Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

SOBRE LA COMPLEJIDAD DE INCRUSTAR ÁRBOLES DE STEINER AL INTERIOR DE POLÍGONOS SIMPLES

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

PRESENTA:
RAÚL EDUARDO MARTÍNEZ CHÁVEZ

DIRECTOR DE TESIS:
LIC. JOSÉ SEBASTIÁN BEJOS MENDOZA

Febrero 2014

A mi amada familia

Agradecimientos

En primer lugar, me gustaría agradecer a mis padres Elvia y Fabián, y a mis hermanos Fabián, Adriana y Bibiana, por todo el apoyo que me han brindado durante toda mi formación académica, y que sin ellos no sería lo que soy ahora.

A mi esposa Isela y a mi hija Cinthya, quienes desde el instante en que formaron parte de mi vida han sido la inspiración y el motor que necesito para cumplir con todos mis objetivos.

Agradezco de manera muy especial a Sebastián Bejos, mi asesor y gran amigo, por toda su dedicación y por sus sabios consejos para terminar esta tesis de manera exitosa.

Al Laboratorio de Algoritmos para la Robótica por brindarme un espacio de trabajo en el cual adquirí las herramientas necesarias para el desarrollo de esta tesis.

A Marcos, mi mejor amigo, por entender las ideas que se me ocurren y darles un enfoque distinto para así lograr siempre resultados positivos, además de regalarme su invaluable amistad.

Finalmente, agradezco a todos mis compañeros y amigos quienes siempre me han apoyado en este camino y quienes siempre han estado conmigo en las buenas y en las malas.

Índice

Introducción

Los problemas de geometría computacional provienen de diversas áreas de aplicación tales como, procesamiento de imágenes, visión por computadora, graficación por computadora y robótica. En estas áreas de aplicación se plantean numerosos problemas geométricos para los cuales son necesarios algoritmos eficientes que los resuelvan. La planeación de movimientos es una sub-área de la robótica que se interesa por encontrar trayectorias libres de colisiones para robots móviles. Un caso particular es el siguiente: imaginemos a un robot móvil el cual sólo puede desplazarse hacia adelante y girar, pero este último movimiento causa un efecto negativo sobre el desplazamiento del robot, por ejemplo puede ser que el robot no quede bien orientado, lo que provocaría perdida en la velocidad al intentar orientarse correctamente. Se desea trasladar a este robot de un punto de la casa a otro evitando cualquier colisión con las paredes, y se pretende realizar la tarea en el menor tiempo posible, y entonces minimizando el número de giros.

Dándole una interpretación geométrica a este problema, podríamos representar con un polígono simple P a la habitación (vista desde arriba) en donde se encuentra ubicado el robot y con dos puntos p y q dentro de P representamos a la posición inicial del robot y a su destino, respectivamente. El problema entonces sería, encontrar una trayectoria $\pi(p, q)$ de p a q, minimizando el número de aristas en la trayectoria. A este problema en geometría computacional se le conoce como: *El problema de encontrar la trayectoria mínima en aristas entre dos puntos*. ElGindy en [17] y Suri en [10] fueron los primeros en estudiar este problema. Suri da un algoritmo usando visibilidad débil en [10] para este problema con una complejidad en tiempo de $O(n)$, donde n es el número de vértices del polígono.

Una generalización a este problema es: Dado un conjunto D de puntos en el plano, dentro de un polígono simple P, encontrar un árbol de Steiner[1] mínimo en aristas dentro de P, tal que conecte a todos los puntos en D. En este trabajo estudiamos una variante de este problema, en la que dado un árbol de Steiner ST con k vértices terminales y un polígono simple P con k puntos en su interior, se desea verificar si el árbol de Steiner ST puede ser incrustado dentro de P, tal que sus vértices terminales tengan una correspondencia uno a uno con los puntos interiores de P y sus aristas sean segmentos rectilíneos totalmente contenidos al interior de P.

[1]Un árbol de Steiner es un árbol cuyo conjunto de vértices esta formado por vértices terminales y vértices especiales llamados Steiner.

En este trabajo de investigación se demuestra que dicho problema pertenece a la clase de problemas NP-Completos, la cual es la clase de problemas que se conjeturan que no pueden ser resueltos por algoritmos en tiempo polinomial. Mostrar que un problema pertenece a dicha clase de complejidad es de suma importancia, puesto que al mostrar esto se asume entonces, que el problema no puede ser resuelto en tiempo polinomial.

En el Capítulo 1 mostramos la notación utilizada en este trabajo, así como algunos conceptos fundamentales. En el Capítulo 2 damos una introducción a conceptos un poco más avanzados como el de visibilidad en el plano, que sera fundamental para el desarrollo de este trabajo. En el capítulo 3 se da la definición formal de la clase de complejidad no determinista polinomial, la cual se le conoce como clase de complejidad NP. Finalmente en el Capítulo 5 mostramos que nuestro problema de estudio pertenece a la clase de problemas NP-Completos, haciendo una reducción polinomial a otro problema conocido dentro de la clase.

Capítulo 1

Conceptos básicos

1.1. Gráficas y árboles

Gráficas

Definición 1.1. Una gráfica G es una pareja ordenada (V, E), que consiste de un conjunto V de vértices y un conjunto $E \subseteq (VxV)$ de aristas, las cuales son subconjuntos de dos elementos de V. Ver Figura 1.1.

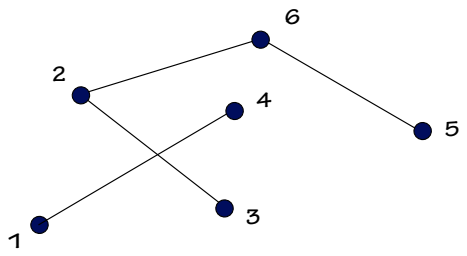


Figura 1.1: Gráfica G con conjunto de vértices $V = \{1, 2, 3, 4, 5, 6\}$ y con conjunto de aristas $E = \{\{1, 4\}, \{2, 3\}, \{2, 6\}, \{6, 5\}\}$

La manera usual de representar una gráfica es dibujando un punto por cada vértice y uniendo dos de esos puntos por una línea recta[1] si los correspondientes dos vértices forman una arista.

Denotaremos al conjunto de vértices de una gráfica G como $V(G)$, y a su conjunto de aristas como $E(G)$. Estas convenciones son independientes de cualquier nombre real de estos dos conjuntos, es decir, el conjunto de vértices de una gráfica $H = (W, F)$ es denotado como $V(H)$ y no como $W(H)$. En una gráfica G el número de vértices es

[1]La manera de como esos puntos y esas lineas son dibujados es irrelevante, lo único importante es saber cuales de esos vértices forman una arista y cuales no.

denotado como $|V(G)|$ y el número de aristas como $|E(G)|$.

Un vértice v es *incidente* con una arista e si $v \in e$. Dos vértices incidentes a un arista e reciben el nombre de *vértices extremos o finales* de e. Por simplicidad representaremos una arista $e = (x, y)$ como xy.

Cuando los extremos de una arista son el mismo vértice diremos que la arista es un *lazo*. Cuando la arista no es un lazo, se le llama *enlace*. Cuando dos aristas tienen los mismos extremos les llamaremos aristas paralelas. Se dice que una gráfica es simple si no presenta lazos ni aristas paralelas. Estas definiciones se ilustran en la Figura 1.2.

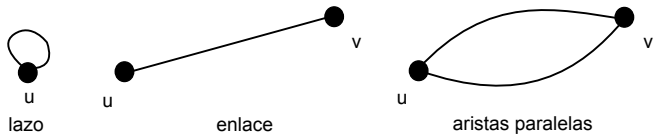


Figura 1.2: Tipos de aristas en una gráfica.

Dos vértices x y y de G son adyacentes, o vecinos, si xy es un arista de G. Dos aristas e y f, $e \neq f$, son adyacentes si tienen un vértice extremo en común. Si todos los vértices de G son adyacentes entre si, entonces G es completa. Una gráfica completa sobre n vértices se representa como K_n; una K_3 es llamada *triángulo*. En la Figura 1.3 se muestran las gráficas completas K_3, K_4, y K_5.

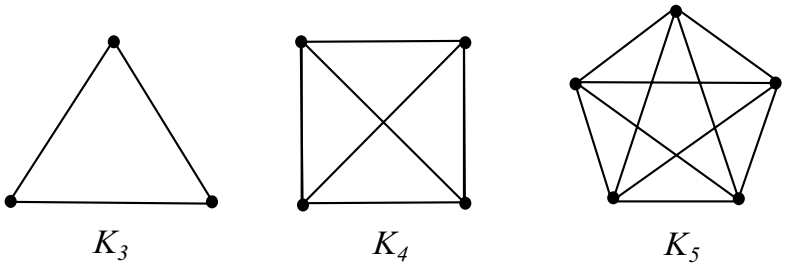


Figura 1.3: Gráficas completas K_3, K_4, y K_5

El *grado* (ó valencia) $d_G(v) = d(v)$ de un vértice v es el número de aristas incidentes a v. Un vértice de grado 0 es llamado vértice *aislado*.

Se dice que una gráfica es *plana* si esta puede ser incrustada en el plano y no existen intersecciones entre sus aristas, el único lugar donde puede haber intersección es en los vértices.

Una *incrustación planar* de una gráfica plana G es una representación topológica de G. En la Figura 1.4 se muestran dos incrustaciones planares de la misma gráfica plana.

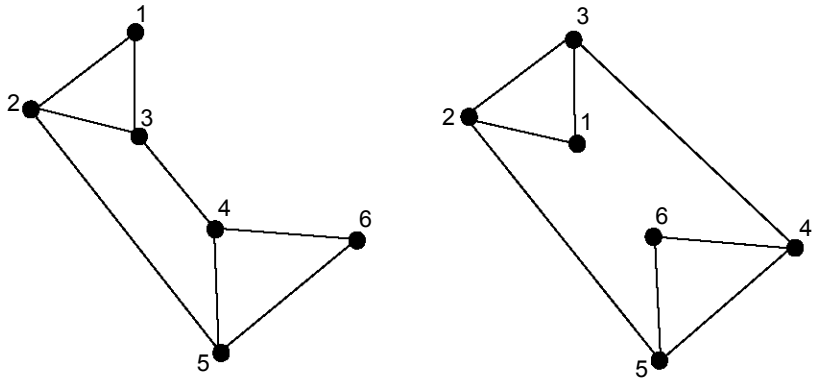


Figura 1.4: Dos incrustaciones planares distintas de una gráfica

Caminos y ciclos

Un *camino* (o trayectoria) es una gráfica no vacía de manera que sus vértices pueden ser ordenados en una secuencia lineal tal que un par de vértices son adyacentes solamente si son consecutivos en la secuencia. De manera formal un camino $P = (V, E)$ tiene conjunto de vértices $V = \{x_0, x_1, x_2, ..., x_k\}$ y conjunto de aristas $E = \{x_0x_1, x_1x_2, ...x_{k-1}x_k\}$, tal que todos los x_i son distintos. Los vértices $x_1, x_2, ..., x_{k-1}$ son llamados los vértices *internos* de P.

