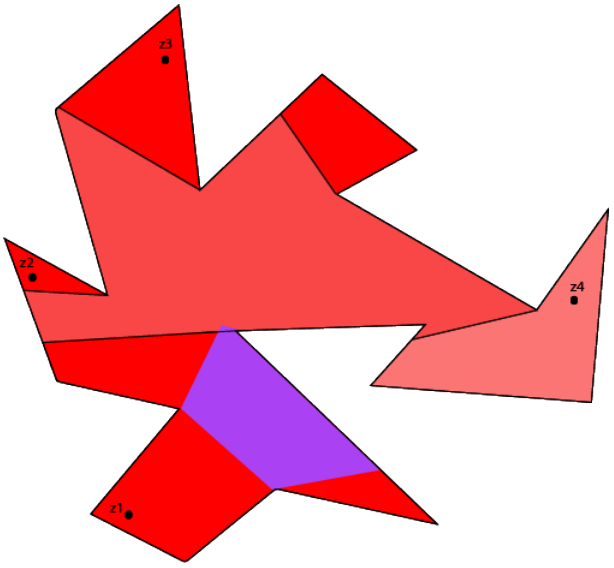


FIGURA 2.11: Visibilidad de z_4

Se escoge una región R_i de $SMT(\{z_1, z_2\})$ la cual minimice $MLD(R_i, z_4)$ y se calcula la intersección de esa región R_i y el polígono de visibilidad de z_4 con el que se encontró R_i, en este caso es $R_2(z_4)$

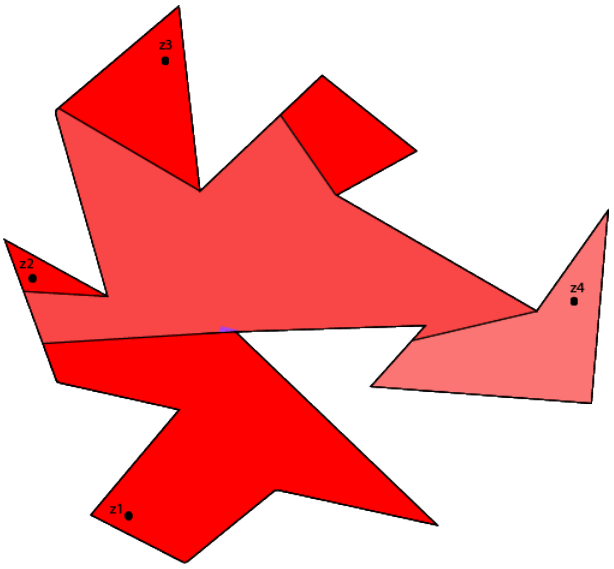


FIGURA 2.12: Intersección entre $R_2(z_4)$ y la región más cercana de $SMT(\{z_1, z_2\})$.

Llamemos a la región resultante del paso anterior R_i^*. Ahora calcularemos $MLP(R_i^*, z_4)$ y calcularemos las intersecciones de las regiones de visibilidad de $MLP(R_i^*, z_4)$ y $MLP(z_4, R_i^*)$ (ver figuras 2.2 y 2.3).

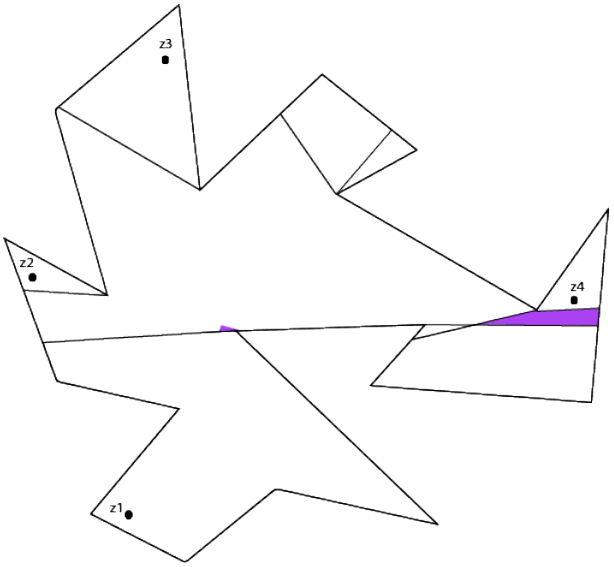


FIGURA 2.13: Visibilidad del polígono descrito en la figura anterior y su conexión con z_4

Así obtenemos un árbol de Steiner para $\{z_1, z_2, z_4\}$ de cinco regiones.

Para las siguientes figuras se muestra el mismo procedimiento para calcular un

árbol de Steiner para $\{z_1, z_2, z_4\}$ a partir de $SMT(\{z_1, z_4\})$ y $SMT(\{z_2, z_4\})$.

Un árbol de Steiner a partir de $SMT(\{z_1, z_4\})$:

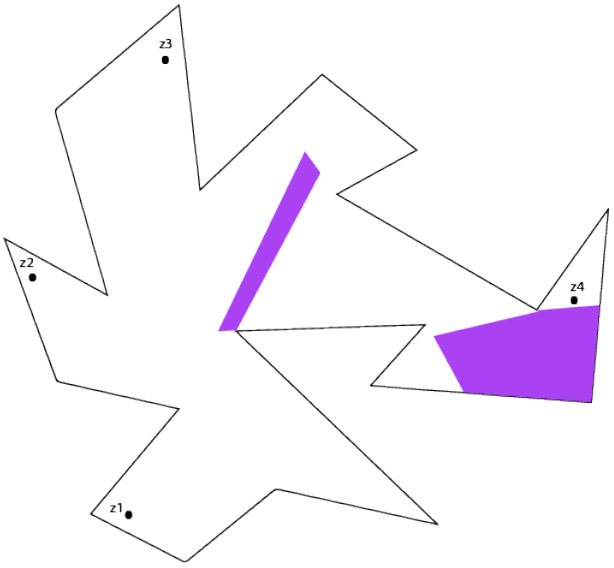


FIGURA 2.14: $SMT(\{z_1, z_4\})$.

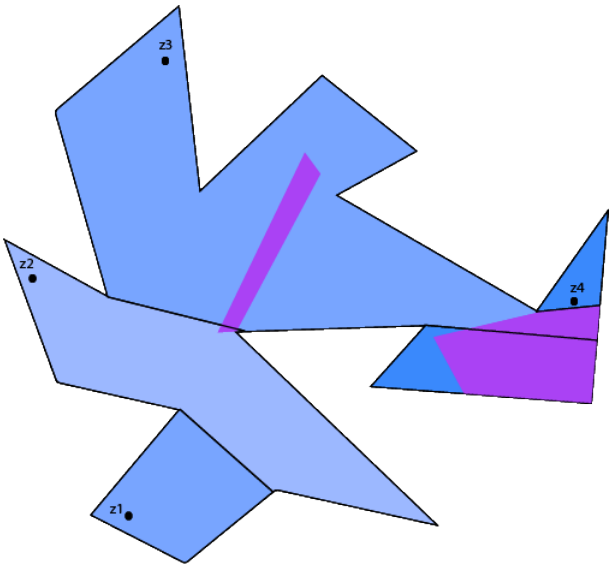


FIGURA 2.15: Visibilidad de z_2

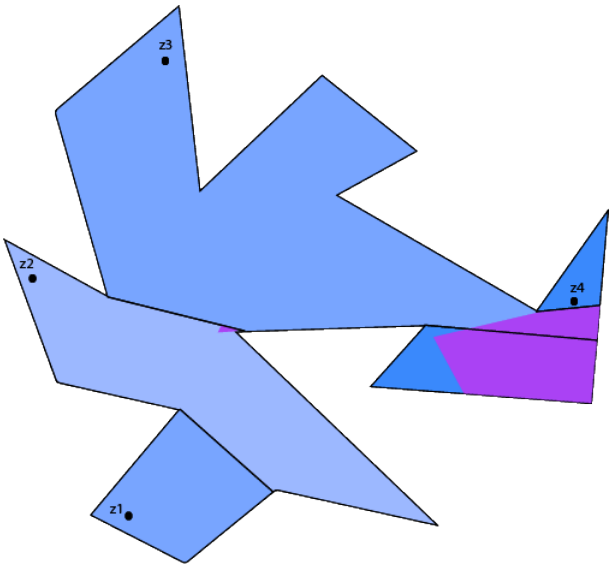


FIGURA 2.16: Intersección entre $R_1(z_2)$ y la región más cercana de $SMT(\{z_1, z_2\})$.

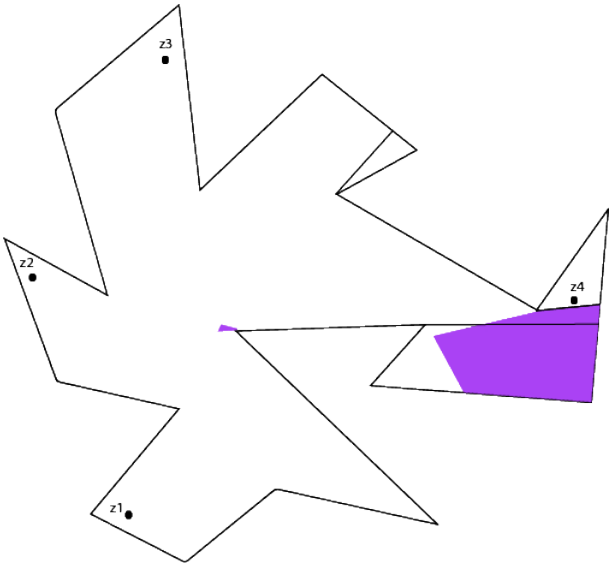


FIGURA 2.17: Visibilidad del polígono descrito en la figura anterior y su conexión con z_2

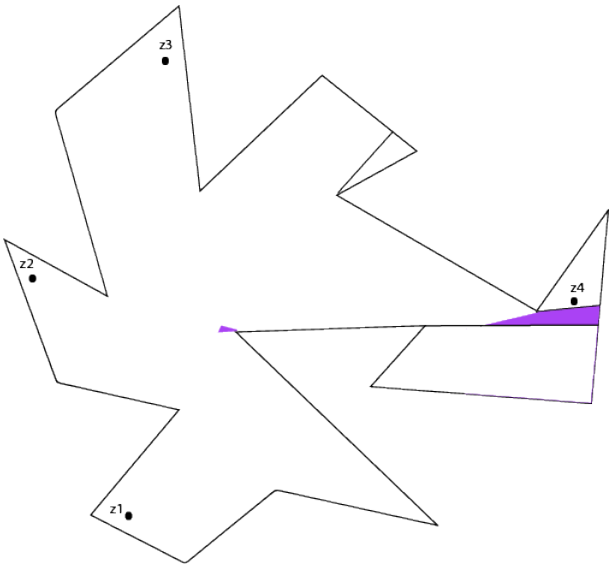


FIGURA 2.18: Reducción de los polígonos vecinos de la región seleccionada.

Se obtiene un árbol de Steiner para $\{z_1, z_2, z_4\}$ de cinco regiones.

Un árbol de Steiner a partir de $SMT(\{z_2, z_4\})$:

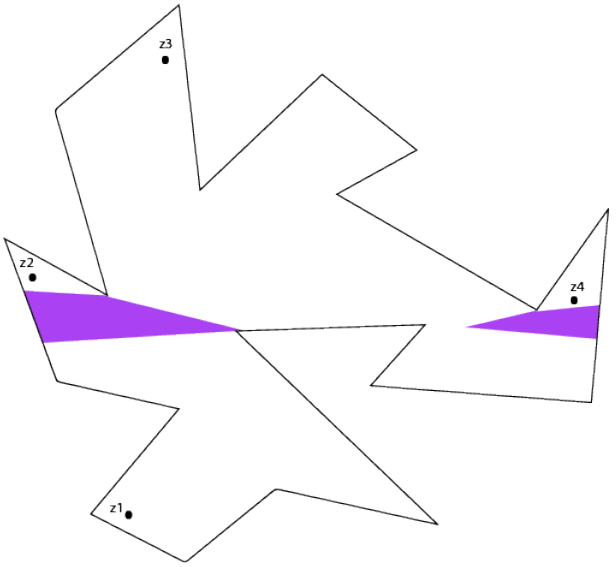


FIGURA 2.19: $SMT(\{z_2, z_4\})$.

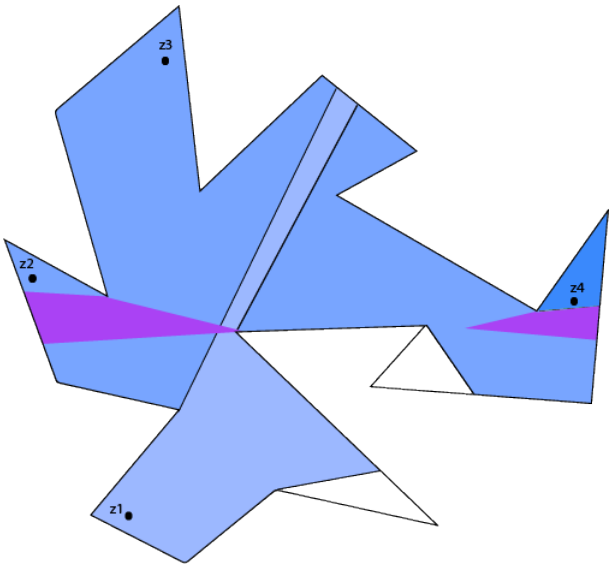


FIGURA 2.20: Visibilidad de z_1

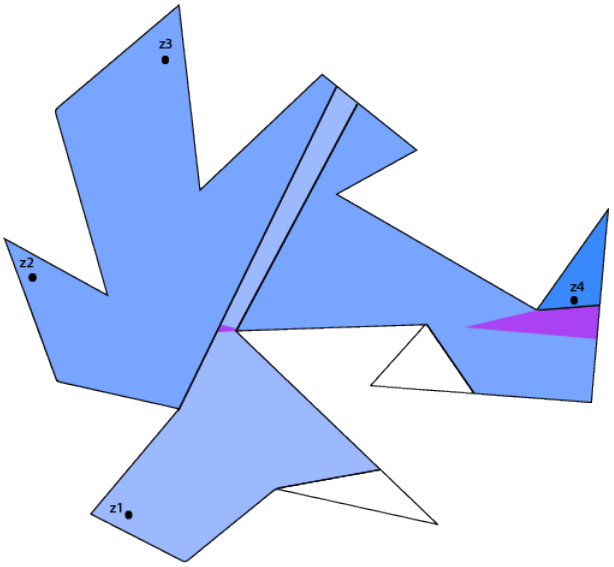


FIGURA 2.21: Intersección entre $R_1(z_1)$ y la región más cercana de $SMT(\{z_2, z_4\})$.

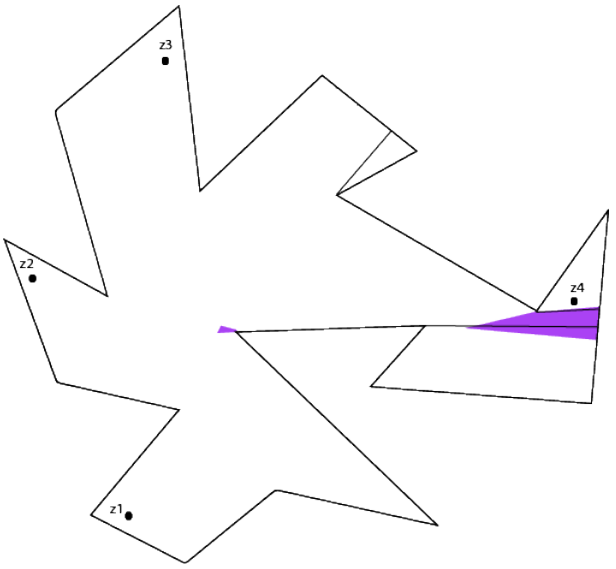


FIGURA 2.22: Visibilidad del polígono descrito en la figura anterior y su conexión con z_1

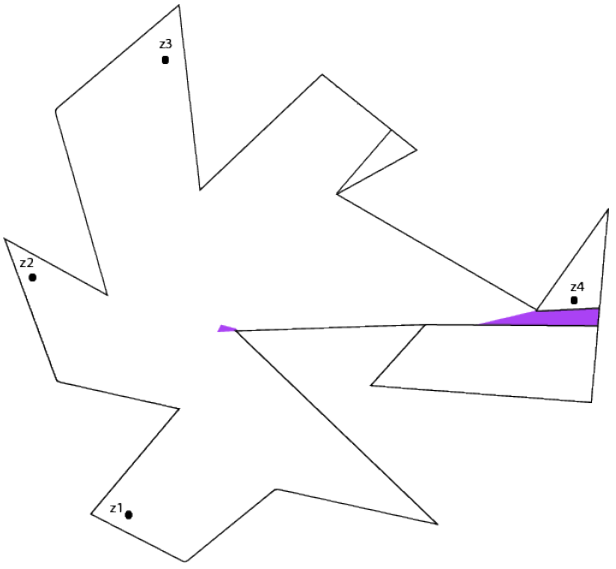


FIGURA 2.23: Reducción de los polígonos vecinos de la región seleccionada.

Se obtiene un árbol de Steiner para $\{z_1, z_2, z_4\}$ de cinco regiones.

Para obtener $SMT(\{z_1, z_2, z_4\})$ a partir de los tres árboles anteriores, se tiene que escoger el árbol con el menor número de regiones, como fue mencionado en la recurrencia del lema 2.4. Podemos observar que en este ejemplo, los tres árboles de Steiner tienen el mismo número de regiones, por lo que cualquiera de los tres

es un buen candidato para ser $SMT(\{z_1, z_2, z_4\})$.

Aplicando el mismo proceso para calcular $SMT(\{z_1, z_2, z_3\})$, $SMT(\{z_1, z_3, z_4\})$ y $SMT(\{z_2, z_3, z_4\})$, obtenemos los siguientes árboles de Steiner.

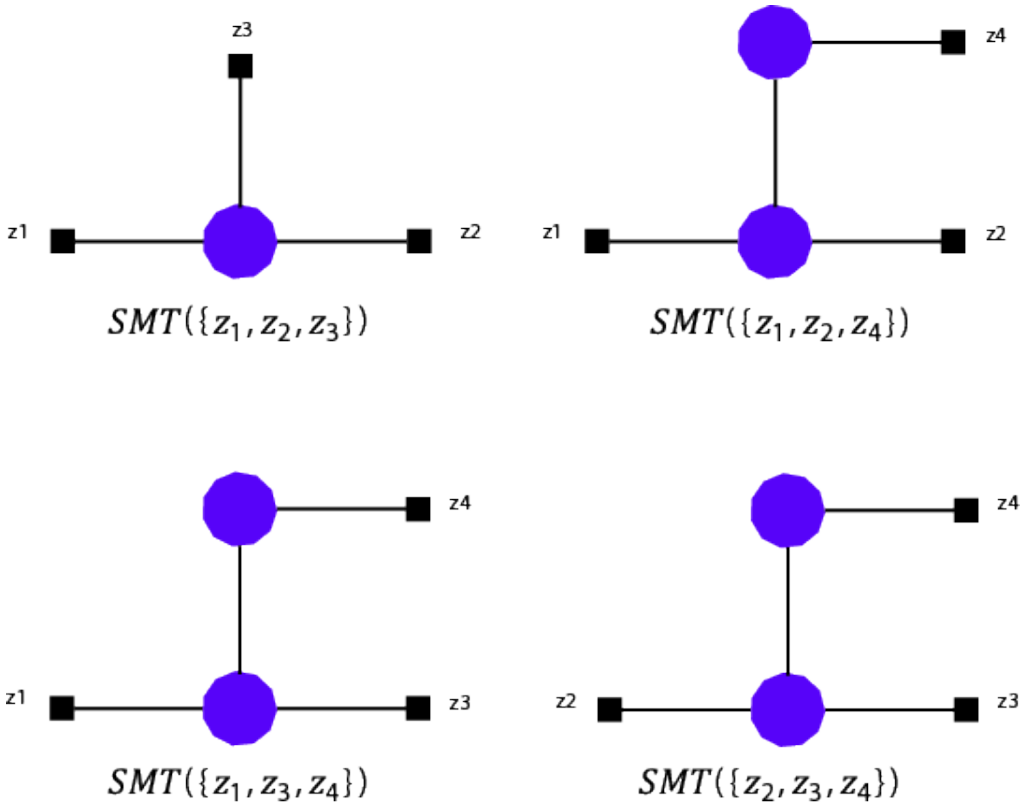
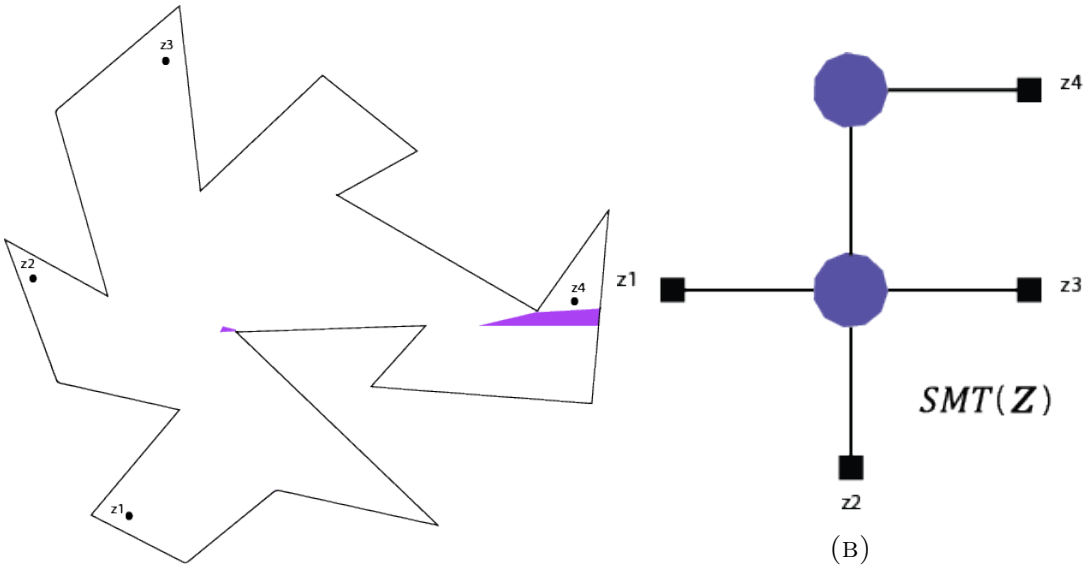


FIGURA 2.24

Repitiendo el proceso que se hizo a partir de la figura 2.9 sobre $SMT(\{z_1, z_2, z_3\})$, $SMT(\{z_1, z_2, z_4\})$, $SMT(\{z_1, z_3, z_4\})$ y $SMT(\{z_2, z_3, z_4\})$ obtenemos $SMT(Z)$, el cual está formado por cinco aristas y seis regiones.



(A) (B)

FIGURA 2.25

Capítulo 3

Algoritmo de aproximación para encontrar un árbol de Steiner mínimo en gráficas

Como su nombre lo indica, un algoritmo de aproximación no busca encontrar la solución óptima de un problema. En vez de eso, busca encontrar una solución que no es "muy lejana" al valor de la solución óptima. Una medida utilizada para medir la *calidad* del algoritmo de aproximación es la razón máxima entre el valor regresado por el algoritmo y el valor de la solución óptima, donde el máximo es tomado de todas las posibles instancias del problema de optimización. Esta razón es llamada *razón de aproximación.*

El algoritmo de aproximación que veremos en este capítulo es la generalización del algoritmo de A. Matsuyama y H. Takahashi [16] el cual consiste en calcular el árbol de expansión mínima de la gráfica de distancias del conjunto de terminales K. Una gráfica de distancias de un subconjunto de vértices $K \subset V$ es una gráfica completa de $|K|$ vértices en la que los pesos de una arista es igual al peso de la trayectoria más corta entre dos vértices $v, w \in K$. Este algoritmo tiene una razón de aproximación de 2.

Podemos ver a la trayectoria más corta entre dos terminales como el árbol de Steiner de esos dos vértices. Esta observación motiva la siguiente generalización: En vez de sólo usar árboles de Steiner mínimos para todos los pares de vértices, también podemos utilizar árboles de Steiner mínimos para todos los posibles tríos de vértices o, incluso, usar árboles de Steiner mínimos para todos con subconjuntos de K de tamaño r. Ahora el problema se reduce a encontrar un árbol de expansión mínima en una *hipergráfica*.

3.1. Árbol de expansión mínima en hipergráficas

Sea $N = (V, E, l; K)$ una gráfica. Con respecto a esto definimos para cualquier $r \geq 2$ una hipergrafica r-acotada ponderada $H_r(N) = (K, E_r, l_r)$ en el conjunto de vértices K de la siguiente manera. El conjunto de hiperaristas E_r consiste en todos los subconjuntos de K de tamaño a lo más r, el peso $l_r(e)$ de una hiperarista $e \in E_r$ consiste en el peso del SMT de esa hiperarista e en la gráfica N. La hipergráfica $H_r(N)$ es llamada la hipergráfica de r-distancia de N. Por simplicidad, denotaremos $mst_r(N)$ a $mst(H_r(N))$.

Lema 3.1. *Sea $N = (V, E, l; K)$ una gráfica con el conjunto de terminales K y sea $H_r(N)$ como fue definido arriba, entonces:*

$$mst_r(N) \geq smt(N).$$

Demostración

Sea T un MST en $H_r(N)$. Para cada hiperarista e en T escogemos un SMT T_e para e en N. El hecho de que T sea un árbol de expansión en $H_r(N)$ implica que $\bigcup_{e \in T} T$ es una subgráfica conectada de (V, E) que contiene todos los vértices en K. □

Lema 3.2. *Sea r una constante fija. Entonces la hipergráfica de r-distancia $H_r(N)$ de una gráfica $N = (V, E, l; K)$ puede ser computado en $O(n^2 log(n) + nm + k^{r+1} n^2)$.*

Demostración

Para poder construir $H_r(N)$ necesitamos calcular un SMT por cada subconjunto de K de tamaño a los más r. Esos son $\sum_{i=2}^{r} \binom{k}{i} \leq \sum_{i=2}^{r} \frac{k^i}{i!} \leq k^{r+1}$ subconjuntos. Para cada uno de estos subconjuntos calculamos el SMT con el algoritmos de Dreyfus-Wagner [17]. La inicialización del algoritmo tarda $O(n^2 log(n) + nm)$. Los demás pasos del algoritmo Dreyfus-Wagner requieren tiempo $O(2^r n^2 + 3^r n) = O(n^2)$ tomando en cuenta que r es constante. Como la inicialización se calcula una sola vez, podemos calcular el SMT para cada conjunto de K de tamaño a los más r en tiempo $O(n^2 log(n) + nm + k^{r+1} n^2)$. □

El lema 3.2 implica que si mostramos para una r fija que (1) Los MST en $H_r(N)$ pueden ser determinados en tiempo polinomial y (2) Los MST tienen la propiedad de que no son mucho más grandes que los SMT, entonces podemos encontrar un buen algoritmo de aproximación. Sea p_r la cota superior de la razón $mst_r(N)/smt(N)$ para todas las gráficas $N = (V, E, l; k)$. Los valores de p_r son llamados "Razones de Steiner". Por el lema 3.1 sabemos que $p_r \geq 1$. Es conocido que $p_2 = 2$. [16]

3.2. Razones de Steiner

A continuación se mostrará una cota inferior y una superior para las razones de Steiner, las cuales nos ayudarán a encontrar un valor para p_r.

3.2.1. Cota inferior para p_r

Considera un espacio métrico particular M_n basado en un árbol ponderado B_n. Sea B_n un árbol binario completo con n niveles de aristas y un nivel de aristas extra donde cada vértice interno sólo tiene un hijo. Las aristas en el nivel $1 \leq i \leq n$ tienen peso 2^{n-i} y las aristas del último nivel, el nivel $n + 1$, tienen peso 1. Ver figura 3.1.

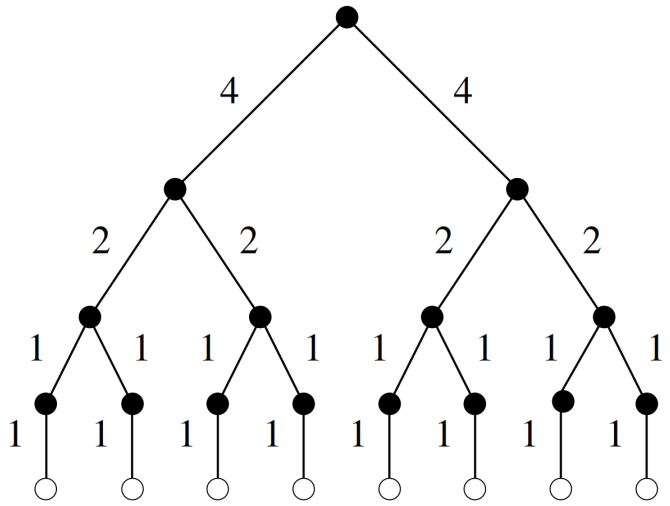


FIGURA 3.1

Los vértices de M_n son los vértices de B_n, la distancia entre dos vértices es la longitud de la trayectoria más corta entre ellos en B_n. Podemos ver que las aristas entre dos puntos en M_n son trayectorias entre esos puntos en B_n.

Sean las hojas de B_n el conjunto K de vértices terminales. Vamos a calcular el límite de la razón $\frac{mst_r(N)}{smt(N)}$ con $n \to \infty$. Esto nos dará una cota inferior para p_r.

Lema 3.3. *La longitud del árbol de Steiner mínimo para K es la longitud total de B_n, esto es, $smt(N) = (n+1)2^n$.*

Demostración

El árbol B_n interconecta a K y es claro que todas las aristas de B_n deben de ser usadas para interconectar K. Por lo tanto, B_n ya es el árbol más pequeño que interconecta a K.

La suma de los pesos de las aristas de B_n en cualquier nivel de aristas de B_n es de 2^n. Esto es cierto para los últimos dos niveles de B_n, siendo que $|K| = 2^n$. Para cada nivel, excepto los últimos dos, el nivel de abajo tiene el doble de cantidad de aristas que el nivel de arriba, pero cada arista con la mitad del peso de la aristas del nivel de arriba. Como hay $n+1$ niveles, entonces $smt(N) = (n+1)2^n$. □

Lema 3.4. *Existe un árbol de Steiner mínimo r-acotado para K donde cada vértice de Steiner es de grado 3.*

Demostración

Los vértices terminales de un componente en cualquier árbol de Steiner r-acotado pueden ser interconectados por árboles binarios embebidos en B_n, y por el lema 3.3, ése es el árbol de longitud mínima que interconecta esos terminales. Los vértices de Steiner de longitud 2 puedes ser reemplazados por una arista con peso igual a la suma de los pesos que tienen las aristas adyacentes a ese vértice. Por lo tanto, podemos suponer que todos los vértices son de grado 3. Estos componentes son usados para construir el árbol de Steiner r-acotado deseado. □

Podemos pensar que un componente en un árbol de Steiner r-acotado para K como un árbol binario regular con a los más r hojas embebido en B_n, donde las aristas son trayectorias en B_n. Las raíces de estos componentes son los vértices de Steiner de grado 2 que están en el nivel más alto; nos referiremos a estos vértices como *raíces componentes*.

Lema 3.5. *Existe un árbol de Steiner mínimo r-acotado para K donde cada vértice de Steiner es de grado 3 y donde los vértices de B_n no son usados más de una vez como vértices de Steiner o raíces componentes.*

Demostración

Considerese un árbol de Steiner mínimo r-acotado T descrito como en el lema 3.4. Sea v un vértice en el nivel más alto de B_n que es usado como raíz componente o vértice de Steiner de dos componentes, A y B, en T. Sea u y w los hijos de v. Las aristas vu y vw son usadas en ambos componentes A y B. Sean A_1 y B_1 partes de los componentes A y B, respectivamente, que están debajo de u, y sean A_2 y B_2 las partes que están debajo de w. Ver figura 3.2.

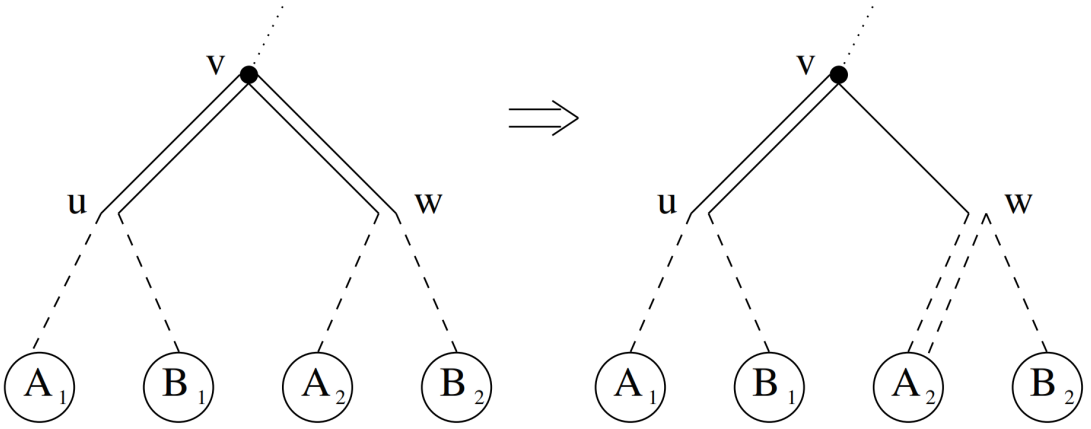


FIGURA 3.2

Los componentes A y B deben de estar conectados por alguna trayectoria en T. pero como esto es un árbol, la trayectoria no puede ir por B_1 y B_2 a la vez. Supongamos que no va por B_2.

Removemos la arista vw del componente B. Hacemos a B_2 un componente separado, y conectamos B_2 al componente A mediante una trayectoria a un terminal de A_2 (una hoja en B_n). Ver figura 3.2.

Obsérvese que en B_n la longitud de la arista vw es la misma que la longitud de la trayectoria de w a un terminal. Esto sucede porque la longitud de una arista tiene el doble de longitud que una arista en el nivel inferior. Por lo tanto, este cambio no modifica la longitud del árbol de Steiner, tampoco aumenta la longitud de ningún componente ni agrega ningún vértice de Steiner, así que todos los vértices de Steiner son de grado 3. El vértice w se convirtió en la raíz de un nuevo componente. Quizá hemos agregado vértices que son usados como vértices de Steiner o raíces componentes en más de un componente, sin embargo, sólo los agregamos debajo de v.

Por inducción, si repetimos este proceso sucesivamente podemos remover los vértices de Steiner o raíces componentes agregadas por pasos anteriores que se traslapan en vértices de B_n de nivel en nivel yendo hacía abajo hasta que no haya dichos vértices sobrantes. □

Lema 3.6. *Existe un árbol de Steiner mínimo r-acotado para K donde cada vértice de Steiner es de grado 3 y donde los vértices de B_n, excepto en el nivel inferior, es usado exactamente una vez un como vértice de Steiner o como raíz de algún componente.*

Demostración

Por el lema 3.5 sabemos que existe un árbol de Steiner mínimo r-acotado T donde cada vértice interno de B_n es usado a lo más una vez. El nivel más bajo de los vértices interiores no pueden ser raíz de componentes o vértices de Steiner porque sólo tienen un hijo. Mostraremos que se necesitan $2^n - 1$ vértices de Steiner y raíces componentes para interconectar todos los 2^n vértices de K y ya que hay $2^n - 1$ vértices internos de B_n (sin contar al nivel más bajo de vértices internos), cada vértice debe de usarse exactamente una vez.

Removemos las raíces componentes y los vértices de Steiner de T uno por uno. Cuando una raíz componente es removida, separamos el árbol en dos piezas. Cuando removemos un vértice de Steiner, separamos el árbol en dos piezas desconectando uno de los dos hijos del vértice de Steiner. Ya que todos los vértices internos (excepto el último nivel de vértices internos) son removidos, los 2^n vértices terminales deben de estar completamente desconectados. Ya que remover un vértice agrega una nueva pieza, debimos haber removido $2^n - 1$ vértices para crear los 2^n piezas desconectadas. □

Teorema 3.7. *Para cualquier r, con $r = 2^s + t$, donde $0 \leq t < 2^s$, tenemos que:*

$$p_r \geq \frac{(s+1)2^s + t}{s2^s + t} \tag{3.1}$$

Demostración

Sea $T_{r,n}$ un árbol de Steiner mínimo r-acotado que satisface el lema 3.6. Podemos tomar un componente C en $T_{r,n}$ como un árbol binario regular, los vértices internos de este árbol son raíces componentes o vértices de Steiner. Debajo de cada uno de estos vértices internos hay dos aristas en B_n. Todos los vértices de Steiner tienen dos aristas debajo por el lema 3.4, y las raíces componentes deben de tener dos

aristas debajo de ellos en un árbol mínimo. Nombremos a estas aristas *Aristas pico* de C y denotémoslas por P_C. Nombremos al resto de las aristas usadas en C como *Aristas de conexión* de C y denotémoslas por C_c. Denotemos todas las aristas pico de $T_{r,n}$ por $P_{r,n}$ y todas las aristas de conexión por $C_{r,n}$. Cuando nos referimos a un pico, se hace referencia a las dos aristas pico que tienen un vértice en común.