Se le llama ciclo a una gráfica tal que los vértices pueden ser ordenados en una secuencia cíclica de tal forma que dos vértices son adyacentes si y sólo si son consecutivos en tal secuencia. Dicho de otra manera, si $P = x_0x_1...x_{k-1}$ es un camino con $k \geq 3$, entonces la gráfica $C = P \cup x_{k-1}x_0$ es llamada *ciclo*, observemos que el vértice inicial y final son el mismo. La *longitud* de un camino o un ciclo esta dada por su número de aristas. El camino de tamaño k es llamado un *k-camino*, así como también un ciclo de tamaño k es llamado un *k-ciclo*.

Subgráficas

Dada una gráfica G, hay dos maneras de derivar gráficas más pequeñas de G. Si e es un arista de G, se puede obtener una gráfica de $m - 1$ aristas eliminando e de G, dejando intactos los vértices y aristas restantes. La gráfica resultante se denota como $G\backslash e$. De manera similar si v es un vértice de G, se puede obtener una gráfica sobre $n - 1$ vértices eliminando de G el vértice v junto con todas las aristas adyacentes con v. La grafica resultante se denota como $G - v$. Estas operaciones de eliminación de aristas y eliminación de vértices son ilustradas en la Figura 1.5.

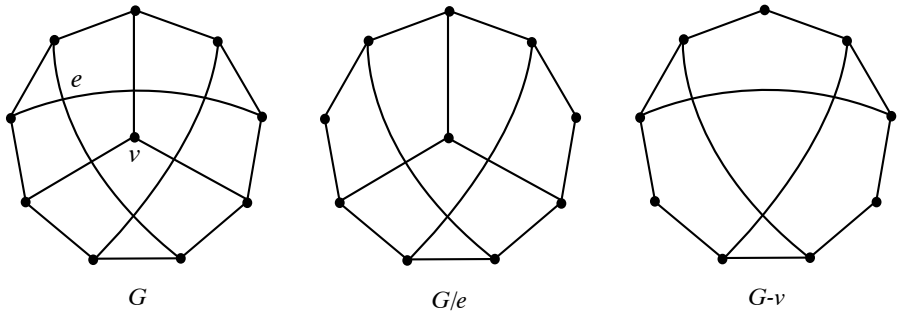


Figura 1.5: Subgráficas de la gráfica de Petersen

Las gráficas $G\backslash e$ y $G - v$ definidas en la Figura 1.5, son ejemplos de subgráficas de G. De manera más general, una gráfica F es llamada una *subgráfica* de una gráfica G si $V(F) \subseteq V(G)$ y $E(F) \subseteq E(G)$.

Definición 1.2. Sea G una gráfica. Se define a un árbol como una subgráfica conectada de G tal que no contiene ningún ciclo. Se denota a un árbol como T y a un árbol de una gráfica G como $T(G)$. A los vértices de grado uno dentro de un árbol T se les llamara hojas[2] de T. En la Figura 1.6 se ilustra esta definición.

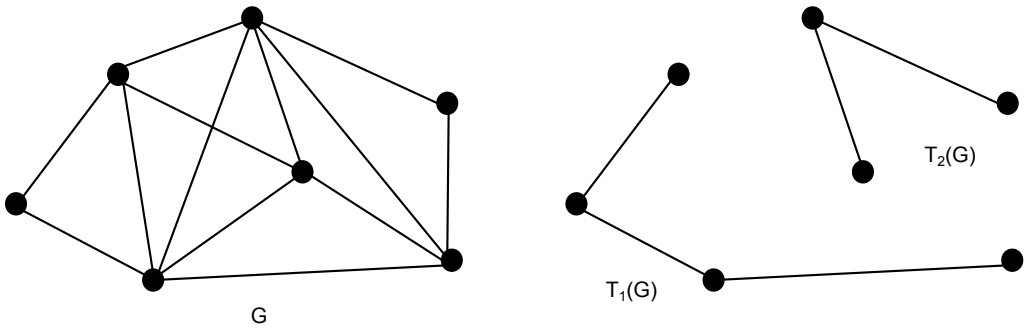


Figura 1.6: Dos árboles, $T_1(G)$ y $T_2(G)$ de la gráfica G.

[2]la raíz de un árbol, incluso si tiene grado uno, nunca es llamada hoja

1.2. Notación Asintótica

Las notaciones que se usan para describir el tiempo de corrida asintótico de un algoritmo, son definidas en términos de funciones cuyos dominios son el conjunto de números naturales $\mathbf{N} = \{0, 1, 2, ...\}$. Tales notaciones son convenientes para describir la función del tiempo de corrida en el peor caso (*worst-case*) $T(n)$, la cual es usualmente definida únicamente sobre entradas de tamaños enteros.

Notación Θ

Sea una función $g(n)$, se denota a $\Theta(g(n))$ como el *conjunto de funciones*:

$\Theta(g(n))$={ $f(n)$ si existen constantes positivas c_1 y c_2, y n_0 tal que
$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para todo $n \geq n_0$}.

Una función $f(n)$ pertenece al conjunto $\Theta(g(n))$ si existen constantes positivas c_1 y c_2, tal que esta pueda ser "encerrada" entre $c_1 g(n)$ y $c_2 g(n)$ para una n suficientemente grande, en la Figura 1.7 se muestra $f(n) = \Theta(g(n))$. Dado que $\Theta(g(n))$ es un conjunto, se puede escribir "$f(n) \in \Theta(g(n))$" para indicar que $f(n)$ es un miembro de $\Theta(g(n))$. También se puede usar "$f(n) = \Theta(g(n))$" para expresar lo mismo, aunque a veces este abuso de igualdad puede causar confusiones.

La definición $\Theta(g(n))$ requiere que cada miembro $f(n) \in \Theta(g(n))$ se **asintóticamente no negativo**, esto es, que $f(n)$ sea no negativa cuando n es lo suficientemente grande. Una función **asintóticamente positiva** es aquella que es positiva para toda n suficientemente grande. Consecuentemente, la función $g(n)$ debe ser por si misma asintóticamente no negativa, o sino el conjunto $\Theta(g(n))$ es vacío. Por lo tanto se asume que toda función usada con notación Θ es asintóticamente no negativa. Esta asunción se mantiene para las otras notaciones asintóticas.

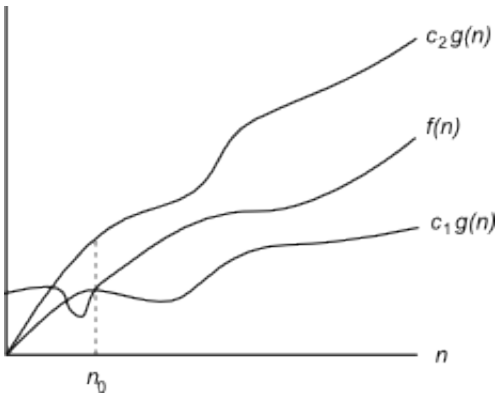


Figura 1.7: $f(n) = \Theta(g(n))$

Notación *O*

La notación Θ acota asintóticamente una función por arriba y por abajo. Cuando se tiene solamente una **cota superior asintótica**, se usa la notación O. Sea una función $g(n)$, se denota a $O(g(n))$ ("oh-grande de g de n") como el conjunto de funciones:

$O(g(n))$={ $f(n)$ si existe una constante positiva c, y n_0 tal que
$0 \leq f(n) \leq cg(n)$ para todo $n \geq n_0$}.

Se usa la notación O para dar una cota superior de una función con un factor constante. Para todos los valores de n a la derecha de n_0, el valor de $f(n)$ esta por abajo de $cg(n)$. Se usa $f(n) = O(g(n))$ para indicar que una función $f(n)$ es un miembro del conjunto $O(g(n))$. En la Figura 1.8 se muestra $f(n) = O(g(n))$.

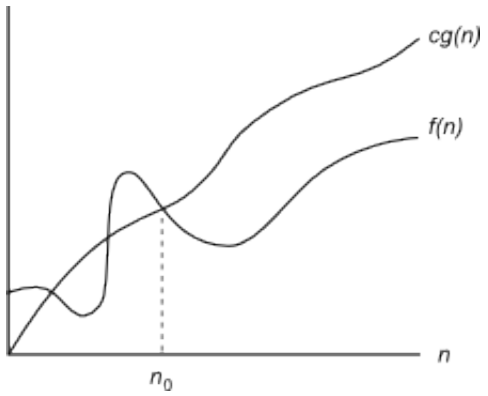


Figura 1.8: $f(n) = O(g(n))$

Notación Ω

Tal como la notación O provee de una cota superior para una función, la notación Ω provee una **cota inferior asintótica**. Sea una función $g(n)$, se denota a $\Omega(g(n))$ ("omega-grande de g de n")como el conjunto de funciones:

$\Omega(g(n))$={ $f(n)$ si existe una constante positiva c, y n_0 tal que
$0 \leq cg(n) \leq f(n)$ para todo $n \geq n_0$}.

La intuición detrás de la notación Ω es mostrada en la Figura 1.9. Para todos los valores de n a la derecha de n_0, el valor de $f(n)$ esta por arriba de $cg(n)$.

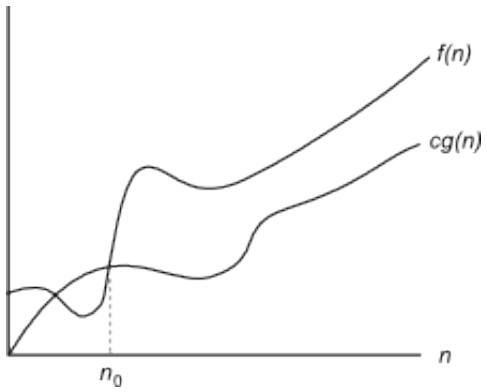


Figura 1.9: $f(n) = \Omega(g(n))$

Por las definiciones de notación asintótica que se han dado, se puede mostrar el siguiente teorema.

Teorema 1.2.1. Para cualesquiera dos funciones $f(n)$ y $g(n)$, se tiene que $f(n) = \Theta(g(n))$ si y solo si $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$.

Demostración. ($\leftarrow$) Por definición tenemos que $O(g(n)) = 0 \leq f(n) \leq cg(n)$ y $\Omega(g(n)) = 0 \leq dg(n) \leq f(n)$, de aquí

$$dg(n) \leq f(n) \leq cg(n)$$

la cual es la definición de $\Theta(g(n))$ (d y c son las constantes c_1 y c_2 que menciona la definición).
($\rightarrow$) Por definición $\Theta(g(n)) = 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, esto es lo mismo que

$$\Theta(g(n)) = f(n)$$

con $c_1 g(n) \leq f(n)$ y $f(n) \leq c_2 g(n)$, lo cual por definición es $\Omega(g(n))$ y $O(g(n))$ respectivamente.

□

Notación *o*

La cota superior proporcionada por la notación O puede o no ser asintóticamente justa. Se usa la notación o para denotar una cota superior que no es asintóticamente justa. Formalmente se define a $o(g(n))$ ("oh-pequeña de g de n") como el conjunto

$o(g(n))$={ $f(n)$ si existe una constante positiva c, y n_0 tal que
$0 \leq f(n) < cg(n)$ para todo $n \geq n_0$}.