Queremos mostrar que

$$\sum_{e \in C_c} l(e) \geq \frac{2^s}{s2^s + t} \cdot \sum_{e \in P_c} l(e) \tag{3.2}$$

Si esto es cierto, entonces sumando sobre todos los componentes de $T_{r,n}$ tenemos

$$\sum_{e \in C_{r,n}} l(e) \geq \frac{2^s}{s2^s + t} \cdot \sum_{e \in P_{r,n}} l(e) \tag{3.3}$$

Supongamos que la inecuación 3.3 es cierta. Por el lema 3.5 los vértices de Steiner y las raíces componentes cubren todos los vértices internos, excepto por el último nivel de vértices internos. Entonces las aristas pico cubren todas las aristas de B_n, excepto el último nivel. La suma de las aristas del último nivel de B_n es 2^n y la suma de las longitudes de todas las aristas es $smt(N)$ (ver lema 3.3). Entonces

$$smt(N) = 2^n + \textstyle\sum_{e \in P_{r,n}}.$$

por la inecuación 3.3,

$$\begin{aligned} mst_r(N) &= \sum_{e \in P_{r,n}} l(e) + \sum_{e \in C_{r,n}} l(e) \\ &\geq \left(1 + \frac{2^s}{s2^s + t}\right) \cdot \sum_{e \in Pr,n} \\ &= \left(\frac{(s+1)2^s + t}{s2^s + t}\right) \cdot (smt(N) - 2^n) \end{aligned}$$

Entonces

$$\begin{aligned} p_r &\geq \frac{mst_r(N)}{smt(N)} \\ &\geq \frac{\frac{(s+1)2^s+t}{s2^s+t}(smt(N)-2^n)}{smt(N)} \\ &= \frac{\frac{(s+1)2^s+t}{s2^s+t}(n+1)2^n}{(n+1)2^n} - \frac{\frac{(s+1)2^s+t}{s2^s+t}2^n}{(n+1)2^n} \\ &= \frac{(s+1)2^s+t}{s2^s+t} - \frac{(s+1)2^s+t}{(s2^s+t)(n+1)} \end{aligned}$$

Siendo que p_r es el máximo de $\frac{mst_r(N)}{smt(N)}$ y que $mst_r(N) \geq smt(N) = (n+1)2^n$. Haciendo que $n \to \infty$ obtenemos una cota inferior para p_r, la inecuación 3.1.

Para terminar la prueba de este teorema, debemos de establecer la inecuación 3.2. Esta inecuación depende en que $r = 2^s + t$, pero el componente C puede ser de tamaño r o más pequeño. probaremos eso para todo componente C en un árbol r-acotado para toda r. Es fácil ver que para $r' \leq r$ y $r' = 2^{s'} + t'$, donde $0 \leq t' < 2^{s'}$, tenemos:

$$\frac{s'2^{s'}+t'}{2^{s'}} \leq \frac{s2^s+t}{2^s}$$

Por esto podemos decir que la inecuación 3.2 es cierta para cualesquiera componentes menores a r en un árbol de Steiner k-acotado. Ahora sólo falta comprobarlo para las componentes de tamaño exactamente r.

Por el lema 3.4, podemos sponer que todos los vértices son de grado 3, así que podemos ver a un componente como un árbol binario regular.

Probaremos la inecuación 3.2 para un componente de tamaño r por inducción sobre la suma de la profundidad de los picos desde la raíz componente. Calcularemos la profundidad de los picos contando aristas de B_n, no de M_n, y las contaremos ignorando su peso. La suma de la profundidad de los picos es la mínima cuando

los picos forman un árbol binario completo, excepto por el último nivel, así que inicialmente trataremos el primer caso.

En un componente C de tamaño r el cual sus aristas pico forman un árbol binario completo, excepto quizá en el último nivel. Tenemos 2^s aristas pico en el segundo nivel más bajo y $2t$ aristas pico en el nivel más bajo. Supongamos que las aristas pico del nivel más bajo tiene peso w. Entonces la suma de los pesos de las aristas del nivel más bajo es de $2tw$ y en los s niveles más arriba la suma de los pesos de cada nivel es de $2^{s+1}w$. Entonces tenemos

$$\sum_{e \in P_C} l(e) = 2w(s2^s + t)$$

Las aristas de conexión de C forman una trayectoria desde los picos en los niveles más bajos hasta los vértices terminales de B_n. Por la construcción de B_n, esas trayectorias tienen el mismo peso que las demás trayectorias que empiezan en su mismo nivel. Hay $2t$ trayectorias de peso w en el nivel más bajo y hay $2^s - t$ trayectorias de longitud $2w$ un nivel más arriba, Por lo tanto,

$$\sum_{e \in C_C} l(e) = 2w \cdot 2^s$$

En éste caso, la inecuación 3.2 se satisface. De hecho, es una igualdad.

A continuación, mostraremos que la razón de la longitud de las aristas de conexión decrece cuando la suma de profundidad de las aristas pico decrece. De hecho, si cambiamos la forma del componente moviendo un pico hacia arriba, la longitud de la suma de las aristas de conexión se mantiene constante, mientras que el peso de la suma de las aristas pico incrementa. Cuando los picos ya no pueden subir más niveles hacia arriba, la suma de la profundidad de las aristas pico es la mínima, y como ya demostramos, la inecuación 3.2 es cierta. Por lo tanto, es cierto para cualquier forma del componente C.

□

3.2.2. Cota Superior para p_r

La prueba para la cota superior seguirá el mismo lineamiento que la de la cota inferior. Primero convertimos el árbol de Steiner en un árbol binario regular ponderado. Después etiquetaremos los vértices de este árbol binario y con base en estas etiquetas, construimos $s2^s + t$ árboles de Steiner r-acotados y mostraremos que uno de esos árboles es suficientemente pequeño como para darnos una cota inferior.

Lema 3.8. *Para cualquier árbol binario, existe un mapeo de uno a uno f de un vértice interno a las hojas del árbol tal que:*

(a) Para cualquier vértice interno u, $f(u)$ es un descendiente de u.

(b) Todos las trayectorias $p(u)$ de u a $f(u)$ son disjuntas en aristas.

Demostración

Primero agregamos el siguiente requerimiento adicional:

(c) Existe una hoja v tal que la trayectoria de la raíz a v es disjunta en aristas de todas las demás trayectorias $P(u)$.

Se demostrará por inducción sobre la altura del árbol que las tres condiciones son ciertas. Cuando la altura es 0, esto es trivialmente cierto.
Sea T un árbol de altura $d \geq 1$. T tiene dos subárboles, T_1 y T_2, cada uno con altura de a lo más $d-1$ t sus raíces son los dos hijos de T. Por inducción, existen las funciones f_1 y f_2 en los vértices terminales de T_1 y T_2 que satisfacen (a) y (b). También existen vértices v_1 y v_2 en T_1 y T_2, respectivamente, que satisfacen (c). Se define f como la unión de f_1 y f_2 y definimos $f(\text{raíz de } T) = v_1$. Claramente, f satisface (a) y (b) y v_2 satisface (c) para T. Esto completa la inducción. □

Teorema 3.9. *Para cualquier r, con $r = 2^s + t$ donde $0 \leq t < 2^s$, tenemos*

$$p_r \leq \frac{(s+1)2^s + t}{s2^s + t} \tag{3.4}$$

Demostración

Podemos reescribir la inecuación de la siguiente manera

$$\frac{mst_r(N)}{smt(N)} \leq \frac{(s+1)2^s + t}{s2^s + t} \tag{3.5}$$

Demostraremos esto por inducción sobre n, el números de terminales en P. Si $n \leq r$, entonces la inecuación es cierta ya que $mst_r(N)/smt(N) = 1$. Para $n > r$, se considera el árbol de Steiner mínimo T en K. Si T no es un árbol de Steiner completo, entonces podemos separar el árbol en dos árboles de Steiner más pequeños desde un vértice terminal interno, cada uno con menos de n vértices terminales. Digamos que el conjunto de vértices de cada árbol es N_1 y N_2. Entonces tenemos $smt(N) = smt(N_1) + smt(N_2)$ y $mst_r(N) \leq mst_r(N_1) + mst_r(N_2)$, y entonces

$$\begin{aligned}
\frac{mst_r(N)}{smt(N)} &\leq \frac{mst_r(N_1) + mst_r(N_2)}{smt(N_1) + smt(N_2)} \\
&\leq \text{máx}\left\{\frac{mst_r(N_1)}{mst(N_1)}, \frac{mst_r(N_2)}{mst(N_2)}\right\} \\
&\leq \frac{(s+1)2^s + t}{s2^s + t}
\end{aligned}$$

por nuestra hipótesis de inducción. Por lo tanto, sólo tenemos que considerar los casos donde T es un árbol de Steiner completo, esto es, todos lo vértices terminales son hojas.

Añadiendo aristas de peso 0 y vértices Steiner, modificamos T para ser un árbol donde cada vértice Steiner tiene grado exactamente 3. Entonces escogemos una raíz en la mitad de una arista para convertir a T en un árbol binario regular ponderado, nombrémoslo T'. Ver figura 3.3

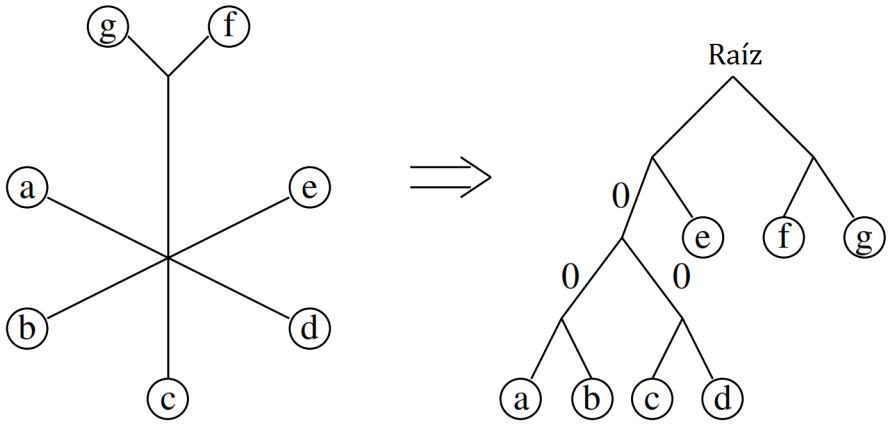


FIGURA 3.3

Etiquetamos todos los vértices internos de t' con conjuntos de tamaño exactamente 2^s escogidos de los números $\{1, 2, \ldots, s2^s + t\}$. El etiquetamiento de un vértice es determinado inductivamente por las etiquetas de los s vértices arriba de él en la trayectoria hacia la raíz, sus s antecesores inmediatos.

Para empezar este etiquetamiento, etiquetamos a la raíz con el conjunto $\{1, 2, \ldots 2^s\}$. Etiquetamos los dos vértices del siguiente nivel con el conjunto $\{2^s + 1, 2^s + 2, \ldots, 2 \cdot 2^s\}$. Y, en general, etiquetamos a todos los vértices en el i-ésimo nivel, para $1 \leq u \leq s$, con el conjunto $\{(i-1)2^s + 1, (i-1)2^s + 2, \ldots, i2^s\}$.

El etiquetamiento inductivo mantendrá la siguiente propiedad, la cual podemos ver que el etiquetamiento para los primeros s niveles lo satisface.

Propiedad de disyuntividad. Los conjuntos de etiquetas de hasta s vértices consecutivos en una trayectoria hacia la raíz del árbol son conjuntos disjuntos.

Supongamos que los primeros i niveles han sido etiquetados, $i \geq s$, y que la propiedad de disyuntividad cumple para los i niveles. Etiquetaremos los vértices del nivel $i + 1$ con las siguientes reglas.

Regla 1. Sea v un vértice en el nivel $i + 1 - s$ etiquetado con el conjunto $S_v = l_1, l_2, \ldots, l_{2^s}$. Etiquetamos el j-ésimo descendiente de v en el nivel $i+1$ con el conjunto $S_j = \{l_j, j_{j+1}, \ldots, l_{j+2^s-t-1}\}$, reduciendo el subíndice (modulo 2^s) para que estén en el rango de 1 a 2^s.

El vértice v tiene a lo más 2^s descendientes en el nivel $i + 1$, así que a lo más necesitamos los conjuntos $S_1, S_2, \ldots, S_{2^j}$. Cada conjunto S_j tiene a lo más $2^s - t$

elementos, y cada etiqueta l_k de S_v aparece en a lo más $2^r - s$ de esos conjuntos, para ser específicos $l_k \in S_k, S_{k-1}, \ldots, S_{k-2^s+t+1}$ donde de nuevo reducimos los subíndices (modulo 2^s).

Para completar el etiquetamiento en el $(i+1)$-ésimo nivel, debemos de agregar t números al conjunto de etiquetas.

Regla 2. Para cada vértice en el nivel $(i+1)$, agrega a su conjunto de etiquetas las t etiquetas que no están en conjuntos de etiquetas de los s ancestros inmediatos.

Por la propiedad de disyuntividad, los s ancestros inmediatos de un vértice en el nivel $i+1$ están etiquetados por s conjuntos disjuntos de tamaño 2^s, así que debe de haber t números de $\{1, 2, \ldots, s2^s + t\}$ que no son utilizados. También, las etiquetas agregadas por la regla 2 serán diferentes a las etiquetas agregadas por la regla 1, así que todos los vértices en el nivel $i+1$ están dados por exactamente 2^s etiquetas por las dos reglas.

La propiedad de disyuntividad es cierta para el nivel $i+1$, ya que el vértice u en el nivel $i+1$ tiene etiquetas tomadas del conjunto de etiquetas del s-ésimo ancestro, las cuales, por la propiedad de disyuntividad, no son usadas por los $s-1$ ancestros inmediatos en el nivel i.

Por inducción, podemos etiquetar todo el árbol. Ver figura 3.4 para ver un ejemplo de este etiquetamiento con $r = 5$.

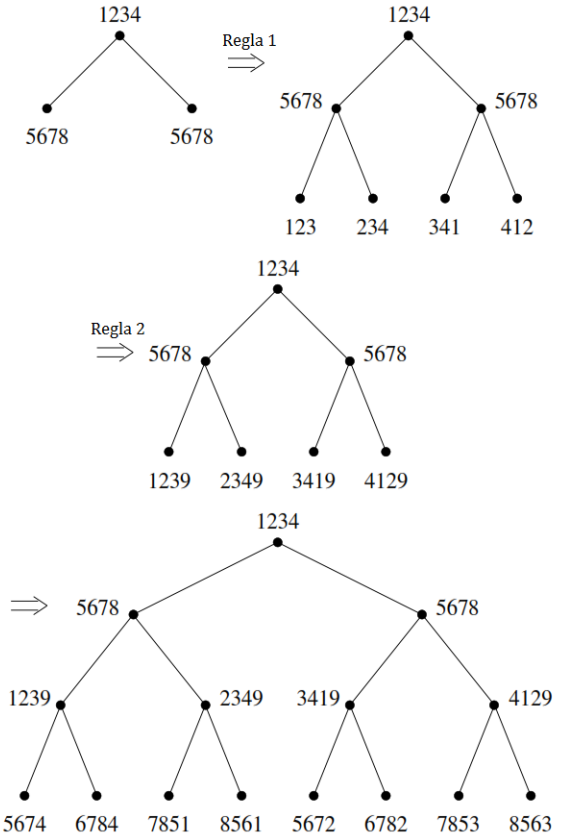


FIGURA 3.4

Ahora usamos el etiquetamiento para obtener $s2^s+t$ árboles de Steiner r-acotados. Cada etiqueta $l = 1, 2, \ldots, s2^s+t$ determina el árbol de Steiner r-acotado T_l. Cada nodo etiquetado con l se vuelve una raíz para un componente en T_l. Adicionalmente, siempre habrá un componente en T_l que su raíz componente es la raíz de T', incluso cuando T' no tiene la etiqueta l. Observemos un componente que tiene una raíz componente v. Conectamos a v con los vértices abajo de el que también tengan a l en su etiqueta. Llamamos a esos vértices *Hojas intermedias* del componente. De la hoja intermedia u, el componente sigue la trayectoria $p(u)$ a la hoja del árbol $f(u)$, como se vio en el lema 3.8. Si no hay vértices con la etiqueta l debajo de v, entonces el componente se extiende directamente hasta una hoja del árbol. Ver figura 3.5.

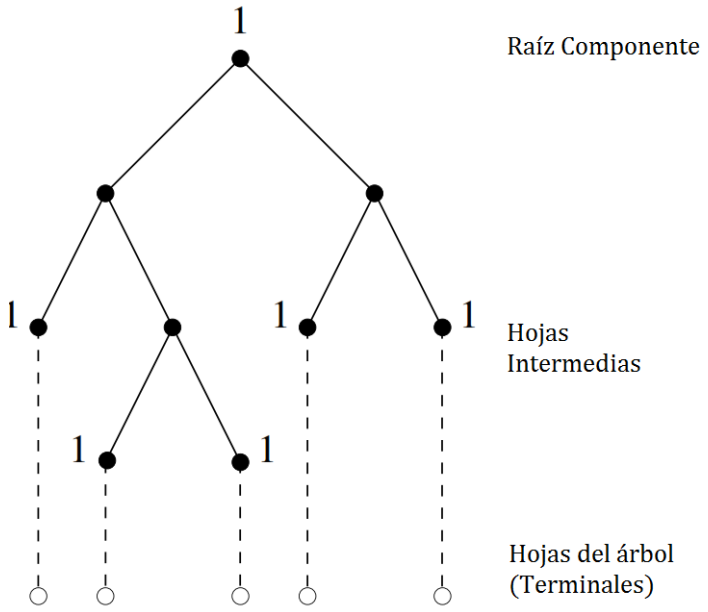


FIGURA 3.5: Un componente en T_1 del etiquetado de la figura 3.4.

Primero, verificamos que T_l sea un árbol que expande los vértices terminales. (Hay que recordar que las hojas de T' son los vértices terminales). Demostraremos esto por inducción sobre la altura de T'. Esto es trivialmente cierto cuando la altura es 0. Tomamos en cuenta al componente que está en la parte más alta de T_l, el componente el cual su raíz es la raíz de T'. Ahora tomamos en cuenta a un árbol que tiene como raíz a una de las hojas intermedias del componente anteriormente mencionado. Por inducción. T_l restringe a este subárbol a expandir vértices terminales los cuales son hojas de ese subárbol. El componente de arriba, que también es un árbol, cubre un vértice terminal que también es cubierto por uno de los árboles debajo de las hojas intermedias.

Ahora observaremos a un componente en T_l que su raíz componente es v. Verificaremos que esos componentes siempre son de tamaño a lo más r. Si un componente termina en una hoja del árbol antes de alcanzar una hoja intermedia, su tamaño será más pequeño, así que sólo observaremos a los componentes que tienen el máximo número posible de hojas intermedias.

Supongamos que v no está etiquetado con l. así que v es la raíz de T' y $l \geq 2^s + 1$. Por el etiquetamiento inicial, $2^s + 1, 2^s + 2, \ldots, s2^s$ aparecen en todos los vértices de uno de los $s-1$ niveles debajo de la raíz. Y, por la regla 2, las demás t etiquetas

aparecen en todos los vértices en el s-ésimo nivel debajo de la raíz. Por lo tanto las hojas intermedias están en el s-ésimo nivel o más arriba y son de tamaño a lo más $2^s \leq r$.

Por lo tanto, podemos suponer que v está etiquetado con l. Por la regla 1 del proceso de etiquetamiento, sabemos que t descendientes de v, que están s niveles abajo, no tienen la etiqueta l y los $2^s - t$ descendientes restantes están etiquetados con l.

Ahora observaremos a un vértice w que está s niveles debajo de la raíz componente v y que no esté etiquetado con l. Ya que v tiene la etiqueta l, por la propiedad de disyuntividad, ninguno de los $s-1$ vértices en la trayectoria de v a w puede tener la etiqueta l. Ya que l no etiqueta a w, por la regla 2 del proceso de etiquetamiento, los hijos de w deben de ser etiquetados con l y deben de ser hojas intermedias.

Entonces el componente con raíz en v tiene $2t$ hojas intermedias $s+1$ niveles debajo de v y $2^s - t$ hojas intermedias s niveles debajo de v. Por lo tanto, hay $2^s + t = r$ hojas intermedias, lo que hace que el componente sea de tamaño r. Por lo tanto, todos los componente son de tamaño a lo más r, y T_l es un árbol de Steiner r-acotado en el conjunto K de vértices terminales.

En T_l, sea L_l ser la suma del peso de las trayectorias $p(u)$ de las hojas intermedias u en T_l a las hojas del árbol. Consideramos la suma $L_1 + L_2 + \cdots + L_{s2^s+t}$. Como cada vértice interno u en T' es una hoja intermedia en exactamente 2^s de los árboles de Steiner r-acotados, en otras palabras, T_l para cada l en el conjunto de u, la longitud de $p(u)$ va a ser contada exactamente 2^s veces en la suma. Como esas trayectorias son disjuntas para diferentes hojas intermedias, la suma de las trayectorias en todos los términos de la suma será a lo más 2^s veces la longitud completa del árbol T', la cual es la longitud del árbol de Steiner mínimo. Entonces

$$L_1 + L_2 + \cdots + L_{s2^s+t} \leq 2^s \cdot smt(N)$$

Debe de existir una d tal que

$$\text{mín}\{L_1, L_2, \ldots, L_{s2^s+t}\} = L_d \leq \frac{2^s}{s2^s + t} \cdot smt(N)$$

La longitud de todo el árbol T_d es la longitud de los componentes desde la raíces componentes hasta las hojas intermedias más la longitud desde las hojas intermedias a las hojas del árbol. Si de cada componente de T_d removemos las trayectorias de las hojas intermedias a las hojas del árbol, los componentes resultantes cubren T' exactamente una vez, así que la suma de esos componentes es $smt(N)$. Las otras partes tienen longitud L_d. Por lo tanto,

$$\begin{aligned} longitud(T_d) &= smt(N) + L_d \\ &\leq \left(1 + \frac{2^s}{s2^s + t}\right) \cdot smt(N) \\ &= \frac{(s+1)2^s + t}{s2^s + t} \cdot smt(N) \end{aligned}$$

Siendo T_d nuestro árbol de Steiner r-acotado, tenemos que

$$\begin{aligned} \frac{mst_r(N)}{smt(N)} &\leq \frac{\frac{(s+1)2^s + t}{s2^s + t} \cdot smt(N)}{smt(N)} \\ &= \frac{(s+1)2^s + t}{s2^s + t} \end{aligned}$$

Esto es la ecuación (3.5), lo que prueba el teorema 3.9.

□

3.3. Algoritmo de aproximación para el problema del árbol de expansión mínima en hipergráficas

Gracias a la última sección podemos definir el siguiente teorema:

Teorema 3.10. *Sea $r \geq 2$ un entero tal que $r = 2^s + t$ y $0 \leq t < 2^s$. Entonces*

$$p_r = \frac{(s+1)2^s + t}{s2^s + t}$$

Demostración

Directo por los teoremas 3.7 y 3.9. □

Teniendo el teorema 3.10 a mano, podemos observar que el árbol de expansión mínima en hipergráficas es un excelente enfoque para aproximar el árbol de Steiner mínimo en gráficas, ya que $p_r \to 1$ cuando $r \to .\infty$. Ver Cuadro 3.1.

r	2	3	4	5	6	7	8	16	32	64
p_r	2	1.67	1.5	1.44	1.4	1.36	1.33	1.25	1.2	1.17

Tabla 3.1: Algunos valores de p_r para una r pequeña.

Entonces podemos decir que lo único que necesitamos para aproximar el problema del árbol de Steiner en gráficas es un algoritmo eficiente para encontrar árboles de expansión mínima en hipergráficas. Sin embargo esto tampoco es una tarea sencilla.

Teorema 3.11. *Encontrar un árbol de expansión mínima en una hipergráfica ponderada r-acotada es un problema NP-hard para toda $r \geq 4$ [18].*

Lo que significa que, para obtener un algoritmo de aproximación para el problema del árbol de Steiner mínimo en gráficas en tiempo polinomial con el enfoque mencionado anteriormente, es necesario poder calcular un árbol de expansión mínima

en una hipergráfica r-acotada en tiempo polinomial. Por lo que se opta por encontrar un algoritmo polinomial de aproximación para el problema de encontrar un árbol de expansión mínima en una hipergráfica r-acotada.

En esta sección se mostrará un algoritmo de aproximación para el problema de encontrar árbol de expansión mínima en hipergráficas r-acotadas con factor de aproximación $(1 + ln2) \approx 1.69$.

Sea $N = (V, E, l, K)$ una gráfica de la cual queremos resolver el problema del árbol de Steiner. La meta es encontrar un árbol de expansión de poco peso en la hipergráfica de r-distancia $H_r(N)$ y entonces convertirla en un árbol de Steiner en N. La estrategia para encontrar este árbol de expansión es ir tomando la hiperarista que más nos convenga en cada iteración. Sin embargo, no podemos sólo escoger la hiperarista con menos peso en cada paso, esto siempre nos daría una hiperarista de dos elementos, y, como resultado, obtendríamos un árbol de expansión en $H_2(N)$. Por lo que tenemos que inventar una métrica para las aristas que permita saber como se deben de seleccionar. Esta métrica permitirá que una hiperarista sea relativamente pesada si su contribución al árbol de expansión es suficientemente grande.

Antes de definir esta métrica, introduciremos una nueva notación.
Sea $H_2(N) = (K, E_2.l_2)$ nuestra gráfica de distancia de N. Para un subconjunto $X \subseteq E_2$, denotamos por $H_2(N; X)$ a la gráfica que surge de H_2 si reemplazamos al peso de todas las aristas de X por cero. Similarmente, para un conjunto $Y \subseteq K$, $mst_2(N; Y)$ denota al árbol de expansión mínima de N donde todas las aristas que conectan a dos vértices de Y tienen peso cero, es decir, $mst_2(N; Y)$ es el árbol de expansión mínima de $H_2(X; \binom{Y}{2})$. Sea $Y_1, Y_2, \ldots, Y_j$ una familia de aristas de $H_r(N) = (K, E_r, l_r)$, utilizamos $mst_2(N; Y_1, Y_2, \ldots, Y_j)$ para denotar al árbol de expansión mínima de $H_2(N; Y_1, Y_2, \ldots, Y_j)$.

Como medida de qué tanto contribuye la hiperaristas $Y_i \in E_r$ en el árbol de Steiner resultante, vemos que efecto tiene el reducir el peso de la hiperaristas Y_i en la longitud del árbol de expansión mínima de $H_2(N; Y_1, Y_2, \ldots, Y_{i-1})$. Siendo precisos, utilizamos

$$mst_2(N; Y_1, \ldots, Y_{i-1}) - mst_2(N; Y_1, \ldots, Y_{i-1}, Y_i)$$

Entre más grande es esta cantidad, más útil será escoger la hiperarista Y_i.

Ahora se mostrará el algoritmo de Zelikovsky para calcular un árbol de Steiner utilizando la métrica mencionada. De hecho, se mostrará una familia de algoritmos A_r, uno para cada entero $r \geq 2$.

Algoritmo 3.3.1 Algoritmo A_r de Zelikovski

Entrada: Una gráfica $N = (V, E, l, K)$.

Salida: Un árbol de Steiner T_z para K.

1: Calcular $H_2(N) = (K, E_2, l_2)$.
2: Calcular $H_r(N) = (K, E_r, l_r)$.
3: $i \leftarrow 1$.
4: **mientras** $Y_1 \cup, \ldots, \cup Y_i$ no es un árbol de expansión conectado de $H_r(N)$ **hacer**
5: $\quad$ *mínimo* $\leftarrow \infty$.
6: $\quad$ **para todo** $e \in E_r$ **hacer**
7: $\quad\quad peso = \dfrac{l_r(e)}{mst_2(N; Y_1, \ldots, Y_i) - mst_2(N; Y_1, \ldots, Y_i, e)}$
8: $\quad\quad$ **si** *peso* $<$ *mínimo* **entonces**
9: $\quad\quad\quad$ *mínimo* $\leftarrow$ *peso*.
10: $\quad\quad\quad Y_{i+1} \leftarrow e$.
11: $\quad\quad$ **fin si**
12: $\quad$ **fin para**
13: $\quad i \leftarrow i + 1$.
14: **fin mientras**
15: De $Y_1, \cup \cdots \cup, Y_i$ calcular un árbol de Steiner T_z para K en N.
16: **devolver** T_z

Antes de demostrar que este algoritmo es un algoritmo de aproximación con un buen factor de aproximación, primero mostraremos que corre en tiempo polinomial.

Proposición 3.12. *Sea $N = (V, E, l, K)$ una gráfica y $r \geq 2$. El algoritmo A_r de Zelikovsky termina en tiempo $O(n^2 \log n + nm + k^{r+1}n^2)$*

Demostración

Por el lema 3.2 sabemos que el paso del preprocesamiento, es decir, calcular $H_r(N)$ y $H_2(N)$, y el paso del postprocesamiento, es decir, calcular el árbol de Steiner de N a partir del árbol de expansión de $H_r(N)$ puede realizarse en tiempo $O(n^2 \log n + nm + k^{r+1}n^2)$.

El ciclo *while* se ejecutará a lo más $k - 1$ veces. Esto sucede cuando el árbol de Steiner que selecciona el algoritmo de Zelikovsky es el árbol de expansión de $H_2(N)$. Para cada iteración, su tiempo está acotado por $O(k^r(n \log n + m))$ ya que E_r contiene a lo más k^r aristas y calcular el árbol de expansión puede hacerse en tiempo $O(n \log n + m)$. Como $m \leq n^2$, nuestra proposición es cierta. □

Empezaremos en análisis del algoritmo de Zelikovsky probando un hecho clave de la función $mst_2(N; X)$.

Lema 3.13. *Sea $H_2(N) = (K, E_2, l_2)$ la gráfica de distancia de N y sea $X, Y \subseteq E_2$ y $z \in E_2$. Entonces*

$$mst_2(N; X \cup Y) - mst_2(N; X \cup Y \cup \{z\}) \leq mst_2(N; X) - mst_2(N; X \cup \{z\}).$$

Demostración

Sea T un árbol de expansión mínima en $H_2(N; X)$ y sea $T(Y)$ el árbol de expansión mínima de $H_2(N; X \cup Y)$ el cual puede ser obtenido insertando sucesivamente las aristas de $Y \setminus E(T)$ en T y en cada paso remover la arista más grande del ciclo que se va formando. Sea $z = \{z_0, z_1\}$. Nótese que $mst_2(N; X) - mst_2(N; X \cup \{z\})$ es igual al peso de la arista más pesada en la trayectoria que conecta z_0 con z_1, en T.

Ahora sea $c \geq 0$ alguna constante. Denotamos por T_c, $T_c(Y)$ respectivamente, a la

subgráfica de T, $T(Y)$ respectivamente, que es inducida por todas las aristas e que satisfacen $l_2(e) \leq c$. Notese que T_c y $T_c(Y)$ pueden ser desconectadas. Entonces tenemos que

$$\begin{aligned} &mst_2(N;X) - mst_2(n;X \cup \{z\}) \\ &\qquad = \text{mín}\{c \geq 0 \mid z_0 \textit{ y } z_1 \textit{ están en el mismo componente de } T_c\} \\ &\qquad \geq \text{mín}\{c \geq 0 \mid z_0 \textit{ y } z_1 \textit{ están en el mismo componente de } T_c(Y)\} \\ &\qquad = mst_2(N;X \cup Y) - mst_2(N;X \cup Y \cup \{z\}) \end{aligned}$$

□

Corolario 3.14. *Sea $H_2(N) = (K, E_2, l_2)$ la gráfica de distancia de N y sea $X, Y, Z \subseteq E_2$. Entonces*

$$mst_2(N;X \cup Y) - mst_2(N;X \cup Y \cup Z) \leq mst_2(N;X) - mst_2(Z;X \cup Z).$$

Demostración

Probamos el corolario por inducción en $|Z|$. Para $|Z| = 1$ aplicamos el lema 3.13. Supongamos que la afirmación es cierta para todo $|Z| \leq t$ para una $t \geq 1$. Sea $Z = Z' \cup \{z\}$ de tamaño $t + 1$. Por la hipótesis inductiva tenemos que

$$mst_2(N;X \cup Y) - mst_2(N;X \cup Y \cup Z') \leq mst_2(N;X) - mst_2(N;X \cup Z')$$

y por el lema 3.13 concluimos que

$$\begin{aligned} &mst_2(N;X \cup Y \cup Z') - mst_2(N;X \cup Y \cup Z' \cup \{z\}) \leq \\ &mst_2(N;X \cup Z') - mst_2(N;X \cup Z' \cup \{z\}) \end{aligned}$$

□

A continuación, veremos un hecho útil sobre secuencias de números enteros.

Proposición 3.15. *Sean* $a_i \geq 0,\ b_i > 0$ *enteros para* $i = 1, \ldots, n$. *Entonces*

$$\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \geq \min_{1 \leq j \leq n} \frac{a_j}{b_j}$$

Demostración

$$\left(\min_{1 \leq j \leq n} \frac{a_j}{b_j} \right) \cdot \left(\sum_{i=1}^{n} b_i \right) = \sum_{i=1}^{n} \left(b_i \cdot \min_{j} \frac{a_j}{b_j} \right) \leq \sum_{i=1}^{n} b_i \cdot \frac{a_i}{b_i} = \sum_{i=1}^{n} a_i$$

□

Ahora estamos listos para probar que el algoritmo de Zelikovsky calcula un árbol de Steiner el cual su peso es un múltiplo pequeño del árbol de expansión mínima de una hipergráfica r-acotada.

Teorema 3.16. *Sea* $N = (V, E, l; K)$ *una gráfica y* $r \geq 2$. *El algoritmo* A_r *de Zelikovsky calcula un árbol de Steiner* T_Z *para* K *en tiempo polinomial tal que*

$$l(T_Z) \leq (1 + \ln 2) \cdot mst_r(N)$$

Demostración

Sea $T^* = Y_1^* \cup \cdots \cup Y_s^*$ un árbol de expansión mínima de $H_r(N)$, es decir, $l_r(T^*) = mst_r(N)$, y sea $Y_1, \ldots, Y_t$ las hiperaristas escogidas por el algoritmo A_r.

Observese que mientras $Y_1 \cup \cdots \cup Y_i$ no es una gráfica de expansión de $H_r(N)$, siempre existe una arista $e \in E_2$ (por ejemplo, cualquier aristas en un árbol de expansión mínima en $H_2(N; Y_1, \ldots, Y_i)$ con aristas de peso diferente de cero) tal que $l_2(e) \leq mst_2(N; Y_1, \ldots, Y_i) - mst_2(N; Y_1, \ldots, Y_i, e)$. Por lo tanto, para cada $i = 1, \ldots, t-1$ tenemos que

$$f_i(Y_{i+1}) \leq 1 \tag{3.6}$$

El algoritmo de Zelikovsky selecciona en el $(i+1)$-ésimo paso la hiperarista $Y_{i+1} \in E_r$ que minimice f_i. por lo tanto tenemos

$$\begin{aligned} f_i(Y_{i+1}) &\leq \min_j f_i(Y_j^*) \\ &= \min_j \frac{l_r(Y_j^*)}{mst_2(N; Y_1, \ldots, Y_i) - mst_2(N; Y_1, \ldots, Y_i, Y_j^*)} \end{aligned}$$

y por lo tanto, aplicando la proposición 3.15, tenemos que

$$f_i(Y_{i+1}) \leq \frac{\sum_j l_r(Y_j^*)}{\sum_j mst_2(N; Y_1, \ldots, Y_i) - mst_2(N; Y_1, \ldots, Y_i, Y_j^*)}$$

El numerador del lado derecho de la inecuación es igual a $mst_r(N)$. Por el corolario 3.14 podemos ver que el denominador es mayor o igual que

$$\textstyle\sum_j (mst_2(N; Y_1, \ldots, Y_i, Y_1^*, \ldots, Y_{j-1}^*) - mst_2(N; Y_1, \ldots, Y_i, Y_1^*, \ldots, Y_{j-1}^*, Y_j^*))$$

Esto es una suma telescópica en donde todos los términos excepto el primero y el último se cancelan. Ese último termino es $mst_2(N; Y_1, \ldots, Y_i, Y_1^*, \ldots, Y_s^* = 0$ ya que $Y_1^* \cup \cdots \cup Y_s^*$ es un árbol de expansión mínima en $H_r(N)$. Entonces tenemos

$$f_i(Y_{i+1}) \leq \frac{mst_r(N)}{mst_2(N; Y_1, \ldots, Y_i)} \tag{3.7}$$

Para simplificar la notación, diremos que $m_i := mst_2(N; Y_1, \ldots, Y_i)$ para $i = 0, \ldots, t$. Combinando la definición de f_i y 3.6, 3.7 obtenemos:

$$\begin{aligned} l(T_Z) = \sum_{i=1}^{t} l_r(Y_i) &= \sum_{i=1}^{t} f_{i-1}(Y_i)(m_{i-1} - m_i) \\ &\leq \sum_{i=1}^{t} \min\left\{1, \frac{mst_r(N)}{m_{i-1}}\right\}(m_{i-1} - m_i) \end{aligned}$$

la secuencia $m_0, \ldots, m_t$ es monótona decreciente con $m_0 = mst_2(N)$ y $m_t = 0$. Por lo tanto, se puede reemplazar la suma por una integral:

$$\begin{aligned}\sum_{i=1}^{t} \text{mín}\left\{1, \frac{mst_r(N)}{m_{i-1}}\right\}(m_{i-1} - m_i) &\leq \int_{m_t}^{m_0} \text{mín}\left\{1, \frac{mst_r(N)}{m_{i-1}}\right\}dx \\ &= \int_0^{mst_r(N)} 1dx + mst_r(N)\int_{mst_r(N)}^{mst_2(N)} \frac{1}{x}dx \\ &= mst_r(N) + mst_r(N)\cdot \ln\left(\frac{mst_2(N)}{mst_r(N)}\right) \\ &= mst_r(N)\left(1 + \ln\left(\frac{mst_2(N)}{mst_r(N)}\right)\right)\end{aligned}$$

Ya que $mst_r(N) \geq smt(N)$ (ver lema 3.1) y que $\frac{mst_2(N)}{smt(N)} \leq p_2 = 2$ esto da $l(T_Z) \leq (1 + \ln 2)\cdot mst_r(N)$.

□

Teniendo a la mano el lema 3.2, el teorema 3.10 y el teorema 3.16 podemos formular el teorema más importante de este capítulo.

Teorema 3.17. *Sea $N = (V, E, l; K)$ una gráfica a la cual se quiere calcular un árbol de Steiner para K y sea r una constante fija. Existe un algoritmo de aproximación que calcula un árbol de Steiner T_Z tal que*

$$l(T_Z) \leq (1 + \ln 2)\cdot p_r \cdot smt(N)$$

Demostración

Tomamos en cuenta que p_r es la razón de qué tan grande es un $MST_r(N)$ respecto a $SMT(N)$, es decir , $mst_r(N) \leq p_r \cdot smt(N)$. Esto es directo por el lema 3.2, el teorema 3.10 y el teorema 3.16. □

Si escogemos una r suficientemente grande, el factor de aproximación de este algoritmo puede ser menor a dos, siendo que si $r \to \infty$ entonces $p_r \to 1$. Sin embargo, el logro de reducir el factor de aproximación a menos de dos ha sido

costoso en tiempo de procesamiento. El valor de r más pequeño que hace que el producto $(1 + \ln 2) \cdot p_r$ sea menor a dos es 49, lo que nos da un algoritmo de aproximación de tiempo $O(n^2 \log n + nm + k^{50}n^2)$.