Las definiciones de la notación O y de la notación o son similares. La diferencia principal es que en $f(n) = O(g(n))$, la cota $0 \leq f(n) \leq cg(n)$ se mantiene para alguna constante $c > 0$, pero en $f(n) = o(g(n))$, la cota $0 \leq f(n) < cg(n)$ se mantiene para

todas las constantes $c > 0$. Intuitivamente, en la notación o, la función $f(n)$ se vuelve insignificante relativa a $g(n)$ como n se aproxime a infinito; esto es,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

Notación ω

Por analogía, notación ω es a notación Ω como notación o es a notación O. Se usa notación ω para denotar una cota inferior que no es asintóticamente justa. Una manera de definir esto es por

$f(n) \in \omega(g(n))$ si y solo si $g(n) \in o(f(n))$.

Formalmente, sin embargo, se define $\omega(g(n))$ ("omega-pequeña de g de n") como el conjunto

$\omega(g(n))$={ $f(n)$ si existe una constante positiva c, y n_0 tal que
$0 \leq cg(n) < f(n)$ para todo $n \geq n_0$}.

La relación $f(n) = \omega(g(n))$ implica que

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

si el limite existe. Esto es, $f(n)$ se vuelve arbitrariamente grande relativa a $g(n)$ como n se aproxime al infinito.

Capítulo 2

Visibilidad en el plano

2.1. Polígonos y visibilidad

Definición 2.1. Un *polígono* P es una región cerrada R en el plano, la cual es acotada por un conjunto finito de aristas de P, denotadas como $E(P)$. Cualquier punto extremo de un arista de P es llamado vértice de P, el cual es un punto en el plano.

Se dice que un polígono P es un *polígono simple* si existe una trayectoria entre cuales quiera dos puntos de R la cual no intersecte con algún arista de P.

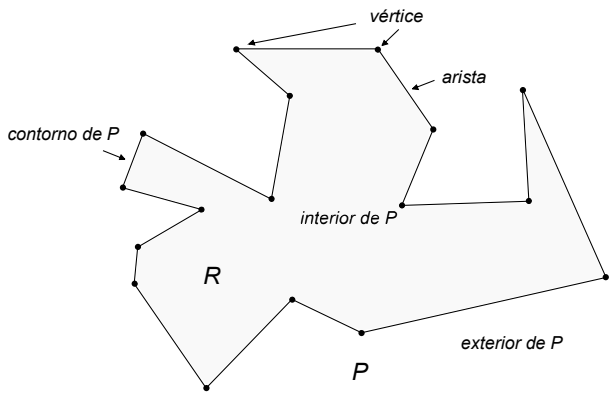


Figura 2.1: Polígono Simple P

Dado que P es una región cerrada y acotada, el *contorno* (o limite) de P consiste de un ciclo de aristas de P.

La región R es llamada la región interna o el interior de P. De manera similar, las regiones del plano que excluyen todos los puntos de R son llamados las regiones externas o el exterior de P. En la Figura 2.1 se ilustran estas definiciones.

Se dice que dos puntos p y q dentro de P son *visibles* si el segmento de línea que une a p y q no contiene algún punto del exterior de P. Esto significa que el segmento pq se encuentra totalmente contenido en el interior de P. Se puede decir también que p *ve* a q, si p y q son visibles en P, es obvio que si p ve a q, q ve a p, entonces en ocasiones se dirá que p y q son visibles mutuamente. En la Figura 2.2, el punto p ve al punto q pero no al punto r.

.

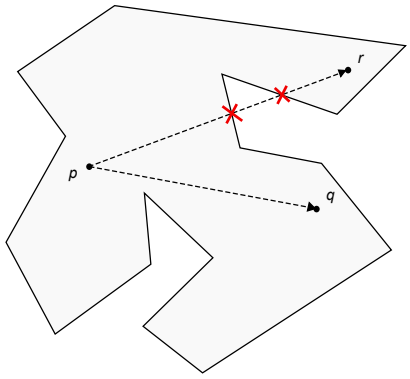
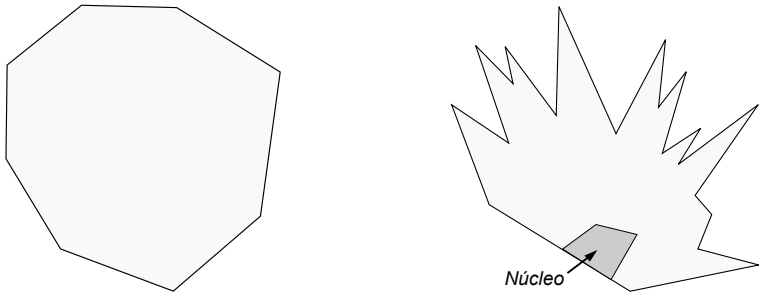


Figura 2.2: Visibilidad de entre dos puntos

Usando la definición de visibilidad en un polígono, se definen dos clases especiales de polígonos, los polígonos *convexos* y los polígonos *estrella* (en inglés *star-shaped*). Un polígono simple P es llamado *convexo* si cada par de puntos en P es mutuamente visible, ver Figura 2.3(a). Un polígono simple P es llamado *estrella* si existe un punto z dentro de P tal que todos los puntos de P son visibles desde z. Ver Figura 2.3(b). Al conjunto de todos los puntos z de P se les conoce como el *núcleo* de P.



(a) Polígono Convexo C. (b) Polígono Estrella S.

Figura 2.3: Dos tipos de polígonos simples.

2.2. Triangulación

Definición 2.2. Sea $\overline{ab}$ un segmento rectilineo dentro de un polígono P, si: 1) a y b son vértices de P, 2) a y b son visibles mutuamente, y 3) $\overline{ab} \notin E(P)$, se dice que $\overline{ab}$ es una

diagonal de P.

Definición 2.3. Una *triangulación* de un polígono P es una partición de P en triángulos por diagonales.

Una triangulación de P no es única debido a que muchos subconjuntos de diagonales pueden dar una triangulación al polígono P.

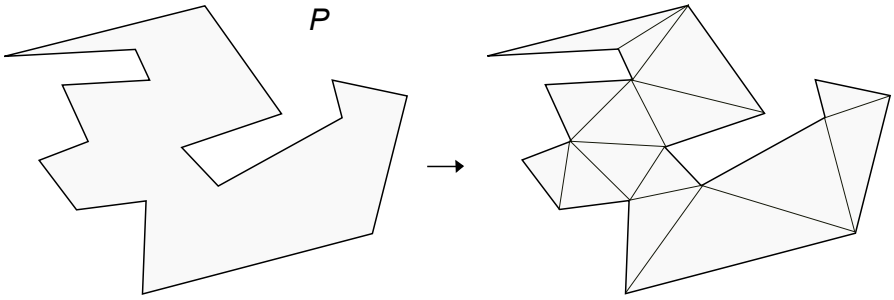


Figura 2.4: Una triangulación del polígono P.

Teorema 2.2.1. Toda triangulación de un polígono simple de n vértices tiene $n-2$ triángulos.

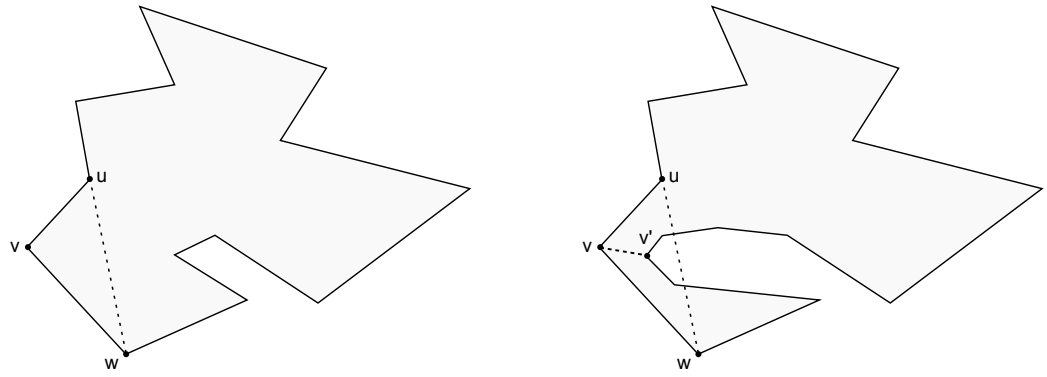
Demostración. Este teorema sera probado por inducción en n. Cuando $n = 3$ el polígono es por si mismo un triángulo, y por lo tanto el teorema es trivialmente verdadero. Sea $n > 3$, y asumimos que el teorema es verdadero para $m < n$. Sea P un polígono con n vértices. El primer paso a realizar es demostrar la existencia de una diagonal en P. Sea v el vértice más a la izquierda de P (en caso de un empate, utilizar el vértice más a la izquierda y más abajo) y sean u y w los vecinos de v sobre el contorno de P. Si el segmento $\overline{uw}$ cae en el interior de P, entonces una diagonal ha sido encontrada. De otra manera, hay uno o más vértices dentro del triángulo definido por u, v y w, o sobre la diagonal $\overline{uw}$. De todos estos vértices, sea v' el vértice más lejano de la linea que pasa por u y w. El segmento que conecta a v' y v no puede intersectar una arista de P, debido a que dicha arista tendría uno de sus vértices extremos al interior del triángulo más alejado de la línea que pasa por u y w, contradiciendo la definición de v'. Por lo tanto $\overline{vv'}$ es una diagonal.

Se dice entonces que una diagonal existe. 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 deben ser menores a n, entonces, por inducción P_1 y P_2 pueden ser triangulados. Por lo tanto P puede ser triangulado.

Lo que resta probar ahora es que cualquier triangulación de P consiste de $n-2$ triángulos. Para esto se considerara una diagonal arbitraria en alguna triangulación T_P. Esta diagonal corta a P en dos subpolígonos con m_1 y m_2 vértices respectivamente. Todo vértice de

P se encuentra en exactamente uno de los dos subpolígonos, excepto para los vértices definidos en la diagonal, ya que cada vértice se encuentra en ambos subpolígonos. Por lo tanto $m_1+m_2 = n+2$. Por inducción, cualquier triangulación de P_i consiste de m_i-2 triángulos, lo cual implica que T_P consiste de $(m_1-2)+(m_2-2) = (m_1+m_2)-4 = n+2-4 = n-2$ triángulos.

□



(a) Caso 1: $\overline{uw}$ esta totalmente contenida dentro P.

(b) Caso 2: $\overline{uw}$ no esta totalmente contenida dentro de P, pero si lo esta $\overline{vv'}$.

Figura 2.5: Prueba de la existencia de una diagonal en P.

2.3. Problema de la Galería de Arte

Este problema fue introducido por Klee en 1973 en una discusión con Chvátal. La idea básica del problema es la siguiente, imaginemos la sala de un museo que contiene varias obras de arte, el cual visto desde arriba puede ser representado por un polígono P de n vértices. Klee se preguntó: ¿Cuántas cámaras son necesarias para vigilar la sala entera? Cada cámara es considerada como un punto fijo que puede ver en todas direcciones, esto es, tiene una visibilidad de 2π o $360°$. Claro que las cámaras no ven a través de las paredes de la sala.

De manera más concisa, el problema es encontrar el mínimo número de cámaras necesarias que puedan cubrir a todo el polígono P.

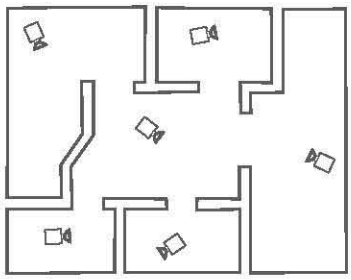
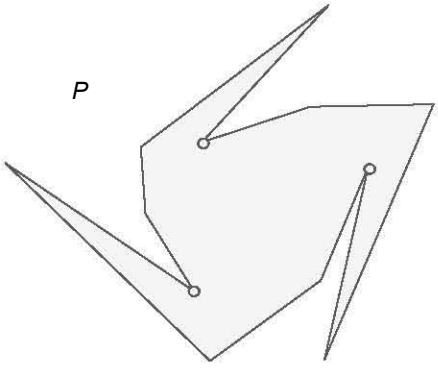
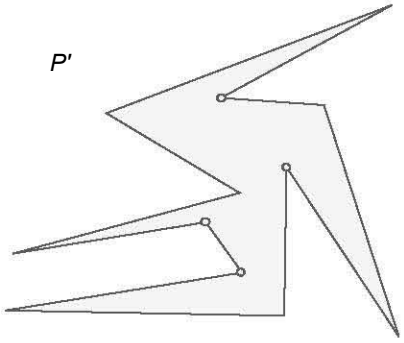


Figura 2.6: Representación de una galeria de arte

Para algún polígono P, hay un número mínimo de cámaras que son necesarios para completar el cubrimiento. En la Figura 2.7(a) es claro que tres cámaras son necesarios para cubrir el polígono P de doce vértices, aunque hay una libertad considerable en la ubicación de las tres cámaras. Pero no siempre son suficientes 3 cámaras para cubrir la totalidad del polígono. Por ejemplo en la Figura 2.7(b) se muestra un polígono del mismo número de vértices y son necesarias 4 cámaras para vigilarlo totalmente. Lo que la pregunta de Klee busca es expresar como una función de n el número más pequeño de cámaras que son suficientes para cubrir cualquier polígono de n vértices. Algunas veces este número se dice que es necesario y suficiente para el cubrimiento: necesario para algunos polígonos ya que si se toman menos cámaras, el polígono ya no es cubierto en su totalidad, y suficiente para algunos polígonos ya que si se toman menos cámaras, se sigue cubriendo a todo el polígono. Por ejemplo, cuatro cámaras son necesarias para cubrir al polígono de la Figura 2.7(b) pero son suficientes para el polígono de la Figura 2.7(a).



(a) Tres puntos (cámaras) son necesarios para cubrir el polígono P de 12 vértices.



(b) Cuatro puntos (cámaras) son necesarios para cubrir el polígono P' de 12 vértices.

Figura 2.7: Puntos neesarios para el cubrimiento de los polígonos P y P'.

Formalizando el problema. Sea $g(P)$ el número más pequeño de cámaras necesarias para cubrir el polígono P: $g(P) = min_S|S : S$ cubre a $P|$, donde S es un conjunto de puntos (cámaras) y $|S|$ es la cardinalidad de S. Sea P_n un polígono de n vértices. $G(n)$ es

el máximo de $g(P_n)$ sobre todos los polígonos de n vértices: $G(n) = max_{P_n} g(P_n)$. El problema de Klee es determinar la función $G(n)$. Al menos una cámara es siempre necesaria, ya que no es posible cubrir a un polígono con cero cámaras. En términos de la notación usada, esto genera una cota inferior sobre $G(n) : 1 \leq G(n)$. Es evidente que n cámaras son suficientes para cualquier polígono: colocando una cámara en cada vértice se cubrirá el polígono. Esto da una cota superior: $G(n) \leq n$. Entonces ahora sabemos que $1 \leq G(n) \leq n$.

El Teorema 2.2.1 implica que un polígono simple con n vértices puede ser vigilado o cubierto con $n - 2$ cámaras. Pero poner una cámara por cada triángulo es un poco exagerado. Una cámara posicionada en una diagonal por ejemplo, puede vigilar dos triángulos, entonces colocando cámaras en diagonales bien seleccionadas podría reducir el número de cámaras a $n/2$. Colocar las cámaras en los vértices suena un poco mejor, ya que un vértice puede ser adyacente a muchos triángulos, y una cámara en ese vértice podría vigilar a todos esos triángulos. Esto sugiere la siguiente aproximación.

Sea T_P una triangulación de P. Seleccionar un subconjunto de vértices de P, tal que cualquier triángulo de T_P tenga al menos un vértice seleccionado, es decir si un triángulo es $v_0v_1v_2$ al menos v_i, para alguna $i = \{0, 1, 2\}$, pertenecerá al subconjunto de vértices de P mencionado anteriormente, y colocar las cámaras en los vértices seleccionados. Para encontrar dicho subconjunto se asignara a cada uno de los vértices de P un color: blanco, gris y negro. La coloración debe ser tal, que cuales quiera dos vértices adyacentes a una misma arista o a una misma diagonal deberán tener colores diferentes, a esto se le conoce como *3-coloración* de un polígono triangulado. En un 3-coloración de un polígono triangulado, todo triángulo tiene un vértice blanco, gris y negro. Por lo tanto, si se colocan cámaras en todos los vértices grises, se dice, que el polígono es vigilado completamente. Si se escoge la clase de color mínima para colocar las cámaras, se puede vigilar P usando a lo más $\lfloor n/3 \rfloor$ cámaras. En la Figura 2.8 se muestra esta asignación.

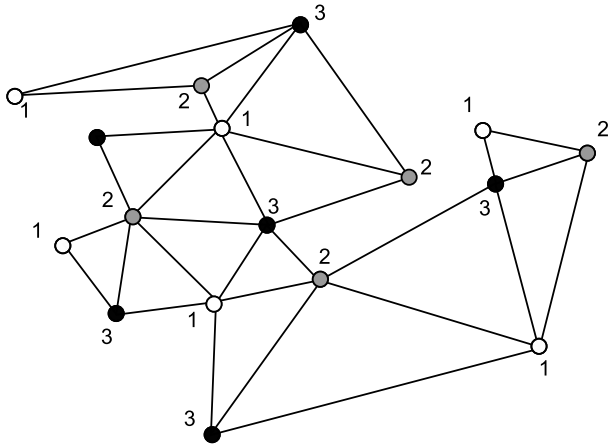


Figura 2.8: 3-coloración del polígono P, si se colocan cámaras en los vértices de color gris (2), el polígono es vigilado completamente con $\lfloor \frac{17}{3} \rfloor = 5$ cámaras.

Ahora la pregunta sería, ¿siempre existe un 3-coloración?. La respuesta es si. Para ver esto, se revisará la *gráfica dual* de T_P. Esta gráfica, denotada como $\zeta(T_P)$, tiene un vértice por cada triángulo en T_P. Se denotará al triángulo correspondiente a un vértice v de $\zeta(T_P)$ como $t(v)$. Si $t(v)$ y $t(\mu)$ comparten una diagonal en T_P, entonces en la gráfica dual $\zeta(T_P)$ existirá la arista $v\mu$ (ver Figura 2.9). Las aristas en $\zeta(T_P)$ corresponden a las diagonales en T_P. Dado que cualquier diagonal corta al polígono P en dos, si se elimina una arista de $\zeta(T_P)$ se parte a la gráfica en dos. Por lo tanto $\zeta(T_P)$ es un árbol. Esto significa que se puede encontrar una 3-coloración haciendo un simple recorrido de la gráfica $\zeta(T_P)$, con búsqueda en profundidad por ejemplo.

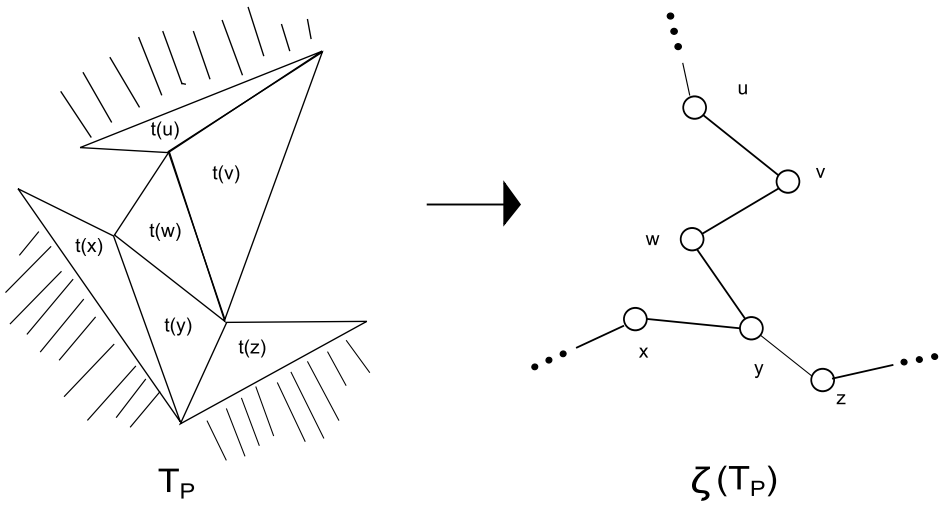


Figura 2.9: Gráfica dual $\zeta(T_P)$ de la triangulación T_P.

A continuación se explica como lograr eso. Mientras se va haciendo la búsqueda en profundidad, se mantendrá la siguiente invariante: todos los vértices de los triángulos encontrados hasta el momento tendrán colores blanco, gris y negro, y dos vértices conectados en la misma arista no tendrán el mismo color. La invariante implica que se puede calcular una 3-coloración válida cuando todos los triángulos han sido encontrados. La búsqueda en profundidad puede ser iniciada en cualquier vértice de $\zeta(T_P)$; los tres vértices del triángulo encontrado serán coloreados como blanco, gris y negro. Ahora supóngase que un vértice v ha sido alcanzado en $\zeta(T_P)$, viniendo de un vértice μ. Entonces $t(v)$ y $t(\mu)$ comparten una diagonal. Dado que los vértices de $t(\mu)$ ya han sido coloreados, solamente un vértice de $t(v)$ falta por ser coloreado. Solo hay un color que puede ser asignado a este vértice, llámese el color i que no ha sido asignado a los vértices de la diagonal entre $t(v)$ y $t(\mu)$. Dado que $\zeta(T_P)$ es un árbol, los otros vértices adyacentes a v no han sido visitados aún, por lo tanto se mantiene la libertad de asignarle al vértice v el color i que restaba. Este procedimiento se puede observar en la Figura 2.10.

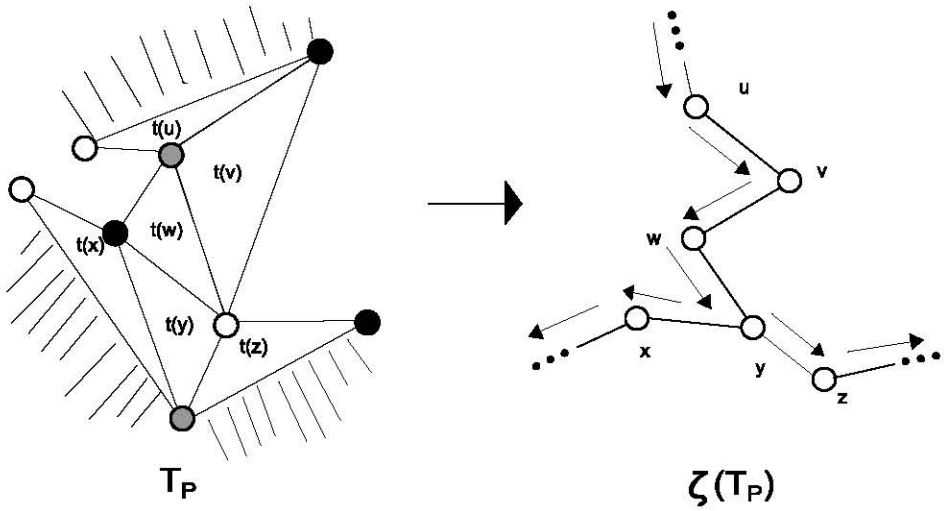


Figura 2.10: Haciendo recorrido en profundidad sobre $\zeta(T_P)$ para colorear T_P.