Capítulo 4

Algoritmo de aproximación para encontrar un árbol de Steiner mínimo en aristas en polígonos simples

Dado un polígono simple P de n vértices y un conjunto de puntos terminales $Z = \{z_1, z_2, \ldots, z_m\}$ dentro de P. El objetivo es encontrar un árbol de Steiner en aristas T'_Z dentro de P que expanda a Z y que el número de aristas de este árbol de Steiner sea menor al doble del número de aristas del árbol del Steiner mínimo T_Z, esto es,

$$\frac{|E(T'_Z)|}{|E(T_Z)|} < 2$$

El algoritmo que se presentará en este capítulo es una modificación a la estructura del algoritmo del capítulo 3 aplicando el conocimiento de los capítulos 1 y 2.

4.1. Construcción del algoritmo

Este algoritmo sigue la misma idea que el algoritmo del capítulo 3:

1. Calcular la gráfica de distancia $H_2(P) = H_2(K, E_2, l_2)$ del conjunto de terminales Z.
2. Dada una constante r, calcular el árbol de Steiner mínimo en aristas r-acotado $H_r(P) = H_r(K, E_r, l_r)$ del conjunto de terminales Z.
3. Utilizando una métrica, escogeremos las hiperaristas $Y_i \in E_r$ que contribuyan más al árbol de Steiner.
4. A partir de las hiperaristas $Y_1 \cup \cdots \cup Y_i$ contruiremos un árbol de Steiner T_Z para Z en P.
5. A partir del árbol de Steiner T_Z, construiremos un árbol de Steiner en aristas dentro de P para Z.

Sin embargo, algunos de estos pasos sólo es posible realizarse en gráficas, por lo que es necesario modificarlos.

4.1.1. Gráfica de distancia $H_2(Z, P)$

Normalmente una gráfica de distancia es generada a partir de una gráfica $N = (V, E, l)$. Sin embargo, en este caso recibimos como entrada un polígono simple y un conjunto de puntos dentro del polígono, por lo que la gráfica de distancia debe ser construida a partir de algoritmos que encuentren trayectorias más cortas entre dos puntos dentro de polígonos, por lo que utilizaremos el algoritmo MLP que vimos en el capítulo 1.
Una arista de la gráfica de distancia entre dos vértices terminales z_i y z_j es igual al MLP de esos dos puntos. Es decir,

$$e_{i,j} = MLP(z_i, z_j)\ \forall e_{i,j} \in E_2.$$

Sin embargo, en este paso sólo nos interesa el peso de la trayectoria más corta, es decir, utilizaremos $MLD(P, z_i, z_j) = |MLP(P, z_i, z_j)|$. A partir de esto, se dará un algoritmo para poder calcular la gráfica de distancia deseada.

Algoritmo 4.1.1 Gráfica de distancia

Entrada: Un polígono simple P y un conjunto de puntos Z dentro de P.

Salida: Una matriz de adyacencia de una gráfica completa de $|Z|$ vértices.

1: Sea m la cantidad de puntos en el conjunto Z.
2: Sea ady una matriz de m x m con todos sus elementos con valor de 0.
3: **para** $i \leftarrow 0$ hasta $m-1$ **hacer**
4: **para** $j \leftarrow i+1$ hasta $m-1$ **hacer**
5: $ady[i][j] \leftarrow MLD(P, z_i, z_j)$.
6: $ady[j][i] \leftarrow ady[i][j]$.
7: **fin para**
8: **fin para**
9: **devolver** ady

Lema 4.1. *La complejidad del algoritmo 4.1.1 es de* $O(nm^2)$

Demostración

Sea n el número de vértices del polígono simple P. La línea 3 y 4 son ciclos anidados en los cuales cada uno tiene m repeticiones. Por cada iteración del ciclo anidado, se ejecuta el algoritmo de MLD una vez. Por el teorema 1.14, la complejidad de MLD es de $O(n)$. Por lo tanto, la complejidad de este algoritmo es de $O(nm^2)$ □

4.1.2. Árbol de Steiner r-acotado $H_r(Z, P)$

Después de crear la gráfica de distancia, el algoritmo de Zelikovsky (algoritmo 3.3.1) calcula un árbol de Steiner de cada subconjunto de Z de tamaño menor o igual a r con el algoritmo de Dreyfus y Wagner, donde r es una constante fija.
Sin embargo, nosotros no podemos utilizar el algoritmo de Dreyfus y Wagner, ya que este espera como entrada una gráfica. En vez de utilizar el algoritmo de

Dreyfus y Wagner, utilizaremos el algoritmo 2.1.2, el cual hace la misma función que el algoritmo de Dreyfus y Wagner, pero enfocado en arboles de Steiner mínimos dentro de polígonos simples.

4.1.3. Métrica

Teniendo la gráfica de distancia $H_2(P)$ y el conjunto $H_r(P)$ de todos los árboles de Steiner mínimos de los subconjuntos de Z de tamaño menor o igual a r, procedemos a escoger algunos arboles de Steiner de este conjunto para crear un árbol de Steiner que expanda a Z dentro de P y que no sea mucho más grande que $SMT(Z, P)$.
Para esto, utilizaremos la misma métrica a la del algoritmo de Zelikovsky, con la diferencia de que utiliza arboles de Steiner mínimos r-acotados dentro de polígonos en vez de arboles de Steiner mínimos en una gráfica.
Utilizando la notación de la sección 3.3, la métrica consiste en escoger el mínimo de la siguiente función:

$$f(e) = \frac{l_r(e)}{mst_2(Z, P; Y_1, \ldots, Y_i) - mst_2(Z, P; Y_1, \ldots, Y_i, e)} \tag{4.1}$$

Donde $Y_1, \ldots, Y_i$ es el conjunto de hiperaristas seleccionadas en las i iteraciones anteriores. Lo que significa que cada Y_i es un árbol de Steiner r-acotado de un subconjunto de Z en P.

4.2. Algoritmo de aproximación para $SMT(Z, P)$

Como se mencionó al principio, el algoritmo de aproximación dado en este capítulo está basado en el algoritmo de Zelikovsky (algoritmo 3.3.1), modificado para poder funcionar sobre puntos dentro de un polígono simple. A continuación mostraremos el nuevo algoritmo que se construyó basado en la modificaciones mencionada en la sección anterior:

Algoritmo 4.2.1 Algoritmo de aproximación para $SMT(Z, P)$

Entrada: Un polígono simple P y un conjunto de puntos Z dentro de P.

Salida: Un árbol de Steiner que expanda a Z en P.

1: Calcular $H_2(Z, P) = (Z, E_2, l_2)$.
2: Calcular $H_r(Z, P) = (Z, E_r, l_r)$.
3: $i \leftarrow 0$.
4: $Y_0 \leftarrow \{\}$.
5: **mientras** $Y_0 \cup, \ldots, \cup Y_i$ no es un árbol de expansión conectado de $H_r(Z, P)$ **hacer**
6: $\quad$ *mínimo* $\leftarrow \infty$.
7: $\quad$ **para todo** $e \in E_r$ **hacer**
8: $\quad\quad$ $peso = \dfrac{l_r(e)}{mst_2(Z, P; Y_0, \ldots, Y_i) - mst_2(Z, P; Y_0, \ldots, Y_i, e)}$
9: $\quad\quad$ **si** *peso* $<$ *mínimo* **entonces**
10: $\quad\quad\quad$ *mínimo* $\leftarrow$ *peso*.
11: $\quad\quad\quad$ $Y_{i+1} \leftarrow e$.
12: $\quad\quad$ **fin si**
13: $\quad$ **fin para**
14: $\quad$ $i \leftarrow i + 1$.
15: **fin mientras**
16: De $Y_1, \cup \cdots \cup, Y_i$ calcular un árbol de Steiner T_z para Z en P.
17: Convertir T_z a su representación dentro del polígono APROX_SMT(Z, P).
18: **devolver** APROX_SMT(Z, P)

Ahora procedemos a calcular la complejidad del algoritmo.

Teorema 4.2. *Dada una constante r, un polígono simple P de n vértices y un conjunto de puntos terminales $Z = \{z_1, z_2, \ldots, z_m\}$ dentro de P. La complejidad del algoritmo 4.2.1, el cual aproxima $SMT(Z, P)$ con una razón de aproximación menor a dos, es de $O(2^r nr(n + r) + k^{r+1} m^3 \log m)$.*

Demostración

Por el lema 4.1, calcular $H_2(Z, P)$ toma $O(nm^2)$ y por el teorema 2.9, calcular $H_r(Z, P)$ toma $\theta(2^r nr(n+r))$.

El máximo número de iteraciones que puede tener el ciclo *mientras* es de $m-1$, ya que el árbol de expansión con más hiperaristas que puede encontrar este algoritmo es un árbol donde todas sus hiperaristas son de cardinalidad 2, es decir, un árbol de expansión mínima.

El ciclo *paratodo* se repite una vez por cada elemento de E_r. El número de elementos de E_r está dado por $\sum_{i=2}^{r} \binom{k}{i} \leq \sum_{i=2}^{r} \frac{k^i}{i!} \leq k^{r+1}$. Dentro de este ciclo, el procedimiento más complejo es calcular un árbol de expansión mínima, el cual toma $O(m^2 \log m)$ utilizando el algoritmo de Kruskal. Por lo tanto, el tiempo que tarda en terminar el ciclo *mientras* es de $O(k^{r+1} m^3 \log m)$.

Por último, la complejidad de construir un árbol de Steiner T_z a partir de $Y_1, \cup \cdots \cup, Y_i$ y construir un árbol de Steiner en aristas dentro del polígono P está dado por el número de vértices que tendrá el árbol de Steiner. Por el teorema 1.3, éste número está acotado por $O(n)$.

Por lo tanto, la complejidad del algoritmo es de $O(2^r nr(n+r) + k^{r+1} m^3 \log m)$

□

4.3. Ejecución del algoritmo de proxumación

A continuación se muestra un ejemplo de ejecución del algoritmo 4.2.1 para ilustrar su funcionamiento.

Para éste ejemplo se utilizará la figura 4.1 como el polígono de entrada P y los 5 vértices dentro del polígono de la figura 4.1 será el conjunto de entrada Z. Se fijará la constante r a 3.

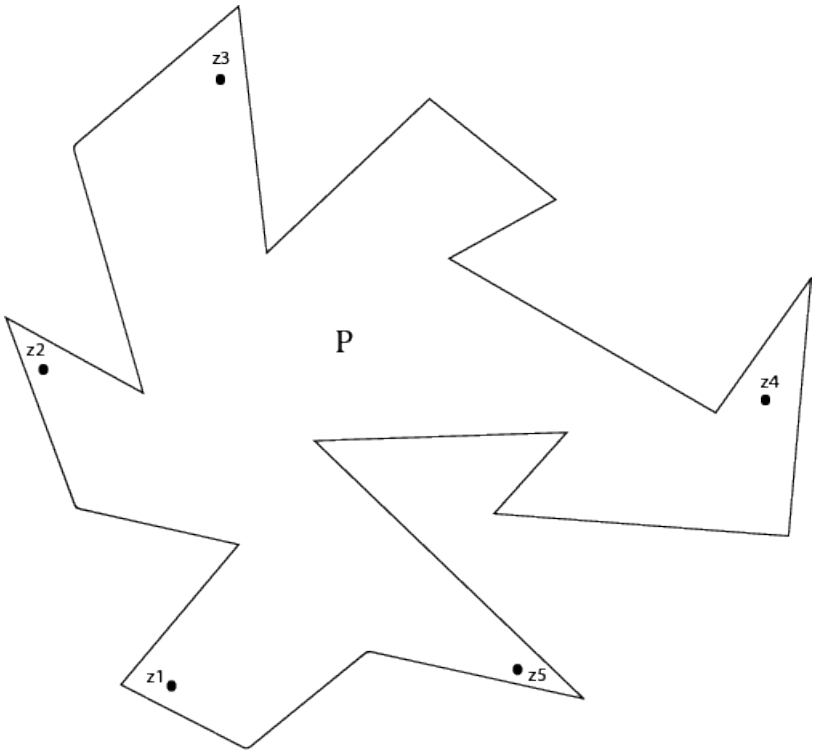


FIGURA 4.1: Polígono de entrada P y conjunto de terminales Z.

Paso 1.- Calcular $H_2(Z, P)$ y $H_r(Z, P)$.

Se puede ver a $H_2(Z, P)$ como todos los $MLP(z_i, z_j)$ $\forall z_i, z_j \in Z$ con $i \neq j$ y a $H_r(Z, P)$ como todos los posibles árboles de Steiner de subconjuntos de Z de cardinalidad 2 y 3, siendo que $r = 3$. Como $MLP(z_i, z_j)$ es equivalente a calcular $SMT(\{z_i, z_j\}, P)$, podemos simplemente calcular $H_r(Z, P)$ y tomar $H_2(Z, P)$ como subconjunto de $H_r(Z, P)$ en los casos donde las hiperaristas $e \in E_r$ tienen cardinalidad 2.

Utilizando el algoritmo 2.1.2 una vez para cada conjunto, podemos obtener el árbol de Steiner para cada hiperarista E_r. Cada uno de las siguientes hiperaristas tiene un ejemplo de un posible árbol de Steiner colocando un vértice de Steiner en cada región encontrada.