Se concluye entonces que un polígono simple triangulado puede ser siempre 3-coloreado.

Como un resultado se tiene que un polígono simple puede ser vigilado con $\lfloor n/3 \rfloor$ cámaras. Claro que esto puede mejorar ya que una cámara colocada en un vértice puede cubrir a más que solo triángulos adyacentes. En la Figura 2.11 podemos observar que hay polígonos con n vértices los cuales requieren $\lfloor n/3 \rfloor$ cámaras para ser cubiertos, por lo que no se puede esperar encontrar una estrategia que siempre encuentre menos de $\lfloor n/3 \rfloor$ cámaras, lo que nos indica que 3-coloración es una aproximación óptima para el peor de los casos.

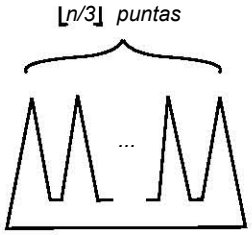


Figura 2.11: Ejemplo de que $\lfloor n/3 \rfloor$ cámaras son necesarias para cubrir completamente el polígono de n vértices.

Lo anterior prueba el siguiente teorema.

Teorema 2.3.1. (Teorema de la galeria de arte) Para un polígono simple P con n vértices, $\lfloor n/3 \rfloor$ cámaras son ocasionalmente necesarias y siempre suficientes para poder ver a cada punto del polígono simple.

2.4. Polígono de Visibilidad

La noción de visibilidad débil de un polígono desde un segmento fue introducido por Avis and Toussaint para el problema de la Galería de Arte. Ellos consideraron una variación del problema: solo hay una cámara, la cual es móvil y puede desplazarse a lo largo de una arista del polígono. Ellos definieron la visibilidad desde una arista v_iv_{i+1} en un polígono simple P en 3 formas diferentes:

1. Se dice que P es de *visibilidad completa* desde v_iv_{i+1} si cada punto $z \in P$ es visible a todo punto $w \in v_iv_{i+1}$.

2. Se dice que P es de *visibilidad fuerte* desde v_iv_{i+1} si existe un punto $w \in v_iv_{i+1}$ tal que para cada punto $z \in P$, w y z son visibles.

3. Se dice que P es de *visibilidad débil* desde v_iv_{i+1} si cada punto $z \in P$, existe un punto $w \in v_iv_{i+1}$ (dependiendo de z) tal que w y z son visibles.

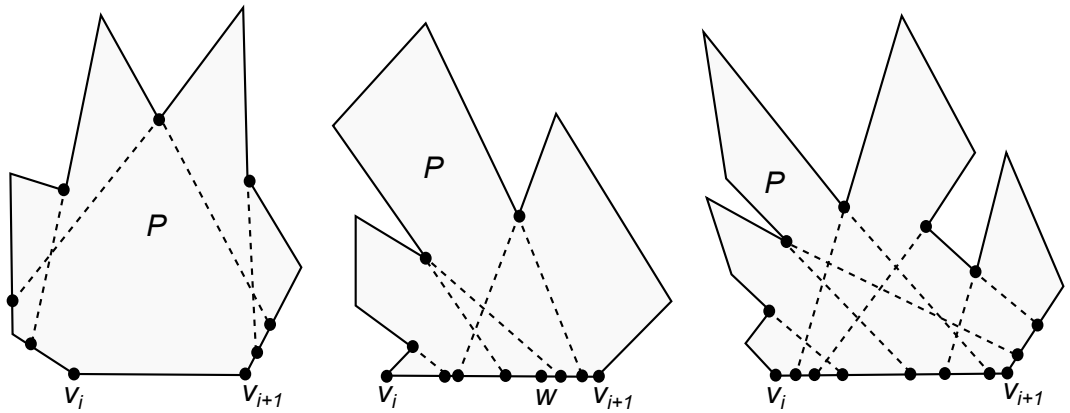


Figura 2.12: El polígono P es (a) completamente, (b) fuertemente y (c) débilmente visible de la arista v_iv_{i+1}

Si P es completamente visible desde v_iv_{i+1}, el guardia puede colocarse en cualquier punto w sobre 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 la misma región. Si P es fuertemente visible desde v_iv_{i+1}, existe al menos un punto w sobre v_iv_{i+1} desde el cual el guardia pueda ver el polígono completo P, es decir, $P = V(w)$. Finalmente, si P es débilmente visible desde v_iv_{i+1}, es necesario que el guardia patrulle a lo largo de v_iv_{i+1} para ver completamente a P.

Un punto $z \in P$ es dicho a ser *débilmente visible* de una arista v_iv_{i+1} si este es visto desde algún punto de v_iv_{i+1}. El conjunto de todos los puntos de P que son débilmente visibles desde v_iv_{i+1} es llamado el *polígono de visibilidad débil* de P desde

$v_i v_{i+1}$. Una noción más general de un polígono de visibilidad débil, permite la visibilidad de P a partir de un segmento interno pq, el cual no es necesariamente una arista de P.

Se considerara ahora el problema de calcular el polígono de visibilidad completa desde un segmento de línea pq en un polígono P. Considerar algún punto $z \in P$ que es visible desde los extremos p y q. Dado que P es simple entonces z es completamente visible desde pq. El siguiente lema sugiere una manera de calcular el polígono de visibilidad completa desde un segmento pq.

Lema 2.4.1. El polígono de visibilidad completa de una línea pq de un polígono P, es el polígono de visibilidad de q dentro del polígono de visibilidad de P a partir de p.

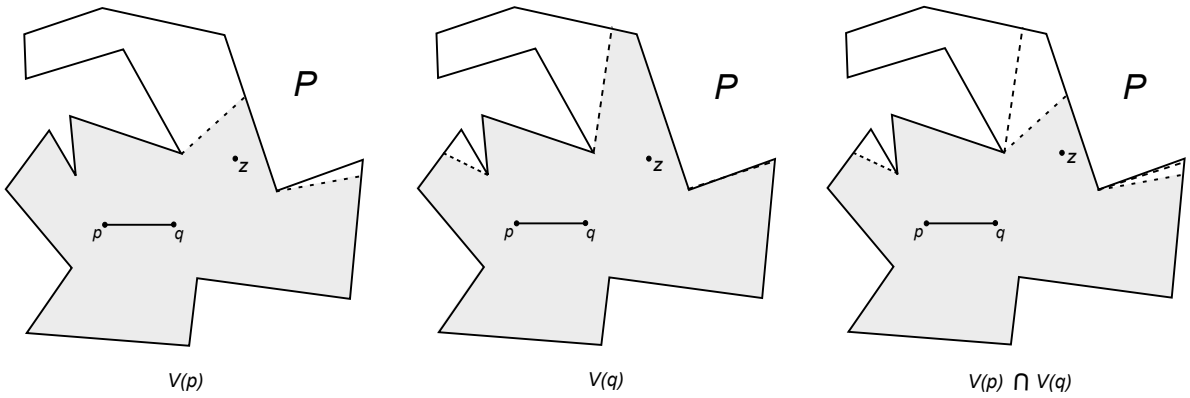


Figura 2.13: Prueba gráfica del Lema 2.4.1.

Demostración. Sea $V(p)$ el polígono de visibilidad de p en P, y $V(q)$ el polígono de visibilidad de q en P. Sea $z \in P$ un punto en el plano visible desde p. Dado que z es visible desde p, entonces $z \in V(p)$, por definición, para que z sea completamente visible desde la línea pq, z también tiene que ser visible desde q, por lo tanto, $z \in V(q)$. La única manera de lograr esto es que $z \in V(p) \cap V(q)$, ya que de otra manera $z \in V(p)$ y $z \notin V(q)$ o viceversa. Por lo tanto el polígono de visibilidad completa de pq en P sera el conjunto de puntos $z \in V(p) \cap V(q)$, lo que es equivalente a obtener el polígono de visibilidad $V(q)'$ de q en $V(p)$.

□

Capítulo 3

Problemas completos de complejidad no determinista polinomial

3.1. Teoría de problemas completos de complejidad no determinista polinomial (NP-completos)

La clase de complejidad P.

Se define a un *problema abstracto* Q como una relación binaria sobre el conjunto I de instancias del problema y el conjunto S de soluciones del problema.

Por ejemplo, el problema de la trayectoria mínima Euclidiana, es una relación $R \subseteq I \times S$ que asocia a cada tripleta $(G, u, v) \in I$, donde G es una gráfica y $u, v \in V(G)$, con cada solución $S(u, v) \in S$.

Un *problema de decisión* es una función $Q : I \to \{0, 1\}$, donde I es el conjunto de instancias del problema. Por ejemplo el problema $PATH$ es el problema de decisión tal que si $i = (G, u, v, k) \in I$, entonces $PATH(i) = 1$ si la trayectoria mínima de u a v tiene a lo más k aristas y $PATH(i) = 0$ en cualquier otro caso.

La teoría de problemas NP-completos está descrita para problemas de decisión y en lo que sigue se enfocara a esta clase de problemas.

Teoría formal de lenguajes

Un *alfabeto*, denotado como Σ, es un conjunto finito no vacío de símbolos. Una *palabra* es una sucesión finita de símbolos de algún alfabeto Σ. Un *lenguaje* L sobre Σ es un conjunto de palabras sobre Σ. Por ejemplo, si $\Sigma = \{0, 1\}$, el conjunto

$L = \{10, 11, 101, 111, 1011, 1101, 10001, \ldots\}$ es lenguaje de los números primos sobre el alfabeto binario $\{0, 1\}$.

Se denota como ϵ a la palabra vacía, como $\emptyset$ al lenguaje vacío y Σ^* al lenguaje de todas las palabras en el alfabeto Σ. Por ejemplo, si $\Sigma = \{0, 1\}$ entonces $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 11, 000, \ldots\}$, el conjunto de todas las cadenas binarias.

Todo lenguaje L sobre Σ es subconjunto Σ^*. El complemento de un lenguaje L, denotado como $\bar{L}$ está dado por:

$$\bar{L} = \Sigma^* - L$$

La concatenación L_1L_2 de dos lenguajes L_1 y L_2 es el lenguaje

$$L = \{x_1x_2 : x_1 \in L_1, x_2 \in L_2\}.$$

La cerradura o estrella de Kleene de un lenguaje L es el lenguaje

$$\Sigma^* = \{\epsilon\} \cup L \cup L^2 \cup L^3 \cup \ldots$$

, donde L^k es el lenguaje obtenido de concatenar L consigo mismo K veces.