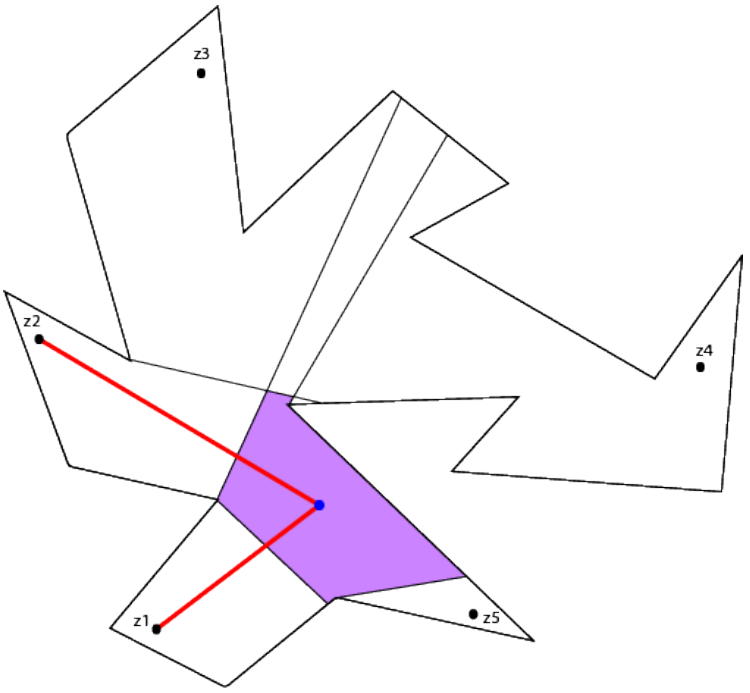


FIGURA 4.2: SMT para los vértices terminales z_1 y z_2

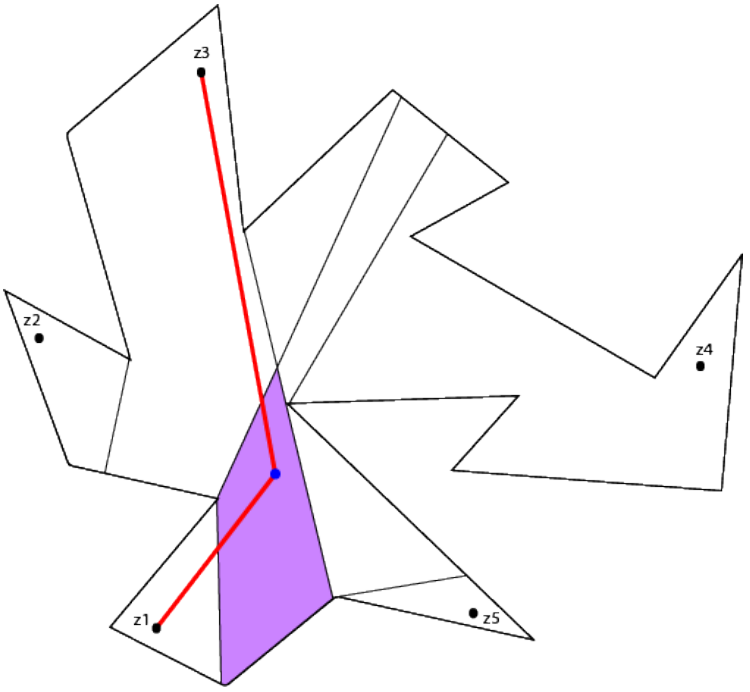


FIGURA 4.3: SMT para los vértices terminales z_1 y z_3

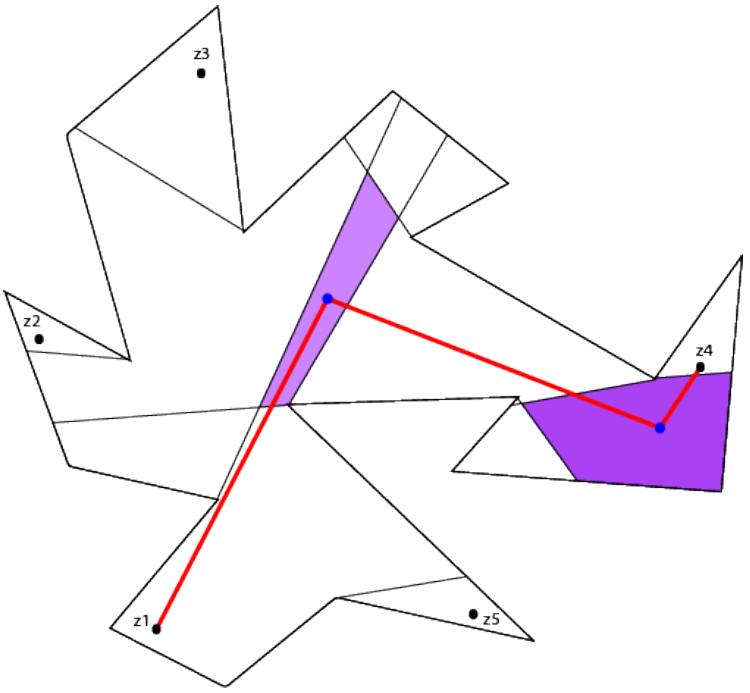


FIGURA 4.4: SMT para los vértices terminales z_1 y z_4

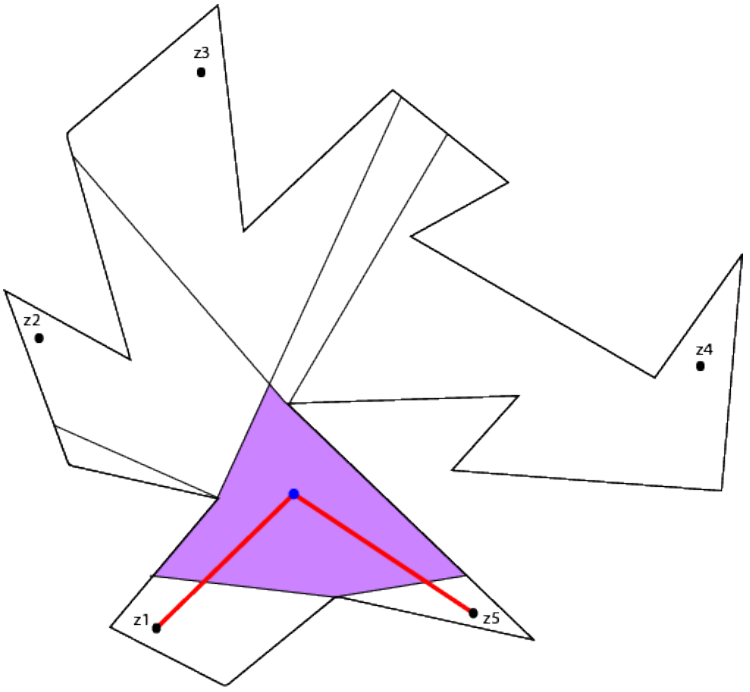


FIGURA 4.5: SMT para los vértices terminales z_1 y z_5

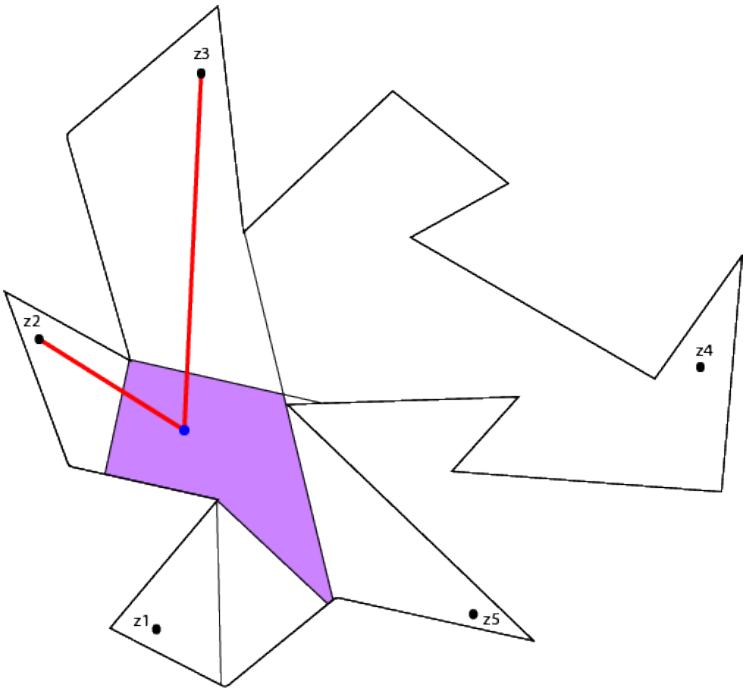


FIGURA 4.6: SMT para los vértices terminales z_2 y z_3

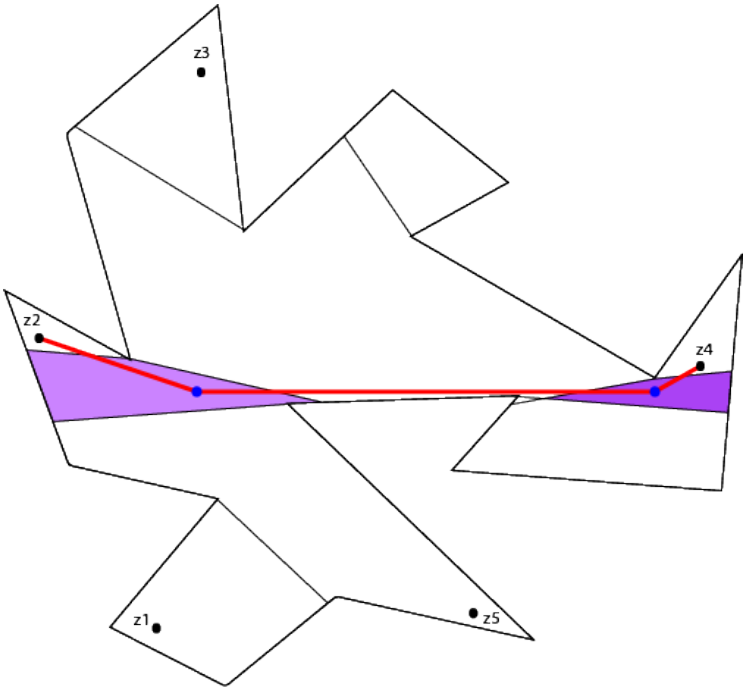


FIGURA 4.7: SMT para los vértices terminales z_2 y z_4

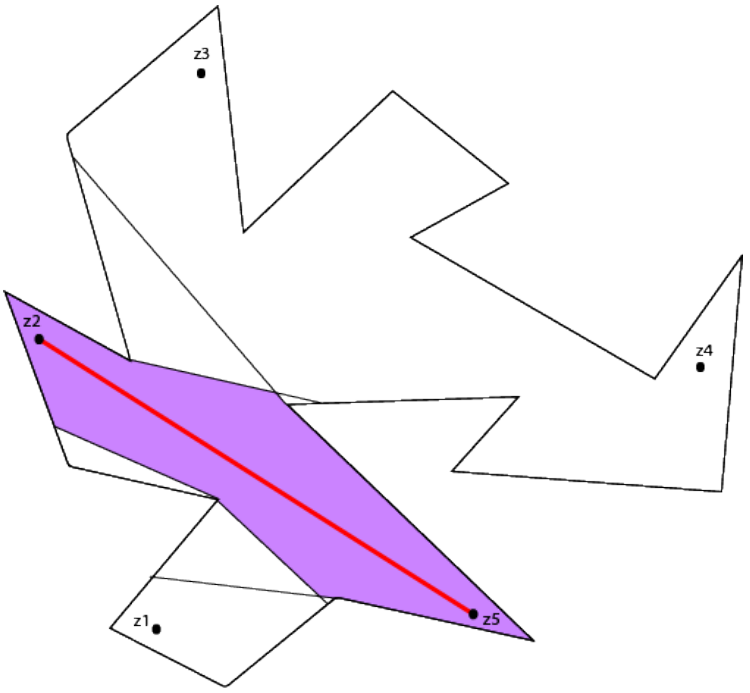


FIGURA 4.8: SMT para los vértices terminales z_2 y z_5

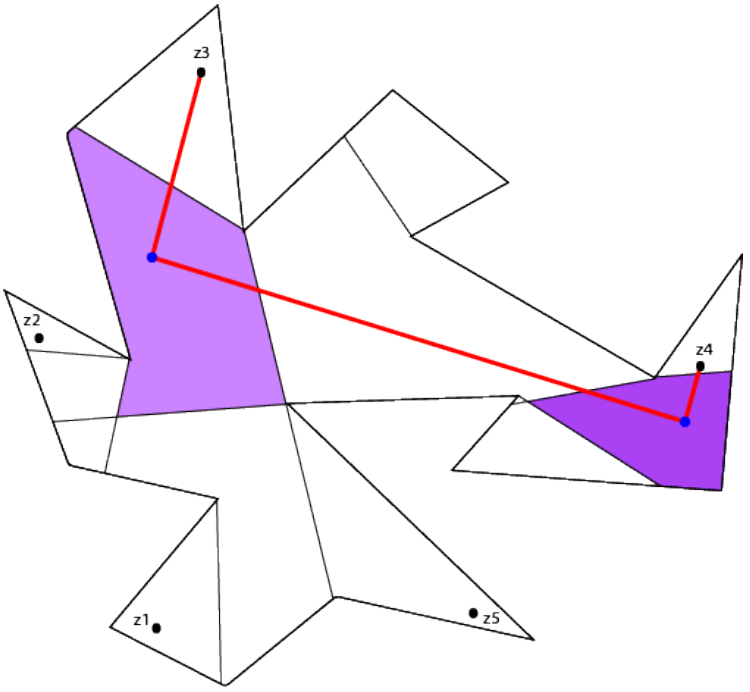


FIGURA 4.9: SMT para los vértices terminales z_3 y z_4

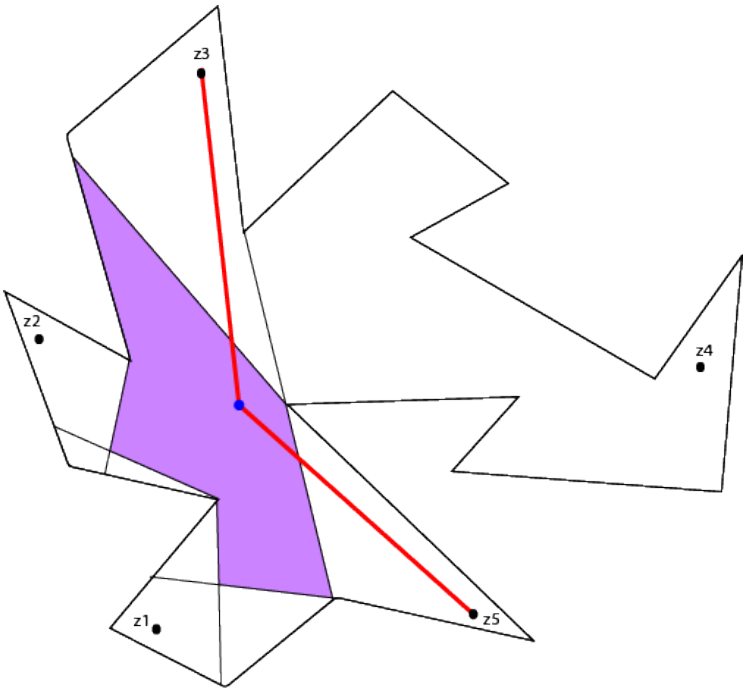


FIGURA 4.10: SMT para los vértices terminales z_3 y z_5

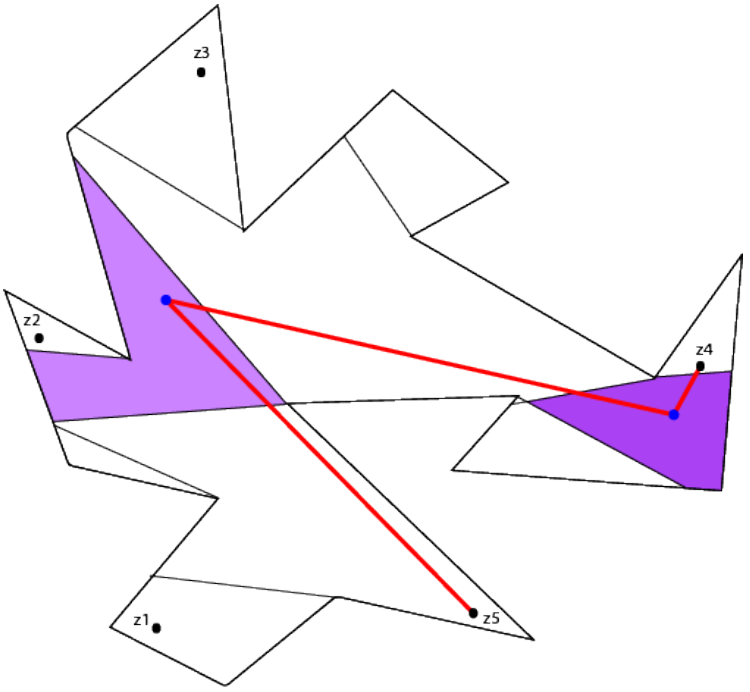


FIGURA 4.11: SMT para los vértices terminales z_4 y z_5

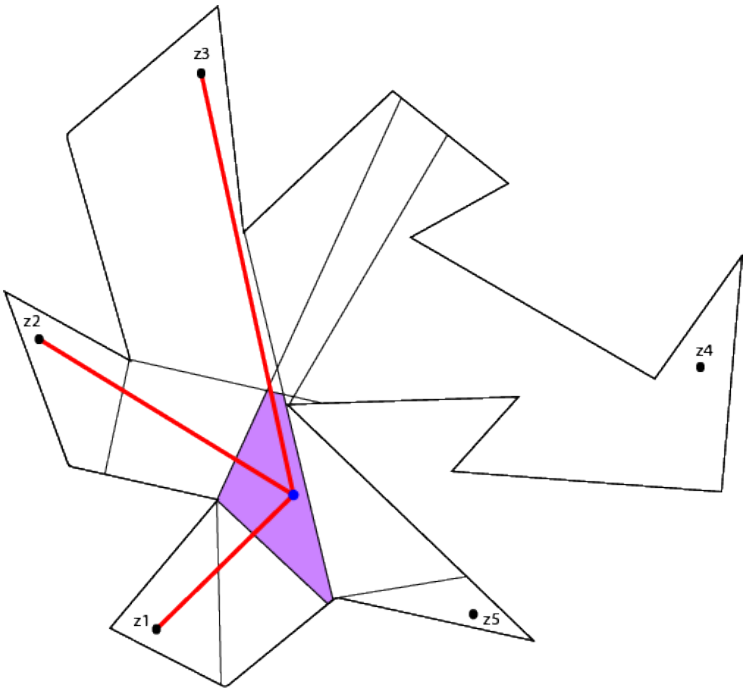


FIGURA 4.12: SMT para los vértices terminales z_1, z_2 y z_3

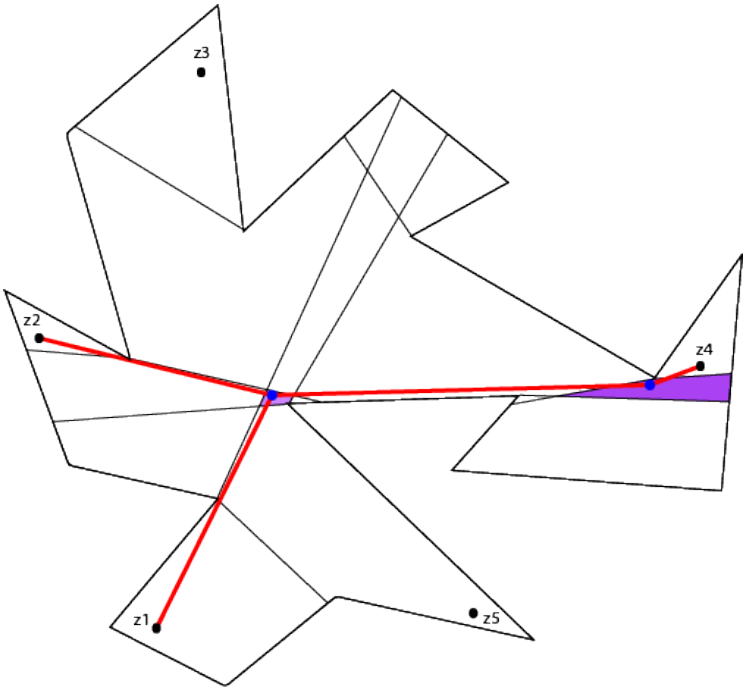


FIGURA 4.13: SMT para los vértices terminales z_1, z_2 y z_4

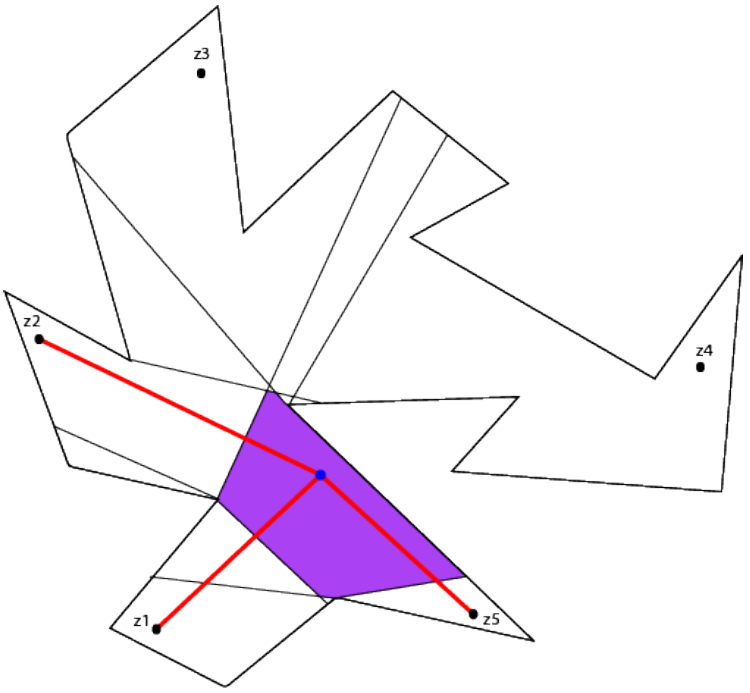


FIGURA 4.14: SMT para los vértices terminales z_1, z_2 y z_5

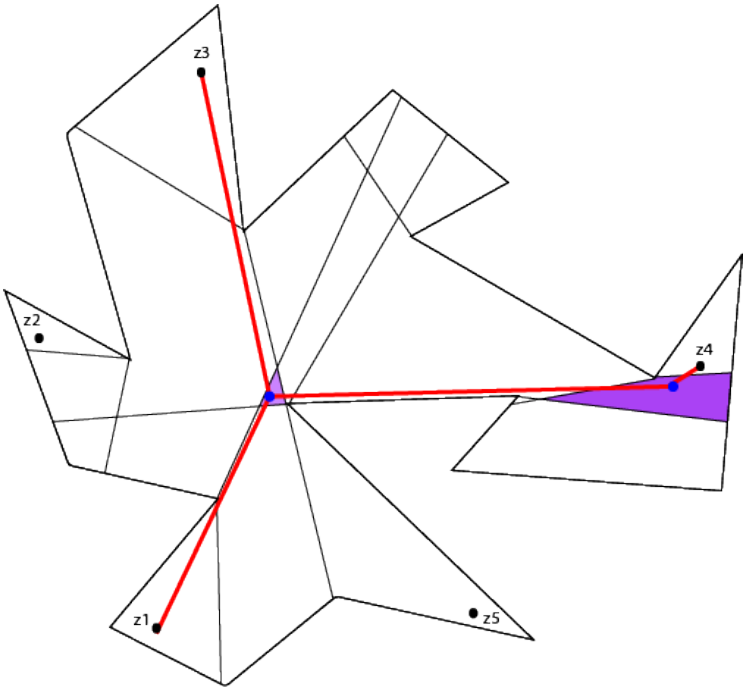


FIGURA 4.15: SMT para los vértices terminales z_1, z_3 y z_4

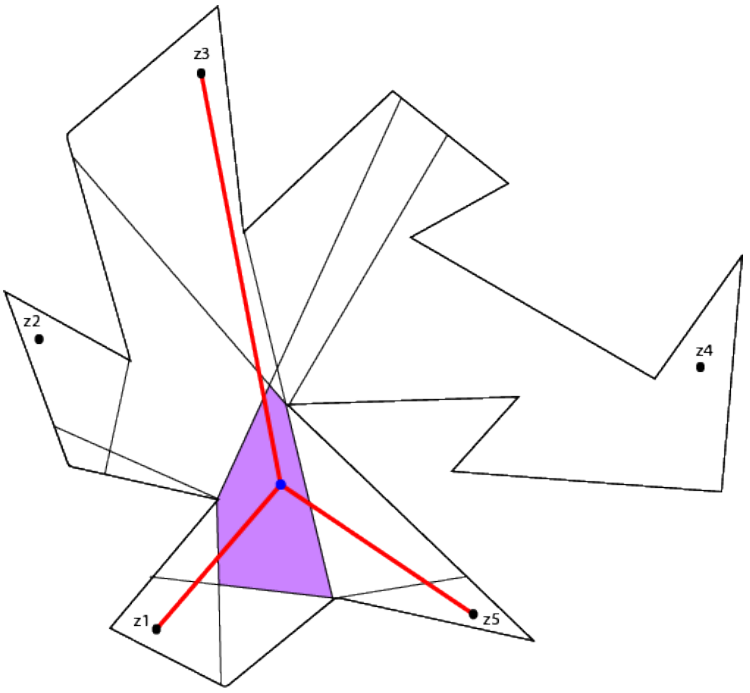


FIGURA 4.16: SMT para los vértices terminales z_1, z_3 y z_5

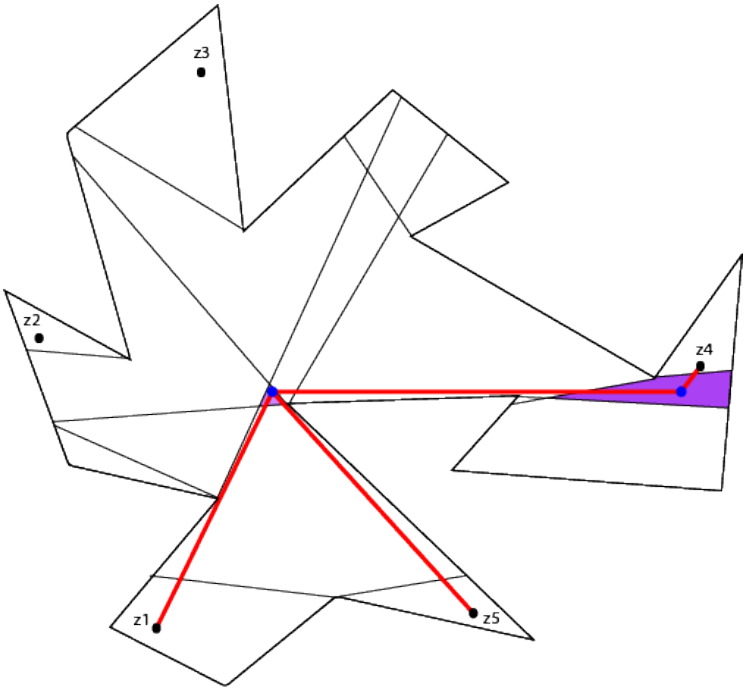


FIGURA 4.17: SMT para los vértices terminales z_1, z_4 y z_5

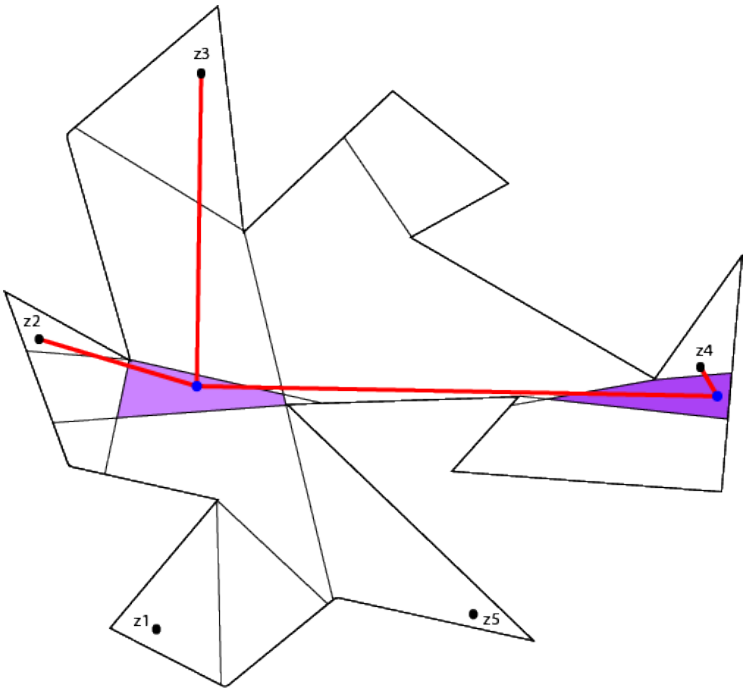


FIGURA 4.18: SMT para los vértices terminales z_2, z_3 y z_4

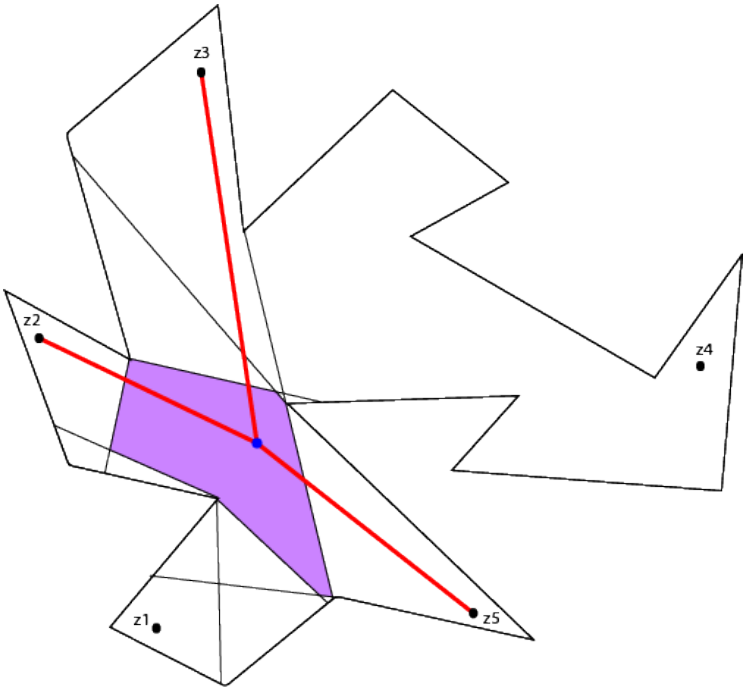


FIGURA 4.19: SMT para los vértices terminales z_2, z_3 y z_5

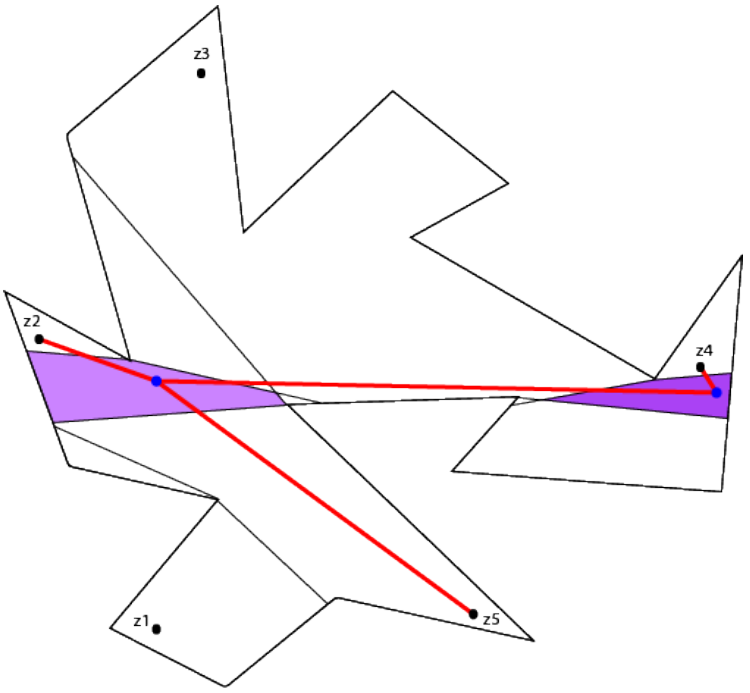


FIGURA 4.20: SMT para los vértices terminales z_2, z_4 y z_5

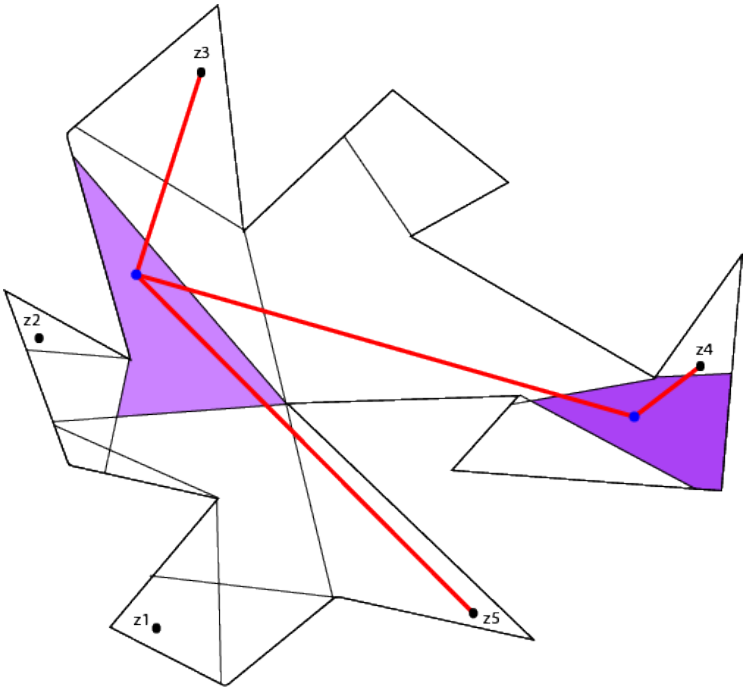


FIGURA 4.21: SMT para los vértices terminales z_3, z_4 y z_5

Resumen de los pesos de cada árbol de Steiner

$$|SMT(\{z_1, z_2\}, P)| = 2 \qquad |SMT(\{z_1, z_2, z_3\}, P)| = 3$$
$$|SMT(\{z_1, z_3\}, P)| = 2 \qquad |SMT(\{z_1, z_2, z_4\}, P)| = 4$$
$$|SMT(\{z_1, z_4\}, P)| = 3 \qquad |SMT(\{z_1, z_2, z_5\}, P)| = 3$$
$$|SMT(\{z_1, z_5\}, P)| = 2 \qquad |SMT(\{z_1, z_3, z_4\}, P)| = 4$$
$$|SMT(\{z_2, z_3\}, P)| = 2 \qquad |SMT(\{z_1, z_3, z_5\}, P)| = 3$$
$$|SMT(\{z_2, z_4\}, P)| = 3 \qquad |SMT(\{z_1, z_4, z_5\}, P)| = 4$$
$$|SMT(\{z_2, z_5\}, P)| = 1 \qquad |SMT(\{z_2, z_3, z_4\}, P)| = 4$$
$$|SMT(\{z_3, z_4\}, P)| = 3 \qquad |SMT(\{z_2, z_3, z_5\}, P)| = 3$$
$$|SMT(\{z_3, z_5\}, P)| = 2 \qquad |SMT(\{z_2, z_4, z_5\}, P)| = 4$$
$$|SMT(\{z_4, z_5\}, P)| = 3 \qquad |SMT(\{z_3, z_4, z_5\}, P)| = 4$$

A partir de estos pesos podemos crear la gráfica de distancia $H_2(Z, P)$ utilizando la columna izquierda.

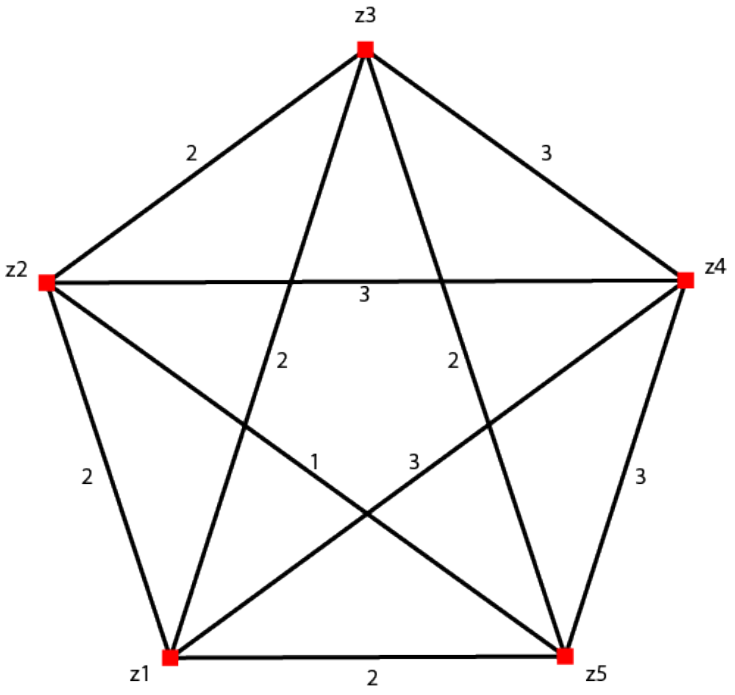


FIGURA 4.22: Gráfica de distancia $H_2(Z, P)$

Paso 2.- Escoger un conjunto de hiperaristas $Y \subset E_r$ tal que $Y_1, Y_2, \ldots, Y_{|Y|}$ sea un árbol de expansión de $H_r(Z, P)$.

Para escoger cuales hiperaristas deben ser parte de nuestro árbol de expansión, se aplicará la ecuación 4.1 para todos los elementos de E_r y se escogerá el mínimo. Recordando la definición de $mst_2(Z, P; Y)$, todos los pesos de las aristas de la gráfica de distancia que conecte dos vértices $zi, zj \in Y$ serán remplazados por 0. A la gráfica de distancia resultante de este proceso se calculará su árbol de expansión mínima. Para ejemplificar este procedimiento, se empezará con el subconjunto $\{z_1, z_2, z_3\}$.

$$f(\{z_1, z_2, z_3\}) = \frac{|SMT(\{z_1, z_2, z_3\}, P)|}{mst_2(Z, P; Y_0) - mst_2(Z, P; Y_0, \{z_1, z_2, z_3\})} \tag{4.2}$$

procedemos a calcular $mst_2(Z, P; Y_0)$ y $mst_2(Z, P; Y_0, \{z_1, z_2, z_3\})$. Como tenemos que $Y_0 = \{\}$ no es necesario cambiar ningún peso de las aristas de $H_2(Z, P)$. Entonces, calcular $mst_2(Z, P; Y_0)$ es un árbol de expansión de la gráfica de la figura 4.22.

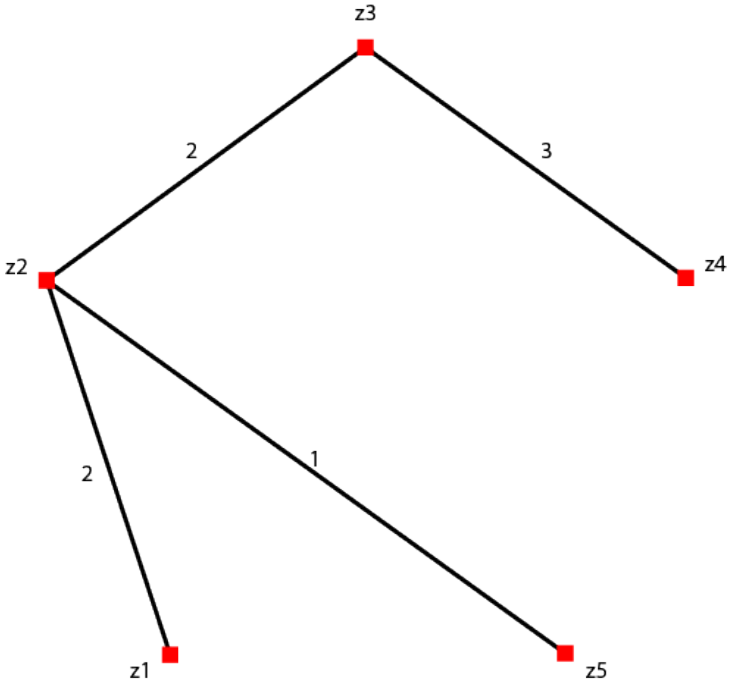


FIGURA 4.23: Un árbol de expansión mínima de la gráfica de distancia $H_2(Z, P)$.

Obtenemos que $mst_2(Z, P; Y_0) = 8$.

Para el caso de $mst_2(Z, P; Y_0, \{z_1, z_2, z_3\})$, sí es necesario cambiar los pesos de las aristas que conectan a este conjunto de vértices, por lo que la gráfica resultante se ve de la siguiente manera:

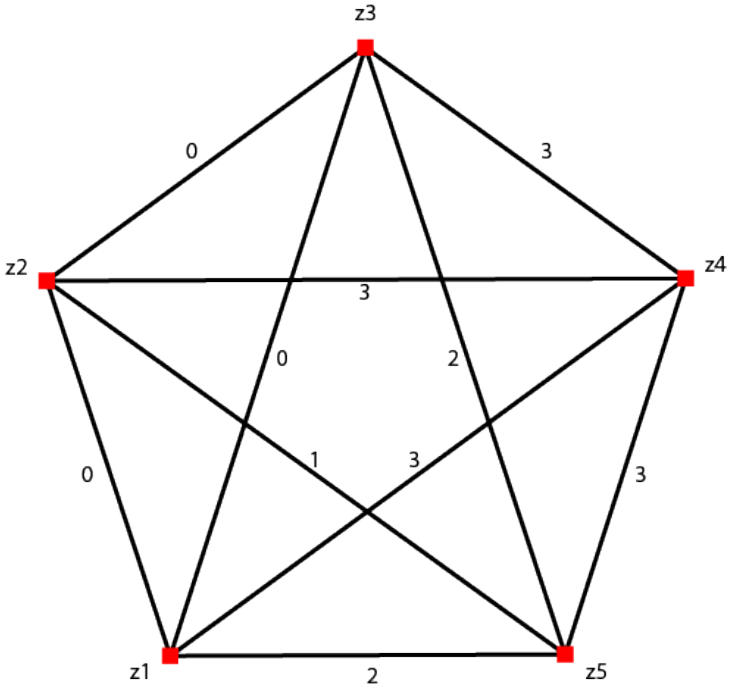


FIGURA 4.24: Gráfica de distancia que define $mst_2(Z, P; Y_0, \{z_1, z_2, z_3\})$.

Se procede a calcular su árbol de expansión mínima.