Una codificación e para un problema de decisión Q es una descripción de cada instancia de Q como un palabra sobre un alfabeto Σ.

Un problema de decisión Q bajo una codificación $e(Q)$ sobre el alfabeto Σ particiona Σ^* en tres clases de palabras.

1) Palabras que no son codificación de ninguna instancia de Q.
2) Palabras x tales que $Q(w) = 0$ y,
3) Palabras x tales que $Q(w) = 1$.

Dado que un problema de decisión Q está totalmente caracterizado por aquellas instancias de problema tales que producen un 1, se puede definir el problema Q como un lenguaje L sobre un alfabeto Σ donde

$$L = \{x \in \Sigma^* : Q(x) = 1\}$$

Por ejemplo el problema de decisión $PATH$ es equivalente al siguiente lenguaje.

$$PATH = \{\langle G, u, v, k\rangle : G = (V, E) \text{ es una gráfica no dirigida,}$$
$$u, v \in V(G),$$
$$k \geq 0 \text{ es un entero,}$$
$$\text{y existe una trayectoria de } u \text{ a } v \text{ en } G$$
$$\text{de longitud } k\}.$$

Donde $\langle I\rangle$ denota la codificación de la instancia.

Se dice que un algoritmo *acepta* una palabra $x \in \Sigma^*$ si, x dada como entrada, la salida del algoritmo $A(x)$ es 1.

Un lenguaje *aceptado* por un algoritmo A es el conjunto de palabras

$$L = \{x \in \Sigma^* : A(x) = 1\}$$

Un algoritmo *rechaza* una palabra x si $A(x) = 0$.

Un lenguaje L es *decidido* por un algoritmo A si toda palabra $x \in L$ es aceptada por A y toda palabra $y \notin L$ es rechazada por A.

Un lenguaje L es aceptado en tiempo polinomial por un algoritmo A si existe una constante k tal que para toda palabra $x \in \Sigma^*$ con $|x| = n$ el algoritmo decide correctamente si $x \in L$ es tiempo $O(n^k)$.

Una clase de complejidad es un conjunto de lenguajes cuya pertenencia a la clase está determinada por una medida de complejidad[1] de un algoritmo que determina si una palabra pertenece al lenguaje L.

La clase de complejidad P, se define de la siguiente manera:

$P = \{L \subseteq \sum^* :$ existe un algoritmo A que decide L en tiempo polinomial$\}$.

La clase de complejidad NP

Un ciclo Hamiltoniano de una gráfica $G = (V, E)$ es un ciclo que contiene a todo vértice en $V(G)$, y solo pasa por el una sola vez. Una gráfica que contiene un ciclo Hamiltoniano se dice que es una gráfica Hamiltoniana y una que no es una gráfica no Hamiltoniana.

Sea *HAM-CYCLE*$= \{\langle G\rangle : G$ es una gráfica Hamiltoniana$\}$

Un algoritmo para decidir *HAM-CYCLE* sería poner en una lista toda permutación de los vértices de G y verificar cada una, hasta encontrar una permutación que sea un

[1] una medida de complejidad con respecto algún criterio, como por ejemplo el tiempo de corrida

ciclo en G. El cual no es un algoritmo polinomial.

Considérese ahora que además de la gráfica G se nos da como entrada una sucesión de los vértices de G. Se podría verificar que ésta sucesión de vértices induce un ciclo Hamiltoniano en G en tiempo polinomial.

Un *algoritmo verificador* es un algoritmo A con dos argumentos de entrada, el primer argumento es la entrada ordinaria dada como una palabra $x \in \Sigma^*$ que es la codificación de una instancia del problema que resuelve, y el segundo argumento es otra palabra $y \in \Sigma^*$ llamada certificado.

Un algoritmo verificador A *verifica* una palabra de entrada x si existe un certificado y tal que $A(x, y) = 1$.

El lenguaje verificado por un algoritmo verificador A es

$$L = \{x \in \Sigma^* : \text{ existe } y \in \Sigma^* \text{ tal que } A(x, y) = 1\}$$

Intuitivamente, un algoritmo A verifica un lenguaje L si para toda palabra $x \in L$ existe un certificado y que A puede utilizar para probar que $x \in L$. Más aún para toda palabra $x \notin L$ no existe un certificado que pruebe que $x \in L$.

Por ejemplo para $HAM-CYCLE$ el certificado es la codificación de un ordenamiento de los vértices en G de algún ciclo Hamiltoniano.

Si la gráfica no es Hamiltoniana, no existe un ordenamiento de los vértices en G que pueda engañar al algoritmo verificador para hacerle creer que es Hamiltoniana dado que el algoritmo verificador checa si el certificado propuesto es un ciclo Hamiltoniano.

La clase de complejidad NP es la clase de lenguajes que pueden ser verificados por un algoritmo polinomial, esto es, un lenguaje L pertenece a NP si y sólo si existe un algoritmo verificador A y una constante c tal que

$$L = \left\{ x \in \sum^{*} : \text{ existe un certificado } y \text{ con } |y| = O\left(|x|^c\right) \text{ tal que } A(x, y) = 1 \right\}$$

Se dice entonces que el algoritmo A verifica el lenguaje L en tiempo polinomial. Por la discusión anterior se puede decir que *HAM-CYCLE* $\in NP$

Teorema 3.1.1. $(P \subseteq NP)$ Sea L un lenguaje. Si $L \in P$ entonces $L \in NP$

Demostración. Dado que si existe un algoritmo polinomial para decidir L, el algoritmo se puede convertir fácilmente en un algoritmo verificador que simplemente ignore cualquier

certificado en su segundo argumento y acepte exactamente aquellas palabras que determine pertenece a L.

□

Problemas NP-completos

Una función $f : \Gamma \subseteq \Sigma^* \to \Sigma^*$ es computable en tiempo polinomial si existe un algoritmo polinomial A que dada una entrada $x \in \Gamma$ devuelve $f(x) \in \Sigma^*$.

Un lenguaje L_1 es reducible en tiempo polinomial a un lenguaje L_2, denotado como $L_1 \leq_p L_2$, si existe una función computable en tiempo polinomial $f : \Gamma \subseteq \Sigma^* \to \Sigma^*$ tal que para toda $x \in \Gamma$, $x \in L_1$ si y sólo si $f(x) \in L_2$.

La función f es llamada función de reducción y el algoritmo polinomial F que calcula f es un algoritmo de reducción.

Si $x \in L_1$ entonces $f(x) \in L_2$ y además si $x \in L_1$ entonces $f(x) \notin L_2$. Por lo tanto, la función de reducción mapea cualquier instancia x del problema de decisión tal que $Q_1(x) = 1$ en el lenguaje L_1, a una instancia $f(x)$ del problema de decisión tal que $Q_2(f(x)) = 1$ en el lenguaje L_2.

Si podemos saber que $f(x) \in L_2$, entonces directamente se podría saber si $x \in L_1$ por lo que las reducciones en tiempo polinomial son una herramienta poderosa para probar que un lenguaje pertenece a P.

Lema 3.1.1. Si $L_1, L_2 \in \sum^*$ son lenguajes tales que $L_1 \leq_p L_2$, entonces $L_2 \in P$ implica $L_1 \in P$.

Demostración. Sea A_2 un algoritmo polinomial que decide L_2 y sea F un algoritmo de reducción polinomial que calcule la función de reducción f. Sea A_1 el siguiente algoritmo polinomial. Para una entrada $x \in \Gamma \subseteq \sum^*$, el algoritmo A_1 corre el algoritmo F para transformar x a $f(x)$ y luego corre A_2 para verificar si $f(x) \in L_2$. Si $A_2(f(x)) = 1$ entonces $A_1(x) = 1$ y si $A_2(f(x)) = 0$ entonces $A_1(x) = 0$. A_1 es correcto porque f es una función de reducción para $L_1 \leq_p L_2$ y es un algoritmo polinomial porque F y A_2 son algoritmos polinomiales.

□

Un lenguaje $L \subseteq \Sigma^*$ es *NP-completo* si:

i) $L \in NP$

ii) $L' \leq_p L$ para todo $L' \in NP$

Se definira a NPC como la clase de lenguajes NP-completos.

Teorema 3.1.2. Si algún problema NP-completo puede ser resuelto en tiempo polinomial, entonces $P = NP$.

Demostración. Supongamos que $L \in P$ y también $L \in NPC$. Para cualquier $L' \in NP$ tenemos que $L' \leq_p L$ por la propiedad 2 de la definición de NP-completo. Además por el Lema reflemaant tenemos que $L' \in P$

□

Corolario 3.1.1. Si algún problema en NP no puede ser solucionado en tiempo polinomial entonces ningún problema NP-completo puede ser solucionado en tiempo polinomial.

$$P \to Q$$
$$\neg Q \to \neg P$$

Un lenguaje $L \subseteq \Sigma^*$ es NP-duro si $L' \leq_p L$ para todo $L' \in NP$.

Lema 3.1.2. La relación $\leq P$ sobre lenguajes es transitiva, esto es, si $L_1 \leq_p L_2$ y $L_2 \leq_p L_3$ entonces $L_1 \leq_p L_3$.

Sean f y g las funciones de reducción de las reducciones polinomiales $L_1 \leq_p L_2$ y $L_2 \leq_p L_3$, entonces la función $g \circ f$ es una función de reducción polinomial para la reducción $L_1 \leq_p L_3$.

Lema 3.1.3. Si L es un lenguaje tal que $L' \leq_p L$ para algún lenguaje tal que $L' \in NPC$, entonces L es NP-duro. Si además, $L \in NP$, entonces $L \in NPC$.

Demostración. Dado que $L' \in NPC$, para todo lenguaje $L'' \in NP$ tenemos que $L'' \leq_p L'$. Por suposición $L' \leq_p L$,y por transitividad de la relación $\leq_p$ sobre lenguajes tenemos que $L'' \leq_p L$, por lo tanto L es NP-duro. Si $L \in NP$, incluso se tiene que $L \in NPC$.

□

Este lema da un método para probar que un lenguaje L es NP-completo.

1) Probar que $L \in NP$.
2) Seleccionar un lenguaje L' que sabemos es NP-completo.
3) Describir un algoritmo A que calcule una función f que mapee toda instancia $x \in \Gamma \subseteq \sum^*$ de L' a una instancia $f(x)$ de L.
4) Prueba que la función f satisface $x \in L'$ si y sólo si $f(x) \in L$ para toda $x \in \Gamma \subseteq \sum^*$.

5) Prueba que A es polinomial.

Teorema 3.1.3. Satisfacibilidad de fórmulas booleanas en 3-forma normal conjuntiva es un problema NP-completo.

La prueba al Teorema 3.1.3 se da en [6]. En las siguientes dos secciones se darán dos problemas los cuales serán mostrados a ser NP-completos, con el fin de ejemplificar el método dado por el Lema 3.1.

3.2. Problema del árbol de Steiner en gráficas

Definición del problema

El problema del árbol de Steiner en gráficas (ver Figura 3.1), llamado por simplicidad ST, es definido en su forma de decisión como sigue:

Instancia: Una gráfica no dirigida $G = (V, E)$; un subconjunto de vértices $R \subseteq V$, llamados vértices terminales; y un número $k \in \mathbb{N}$

Pregunta: ¿Existe un subárbol de G que incluya todos los vértices de R (es decir un árbol de expansión para R) y que contenga a lo más k aristas ?.