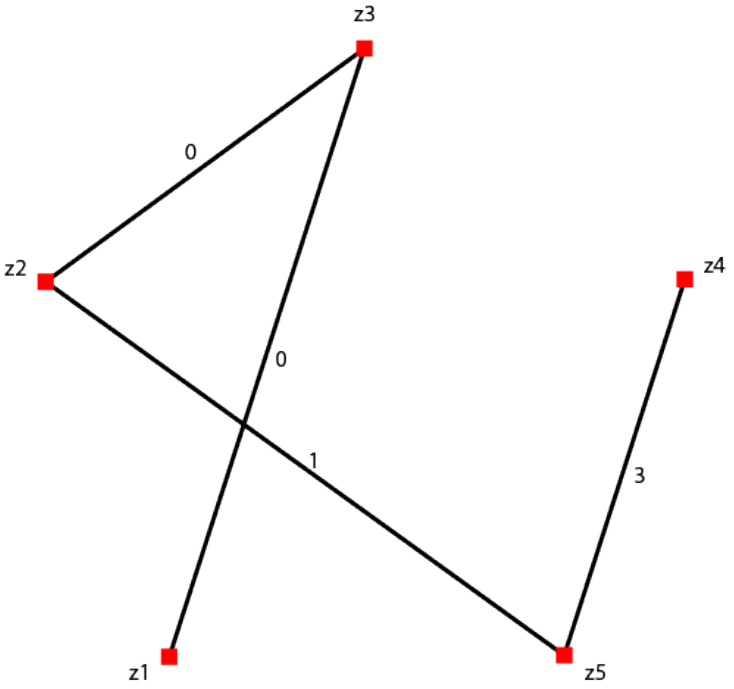


FIGURA 4.25: Un árbol de expansión mínima de la gráfica de la figura 4.24.

Obtenemos que $mst_2(Z, P; Y_0, \{z_1, z_2, z_3\}) = 4$. Con estos resultados ya se puede calcular el valor para la ecuación 4.2.

$$\begin{aligned} f(\{z_1, z_2, z_3\}) &= \frac{|SMT(\{z_1, z_2, z_3\}, P)|}{mst_2(Z, P; Y_0) - mst_2(Z, P; Y_0, \{z_1, z_2, z_3\})} \\ &= \frac{3}{8-4} \\ &= \frac{3}{4} = 0.75 \end{aligned}$$

Aplicando éste mismo proceso para cada uno de las hiperaristas de E_r, se obtienen los siguientes resultados:

$f(\{z_1, z_2\}) = \frac{2}{8-6} = \frac{2}{2} = 1$
$f(\{z_1, z_3\}) = \frac{2}{8-6} = \frac{2}{2} = 1$
$f(\{z_1, z_4\}) = \frac{3}{8-5} = \frac{3}{3} = 1$
$f(\{z_1, z_5\}) = \frac{2}{8-6} = \frac{2}{2} = 1$
$f(\{z_2, z_3\}) = \frac{2}{8-6} = \frac{2}{2} = 1$
$f(\{z_2, z_4\}) = \frac{3}{8-5} = \frac{3}{3} = 1$
$f(\{z_2, z_5\}) = \frac{1}{8-7} = \frac{1}{1} = 1$
$f(\{z_3, z_4\}) = \frac{3}{8-5} = \frac{3}{3} = 1$
$f(\{z_3, z_5\}) = \frac{2}{8-6} = \frac{2}{2} = 1$
$f(\{z_4, z_5\}) = \frac{3}{8-5} = \frac{3}{3} = 1$

$f(\{z_1, z_2, z_3\}) = \frac{3}{8-4} = \frac{3}{4} = 0.75$ $f(\{z_1, z_4, z_5\}) = \frac{4}{8-3} = \frac{4}{5} = 0.80$

$f(\{z_1, z_2, z_4\}) = \frac{4}{8-3} = \frac{4}{5} = 0.80$ $f(\{z_2, z_3, z_4\}) = \frac{4}{8-3} = \frac{4}{5} = 0.80$

$f(\{z_1, z_2, z_5\}) = \frac{3}{8-5} = \frac{3}{3} = 1$ $f(\{z_2, z_3, z_5\}) = \frac{3}{8-5} = \frac{3}{3} = 1$

$f(\{z_1, z_3, z_4\}) = \frac{4}{8-3} = \frac{4}{5} = 0.80$ $f(\{z_2, z_4, z_5\}) = \frac{4}{8-4} = \frac{4}{4} = 1$

$f(\{z_1, z_3, z_5\}) = \frac{3}{8-4} = \frac{3}{4} = 0.75$ $f(\{z_3, z_4, z_5\}) = \frac{4}{8-3} = \frac{4}{5} = 0.80$

De todos esos resultados, tomamos una hiperarista con la función de peso menor. En este caso, podemos tomar $\{z_1, z_2, z_3\}$ o $\{z_1, z_3, z_5\}$. Se tomará a $\{z_1, z_3, z_5\}$ como una nueva hiperarista de Y, es decir, $Y_1 = \{z_1, z_3, z_5\}$.

Repetimos el proceso sin tomar en cuenta la hiperarista $\{z_1, z_3, z_5\}$ ni cualquier hiperarista que sea subconjunto de $\{z_1, z_3, z_5\}$, pero ahora debemos tomar en cuenta que $\{z_1, z_3, z_5\} \in Y$. La ecuación a considerar para la próxima iteración es la siguiente:

$$f(e) = \frac{l_r(e)}{mst_2(Z, P; Y_0, Y_1) - mst_2(Z, P; Y_0, Y_1, e)} \tag{4.3}$$

Los resultados de la función de peso para la nueva iteración son los siguientes:

$f(\{z_1, z_2\}) = \frac{2}{4-3} = \frac{2}{1} = 2$ $f(\{z_1, z_2, z_4\}) = \frac{4}{4-0} = \frac{4}{4} = 1$

$f(\{z_1, z_4\}) = \frac{3}{4-1} = \frac{3}{3} = 1$ $f(\{z_1, z_2, z_5\}) = \frac{3}{4-3} = \frac{3}{1} = 3$

$f(\{z_2, z_3\}) = \frac{2}{4-3} = \frac{2}{1} = 2$ $f(\{z_1, z_3, z_4\}) = \frac{4}{4-1} = \frac{4}{3} = 1.33$

$f(\{z_2, z_4\}) = \frac{3}{4-1} = \frac{3}{3} = 1$ $f(\{z_1, z_4, z_5\}) = \frac{4}{4-1} = \frac{4}{3} = 1.33$

$f(\{z_2, z_5\}) = \frac{1}{4-3} = \frac{1}{1} = 1$ $f(\{z_2, z_3, z_4\}) = \frac{4}{4-0} = \frac{4}{4} = 1$

$f(\{z_3, z_4\}) = \frac{3}{4-1} = \frac{3}{3} = 1$ $f(\{z_2, z_3, z_5\}) = \frac{3}{4-3} = \frac{3}{1} = 3$

$f(\{z_4, z_5\}) = \frac{3}{4-1} = \frac{3}{3} = 1$ $f(\{z_2, z_4, z_5\}) = \frac{4}{4-0} = \frac{4}{4} = 1$

$f(\{z_1, z_2, z_3\}) = \frac{3}{4-3} = \frac{3}{1} = 3$ $f(\{z_3, z_4, z_5\}) = \frac{4}{4-1} = \frac{4}{3} = 1.33$

De los resultados anteriores, se toma la hiperarista de menor peso. En este caso se tomará el conjunto $\{z_1, z_2, z_4\}$ y lo agregamos al conjunto Y, es decir, $Y_2 = \{z_1, z_2, z_4\}$.

Tomando esa hiperarista, se puede observar que el conjunto de hiperaristas Y ya es un árbol de expansión sobre Z, obteniendo la hipergráfica siguiente:

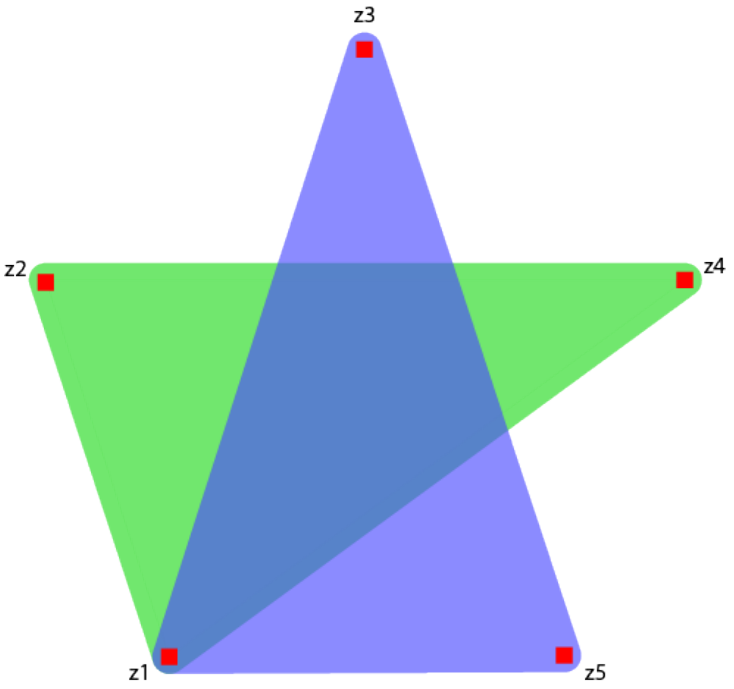


FIGURA 4.26: Árbol de expansión para la hipergráfica $H_r(Z, P)$.

Paso 3.- Convertir la hipergráfica resultante a un árbol de Steiner en aristas en P.

El resultado anterior indica que al unir $SMT(\{z_1, z_3, z_5\}, P)$ (figura 4.16) y $SMT(\{z_1, z_2, z_4\}, P)$ (figura 4.13) es una buena aproximación para $SMT(Z, P)$.

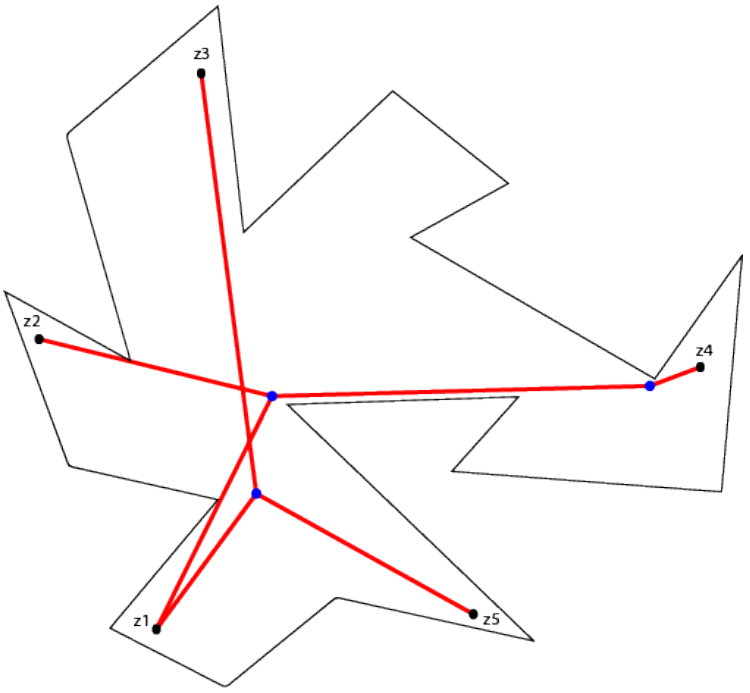


FIGURA 4.27: Árbol de Steiner resultante del algoritmo 4.2.1.

Sin embargo, el árbol de Steiner mínimo para este ejercicio es el siguiente:

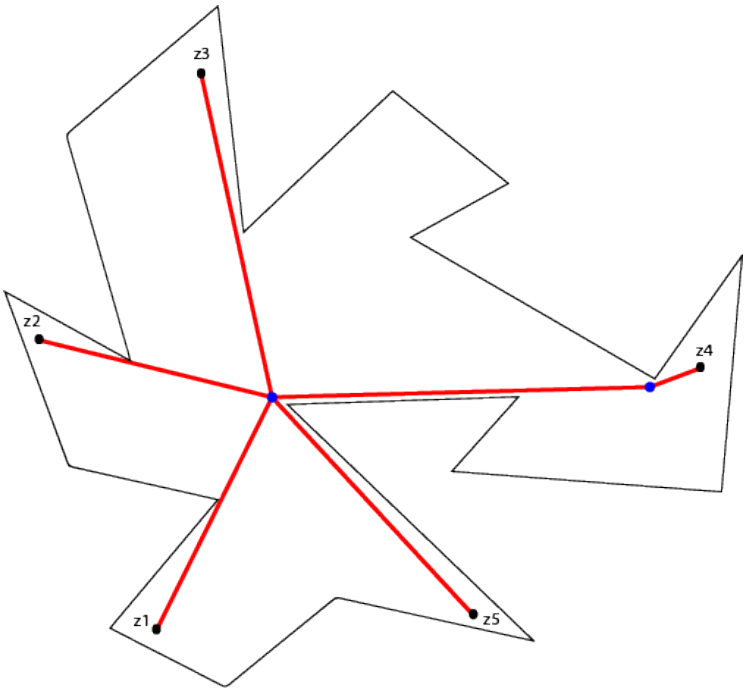


FIGURA 4.28: Árbol de Steiner mínimo del ejemplo de la figura 4.1

Se puede observar que el número de aristas del árbol de Steiner de la figura 4.27 es 7 y el número de aristas del árbol de Steiner mínimo (figura 4.28) es 6.

Utilizando el teorema 3.17, podemos evaluar que tan eficaz es el resultado respecto a su cota superior. $ST(Z, P)$ se referirá al árbol de Steiner regresado por el algoritmo 4.2.1.

$$|ST(Z,P)| \leq (1 + \ln 2) \cdot p_r \cdot smt(N)$$

Se tomará el valor de p_3 de la tabla 3.1, obteniendo:

$$\begin{aligned} |ST(Z,P)| &\leq (1 + \ln 2) \cdot p_3 \cdot smt(N) \\ 7 &\leq 1.69 \cdot 1.67 \cdot 6 \\ 7 &\leq 16.93 \end{aligned}$$

Ahora se calculará la razón de aproximación para el ejemplo anterior.

$$\frac{|ST(Z,P)|}{|SMT(Z,P)|} = \frac{7}{6} = 1.166$$

Conclusiones y trabajo futuro

En este trabajo se desarrolló un algoritmo con factor de aproximación 1.69 de complejidad $O(2^r nr(n+r) + k^{r+1}m^3 \log m)$ para el problema de encontrar un árbol de Steiner mínimo en aristas al interior de un polígono simple. El factor de aproximación de este algoritmo es una mejora importante sobre el factor de aproximación del algoritmo [1, pág. 56] el cual es un algoritmo de 2-aproximación. Este problema ha sido investigado desde el año 2009 por el asesor de ésta tesis [1] y se ha demostrado con anterioridad que es un problema NP-Completo que vive en la clase de los problemas no deterministas polinomiales [19]. Además, se diseñó la subrutina BFS-Ajuste, la cual es una corrección al algoritmo exponencial exacto que podemos encontrar en [1, pág. 46].

Para desarrollar el algoritmo de 1.69-aproximación para el problema del árbol de Steiner mínimo en aristas al interior de un polígono simple, primero fue necesario investigar su equivalente en gráficas [20, pág. 87]. De esta manera, se encontró que el enfoque del algoritmo de aproximación para encontrar un árbol de Steiner mínimo en hipergráficas. Después de investigar éste enfoque, fue posible reducir el problema del árbol de Steiner en polígonos a su equivalente a gráficas, el cual, fue reducido al problema del árbol de expansión mínima en hipergráficas.

Una desventaja de éste algoritmo es que es muy complicado de implementar por la gran cantidad de subrutinas necesarias para su funcionamiento [20, pág. 121]. Sin embargo, fue interesante haber encontrado un algoritmo de aproximación con factor de aproximación menor a dos para resolver éste problema.

Como trabajo futuro, sería interesante investigar diferentes variantes de este problema, por ejemplo, restringir la visibilidad de cada vértice a un cierto ángulo de visión, como la visión de una cámara de seguridad. Otra variación interesante puede ser el equivalente a este problema con polígonos con agujeros. En mi opinión, la variación más compleja sería este problema en su versión tridimensional, es decir, el árbol de Steiner mínimo en aristas dentro de un poliedro no convexo.

Bibliografía

[1] José Sebastián Bejos Mendoza. Árboles de steiner con un número mínimo de aristas en polígonos simples. Master's thesis, UNAM, 2014.

[2] Marcos Cortés Valadez. Un algoritmo exponencial para acoplar un árbol de steiner al interior de un polígono simple. Master's thesis, UNAM, FES Acatlán, 2014.

[3] L. Davis and M. Benedikt. Computational models of space: Isovists and isovist fields. *Computer Graphics and Image Processing*, 11:49–72, 1979.

[4] H. ElGindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2:186–197, 1981.

[5] D. T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics and Image Processing*, 22:207–221, 1983.

[6] S. E. Dorward. A survey of object-space hidden surface removal. *International Journal of Computational Geometry and Applications*, 4:325–362, 1994.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, 3 edition, 2009.

[8] D. Avis and G. T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Transactions on Computers*, 40: 910–1014, 1981.

[9] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[10] S. K. Ghosh, A. Maheshwari, S. P Pal, S. Saluja, and C. E. Madhavan. Computing the shortest path tree in a weak visibility polygon. In Somenath Biswas and Kesav V. Nori, editors, *Foundations of Software Technology and Theoretical Computer Science*. Springer Berlin Heidelberg, 1991.

[11] H. ElGindy. *Hierarchical decomposition of polygons with applications*. PhD thesis, McGill University, Montreal, Quebec, Canada, 1985.

[12] S. Suri. *Minimum link paths in polygons and related problems*. PhD thesis, Johns Hopkins University, Baltimore, USA, 1987.

[13] S. Suri. A linear time algorithm for minimun link paths inside a simple polygon. *Computer Vision, Graphics, and Image Processing*, 5:99–110, 1986.

[14] Mark de Berg, O. Cheong, M. Kreveld, and M. Overmars. *Computational Geometry*. Springer, 3 edition, 2008.

[15] Koshy Thomas. *Catalan Numbers with Applications*. Oxford University Press, 1 edition, 2008.

[16] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.

[17] S. Dreyfus and R. Wagner. The steiner problem in graphs. *Networks*, 1: 195–207, 1972.

[18] I. Tomescu and M. Zimand. Minimum spanning hypertrees. *Discrete Applied Mathematics*, 54:67–76, 1991.

[19] Raúl Eduardo Martínez Chávez. Sobre la complejidad de incrustar árboles de steiner al interior de polígonos simples. Master's thesis, UNAM, FES Acatlán, 2014.

[20] Hans Jürgeb Prömel and Steger Angelika. *The Steiner tree problem - A tour through graphs, algorithms and complexity*. Vieweg, 1 edition, 2002.

[21] W. Johan. An elementary proof of the wallis product formula for pi. *The American Mathematical Monthly*, 114:914–917, 2007.

[22] Al Borchers and Ding-Zhu Du. The k-steiner ratio in graphs. *SIAM Journal on Computing*, 26:857–86, 1997.

[23] Ghosh S.K. *Visibility algorithms in the plane*. Cambridge, 1 edition, 2007.

[24] D. S. Richards, Frank K. Hwang, and James P. Winter. *Steiner Tree Problem, The Annals of Discrete Mathematics, Volume 53*. Elsevier Science & Technology, 1 edition, 1992.

[25] Kratsch Dieter and Fedor V. Fomin. *Exact Exponential Algorithms*. Springer, 1 edition, 2010.

[26] Dingzhu Du, Hung Q. Ngo, Yanhua Li, and X. Cheng. *Steiner Trees in Industry*. Springer, 1 edition, 2002.

[27] Dingzhu Du and Xiaodong Hu. *Steiner Tree Problems in Computer Communication Networks*. World Scientific, 1 edition, 2008.