Este problema tiene muchas aplicaciones, especialmente cuando se tiene que planificar una estructura de conexión entre puntos terminales diferentes. Por ejemplo, cuando se desea encontrar una manera optima de construir carreteras y vías de ferrocarril que conectan un conjunto de ciudades o decidir las políticas de enrutamiento a través de Internet para el tráfico multicast, usualmente de una fuente a muchos destinos.

Desafortunadamente este problema ha sido mostrado a ser intratable, en el sentido de que no existe algoritmo polinomial que lo resuelva, a menos que $P = NP$. A continuación se mostrara que este problema es NP-completo.

ST es NP-Completo

Teorema 3.2.1. El problema del árbol de Steiner en gráficas es NP-Completo.

Demostración. Para mostrar que ST es NP-completo, es necesario mostrar que $ST \in NP$. Asumir que $(G, R, k) \in ST$, es decir que (G, R, k) es una instancia de ST, y asumir que la instancia reserva una respuesta afirmativa. En este caso dada una solución hipotética positiva $T \subseteq G$, se puede verificar en tiempo polinomial que:

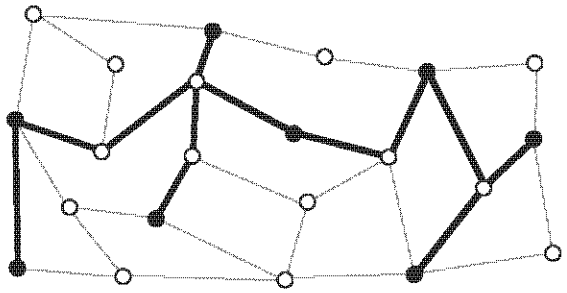


Figura 3.1: Ejemplo de un árbol de Steiner en gráficas. Los círculos de color negro representan el conjunto de vértices R, y las aristas en negrita representan el árbol de Steiner T.

- T es un árbol, es decir que no contiene ciclos y es conexo,
- el árbol T conecta a todos los vértices terminales especificados por el conjunto R, y
- el número de aristas usadas por el árbol no es mayor que k.

Una vez mostrado que $ST \in NP$ lo que sigue ahora es elegir un problema conocido a ser NP-Completo. Se elegirá el problema de 3-Satisfacibilidad Booleana (3-SAT).

Problema: 3-Satisfacibilidad Booleana (3-SAT)

Instancia: Un conjunto $U = \{u_1, u_2, ..., u_h\}$ de variables booleanas y una colección $C = \{c_1, c_2, ..., c_m\}$ de cláusulas sobre U tal que toda $c_i \in C$, con $i = 1, 2, ..., m$, es una disyunción de precisamente tres literales.

Pregunta: ¿Existe una valuación de las variables de C, de tal manera que la fórmula $I = c_1 \wedge c_2 \wedge c_3 \wedge ... \wedge c_m$ es verdadera?

Sean $x_1, x_2, ..., x_n$ las variables y $c_1, c_2, ..., c_m$ las cláusulas en una instancia arbitraria de $3 - SAT$. El objetivo es construir una gráfica $G = (V, E)$, un conjunto de terminales R, y una cota B, tal que G contiene un árbol de Steiner T que expande a R de tamaño a lo más B, si y solo si la instancia dada de $3 - SAT$ es satisfacible.

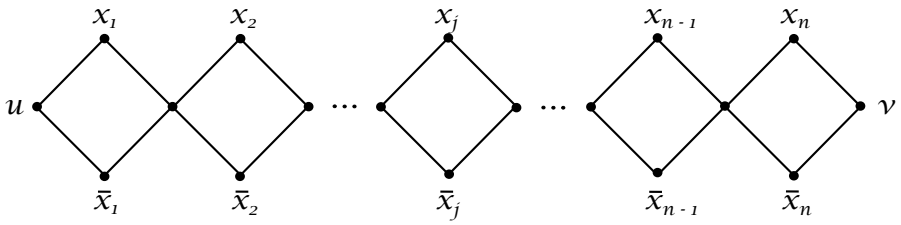


Figura 3.2: Transformando $3-SAT$ al problema del árbol de Steiner en gráficas: la trayectoria de variables.

La gráfica G es construida como sigue. Conectar dos vértices u y v por una *trayectoria de variables* como es mostrado en la Figura 3.2.

Crear para toda cláusula c_i un dispositivo de cláusula (ver Figura 3.3) que consiste de un vértice c_i conectado por trayectorias de longitud t con sus correspondientes literales en la trayectoria de variables. Como conjunto de terminales de escogerá $R = \{u, v\} \cup \{c_1, c_2, ..., c_m\}$ y hacer a B como $B = 2n + t \cdot m$.

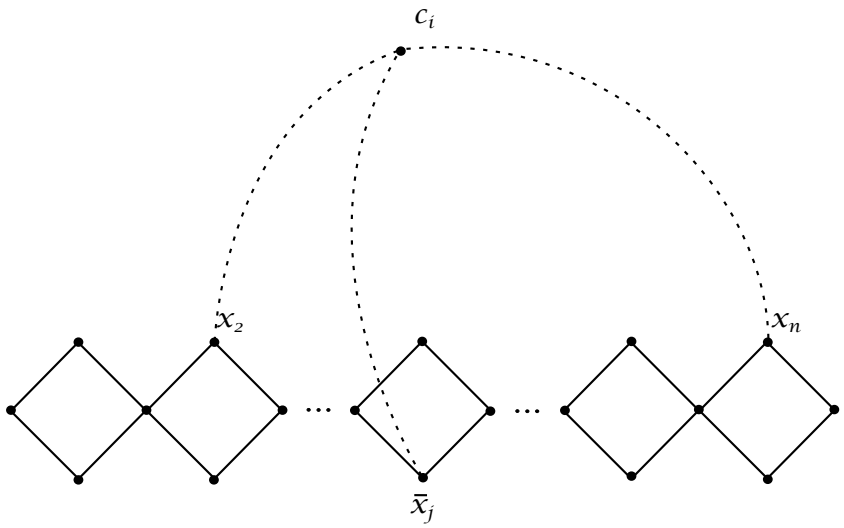


Figura 3.3: El dispositivo de cláusulas para la cláusula $c_i = x_2 \vee \overline{x_j} \vee x_n$. Las lineas punteadas indican trayectorias de tamaño t de c_i a los vértices apropiados en la trayectoria de variables.

Asumir primero que la instancia de $3-SAT$ es satisfacible. Para construir un árbol de Steiner que expanda a R se comenzará con una trayectoria P de u a v que represente una asignación satisfactoria. Esto es, $x_i \in P$ si x_i es establecida como verdadera en la asignación y $\overline{x_i} \in P$ en otro caso. Se puede observar que para toda cláusula el vértice c_i puede ser conectado a P por una trayectoria de longitud t como se puede observar en la Figura 3.4. De esta forma se obtiene un árbol de Steiner para R de longitud $2n+t\cdot m = B$.

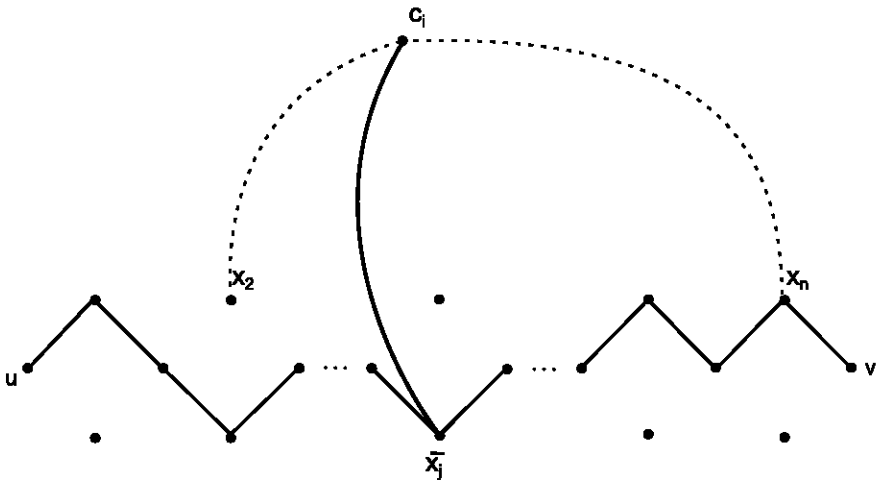


Figura 3.4: La trayectoria de u a v representa una asignación satisfactoria para la fórmula a la que c_i pertenece. La cláusula c_i puede ser conectada a la trayectoria ya que al menos una de sus literales debe pertenecer a ella.

Para ver la otra dirección, asumir ahora que T es un árbol de Steiner para R de tamaño a lo más B. De manera trivial, para cada cláusula el vértice c_i ha sido conectado a la trayectoria de variables. Asumir por el momento que existe una cláusula c_j para la cual c_j esta conectada a la trayectoria de variables por al menos dos de estas trayectorias. Entonces $|E(T)| \geq (m+1) \cdot t > B$. Esto muestra que u y v pueden ser solamente conectadas en la trayectoria de variables, el cual requiere de al menos $2n$ aristas. Como cada cláusula necesita al menos t aristas para conectar a c_i a la trayectoria de variables se concluye que la trayectoria $u - v$ contiene *exactamente* $2n$ aristas y que cada dispositivo de cláusula esta conectado a esta trayectoria usando exactamente t aristas. Por lo tanto, la trayectoria $u - v$ representa una asignación satisfactoria.

Se puede ver que esta construcción es obtenida en tiempo polinomial: por cada cláusula se construye un dispositivo en tiempo constante $O(1)$ por lo tanto el problema del árbol de Steiner en gráficas es NP-completo. □

3.3. Problema de la incrustación planar de árboles al interior de polígonos simples

Se dice que una gráfica $G = (V, E)$, de tamaño $|V| = n$, puede ser *incrustada* (con líneas rectas) dentro de un conjunto de n puntos P si existe un mapeo uno a uno $\psi : V \to P$ de los vértices de G a los puntos de P. Se dice que esta incrustación es una *incrustación planar* si las aristas de G solamente se intersectan en los vértices, esto es, $(\psi(u_1), \psi(v_1)) \cap (\psi(u_2), \psi(v_2)) = \emptyset$, para todo $(u_1, v_1) \neq (u_2, v_2) \in E$.

Teorema 3.3.1. Sea P un conjunto de n puntos en el plano al interior de un polígono

simple S y sea T un árbol de n vértices. Decidir si existe un incrustación planar recta de T dentro del conjunto de puntos P tal que las aristas de T no se intersectan con el contorno del polígono S es NP-Completo.

Demostración. Es claro que el problema pertenece a NP. Dado un incrustamiento planar recto de un árbol T dentro de un conjunto de puntos P se puede probar en tiempo polinomial, en el número de vértices de T y en el número de vértices de S, si el incrustamiento es planar y no se intersecta con el contorno de S. Para mostrar la NP-Completez, la reducción es del problema *3-partición*.

Problema: 3-Partición.

Entrada: Un número natural B, y $3n$ números naturales $a_1, a_2, ..., a_{3n}$, donde $B/4 < a_i < B/2$, $i = 1, 2, ..., 3n$.

Salida: n conjuntos disjuntos $S_1, ..., S_n \subset \{a_1, a_2, ..., a_{3n}\}$, donde $|S_j| = 3$ y $\sum_{a \in S_j} a = B$, para toda S_j, $j = 1, 2, ..., n$.

Se puede observar que como $B/4 < a_i < B/2$, no tiene sentido tener conjuntos S_j con menos o más de tres elementos. Esto es, es equivalente a preguntar por las subdivisiones de todos los elementos en conjuntos disjuntos, tal que sumen a B. De hecho se puede asumir que $\sum_{i=1}^{3n} a_i = Bn$, de otra forma seria imposible que alguna solución existiera. Dada una instancia de 3-partición, se construira el siguiente árbol T (ver Figura 3.5).

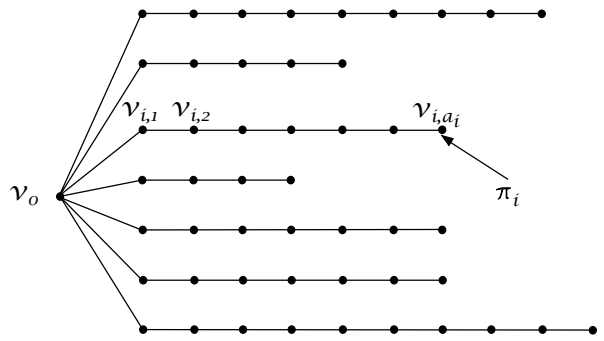


Figura 3.5: Árbol T para la reducción NP-Completo

- Por cada número natural a_i en la entrada, hacer una trayectoria π_i que consista de a_i vértices $v_{i,1}, v_{i,2}, ..., v_{i,a_i}$.

- Considerar un vértice adicional v_0 para T. Por cada trayectoria π_i conectar el vértice $v_{i,1}$ al vértice v_0.

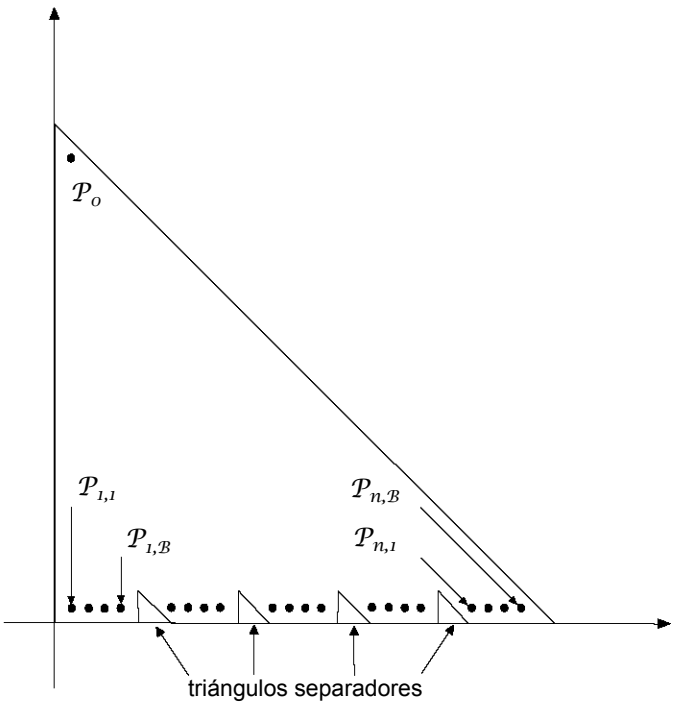


Figura 3.6: Conjunto de puntos P para la reducción NP-Completo

La idea es diseñar un conjunto de puntos P, al interior de un polígono simple S, tal que el vértice v_0 pueda ser mapeado esencialmente a un punto de P. El incrustamiento de los vértices restantes de T sera posible de una manera planar si y solo si las trayectorias π_i pueden ser descompuestas en grupos de exactamente B vértices, lo cual es equivalente a la instancia original de 3-partición. El siguiente conjunto de puntos P y el polígono simple S harán el trabajo (ver Figura 3.6).

- Considerar un polígono simple S como un triángulo construido sobre tres puntos $(0,0),(n(B+2),0)$ y $(0,n(B+2))$, de los cuales $n-1$ triángulos separadores son excluidos. Las coordenadas de los tres vértices de cada triángulo separador son $(B+1+k(B+2),0),(B+1+k(B+2),2)$ y $(B+3+k(B+2),0)$ donde $0 \leq k \leq (n-2)$.

- Considerar $nB+1$ puntos dentro de un polígono simple S como sigue. Un solo punto p_0 con coordenadas $(1,n(B+2)-2)$, en adición a n grupos de B puntos. El punto p_{ij}, el cual es el j-ésimo punto del i-ésimo grupo, tiene coordenadas $((i-1)(B+2)+j,1)$, donde $1 \leq j \leq B$ y $1 \leq i \leq n$.

El polígono simple S es construido de tal forma que los puntos de cada grupo son invisibles desde los puntos de otros grupos. El único punto que es visible desde los otros

puntos es el punto p_0.

Es claro que en cualquier incrustación planar recta de T en el conjunto de puntos P tal que las aristas de T no intersectan el contorino del polígono simple S, el vértice v_0 tiene que ser mapeado en el punto p_0. De otra manera, podria haber un arista $(v_0, v_{i,1})$ que intersecta el contorno del polígono simple S. También, en tal incrustación, los vértices de la trayectoria π_i del árbol T tienen que ser mapeados completamente en los puntos de justamente un grupo, de otra manera la trayectoria intersectaria con el contorno de S. Por lo tanto, un incrustamiento planar geométrico de T es posible si y solo si las trayectorias de T pueden ser organizadas en grupos de tal manera que cada grupo tiene exactamente B vértices.

Recordando que el árbol T tiene $3n$ trayectorias en adición del vértice v_0, por cada trayectoria π_i se tiene $|\pi_i| = a_i$, y el conjunto de puntos P consiste de n grupos de B puntos en adición del punto p_0. Por lo tanto, cualquier mapeo de los vértices de las trayectorias de T, en los puntos de P, provee una gráfica planar geométrica dentro del polígono simple S si y solo si la intancia de 3-partición tiene una solución.

El árbol T tiene un número polinomial de vértices, y un conjunto P con un número polinomial de puntos. Además, las coordenadas de los puntos en P estan acotadas por polinomios y por lo tanto la reducción total puede hacerse en tiempo polinomial. Lo cual concluye la prueba.

□

Capítulo 4

Trayectorias mínimas en el número de aristas en polígonos simples

4.1. Introducción

Definición 4.1. La distancia mínima en el número de aristas entre dos puntos $x, y \in P$ es el mínimo número de aristas (segmentos rectilineos) en una trayectoria que conecta a x y y dentro de P. Denotaremos como $d_L(x, y)$ a la distancia mínima en el número de aristas entre dos puntos $x, y \in P$. En la Figura 4.1 se observa que al menos 4 aristas son necesarias para cualquier trayectoria que conecte a x y y dentro de P. Es decir, $d_L(x, y) = 4$, para el polígono de la Figura 4.1.

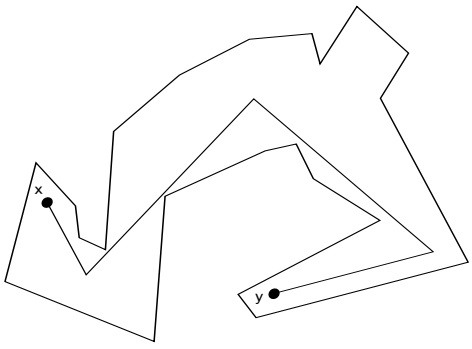


Figura 4.1: Trayectoria mínima en aristas entre x y y

Definición 4.2. Una *trayectoria mínima en aristas* entre dos puntos $x, y \in P$ tiene exactamente $d_L(x, y)$ aristas. Denotaremos a dicha trayectoria como $\pi_L(x, y)$.

Es evidente que una $\pi_L(x, y)$ no es única, de hecho hay un número infinito de trayectorias $\pi_L(x, y)$.

En este capítulo describiremos y analizaremos un algoritmo para el problema de encontrar la trayectoria mínima en aristas qu consiste en lo siguiente: Dados dos puntos $x, y \in P$, calcular una trayectoria mínima en aristas entre los puntos x y y.

El algoritmo que presentaremos tiene complejidad $O(n)$ en tiempo. El algoritmo preprocesa el polígono P para construir una estructura de datos lineal en espacio, que llamaremos el árbol de ventanas de P. Una vez construido este árbol de ventanas, el problema de encontrar una trayectoria mínima en aristas puede ser resuelto en tiempo $O(k + \log n)$, donde $k = d_L(x, y)$.

4.2. Partición por ventanas y el árbol de ventanas

La partición por ventanas de un polígono simple P con respecto a un punto x es particionar a P en regiones tales que todos los puntos de una región tienen la misma distancia en aristas con respecto a x. La idea para el calculo de la partición por ventanas del polígono P es básicamente la siguiente. El polígono P es partido, con respecto a un punto $x \in P$, en regiones sobre las cuales la distancia en aristas desde x es la misma. Denotaremos como $R_i(x)$ a la región cuyo conjunto de puntos tiene distancia en aristas i desde el punto x. Notemos que $V(x)$ es exactamente $R_1(x)$. Eliminando $V(x)$ de P, se tendrá entonces el conjunto de puntos que se encuentran al menos a dos aristas de x. De estos, los únicos puntos que pertenecen a $R_2(x)$, son aquellos que son visibles desde algún punto en $R_1(x)$. De hecho, los puntos en $R_2(x)$ son aquellos puntos de $P - V(x)$ que son visibles desde alguna arista construida de $V(x) = R_1(x)$. En general los puntos en $R_i(x)$ son aquellos puntos de $P - \bigcup_{k=1}^{i-1} R_k(x)$ que son visibles desde alguna arista construida de $R_{i-1}(x)$. En la Figura 4.2 se muestra la partición por ventanas con respecto a x en P.

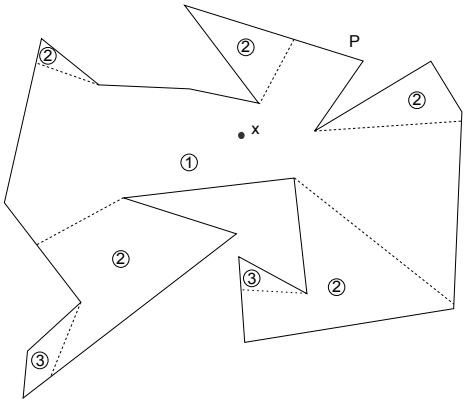


Figura 4.2: Partición en ventanas de P con respecto de x

Definición 4.3. Sea x un punto o un segmento de linea de P y sea $V(x)$ el polígono de visibilidad de x. Una *ventana* de $V(x)$ es una arista del limite o contorno de $V(x)$ que no

pertenece al limite o contorno de P.

Definición 4.4. El *árbol de ventanas* $W(x)$ es el árbol que se forma a partir de la gráfica dual de la partición por ventanas de P con respecto a x, en otras palabras, $W(x)$ tiene un nodo por cada región de la partición y una arista entre cada par de nodos cuyas regiones comparten una ventana.

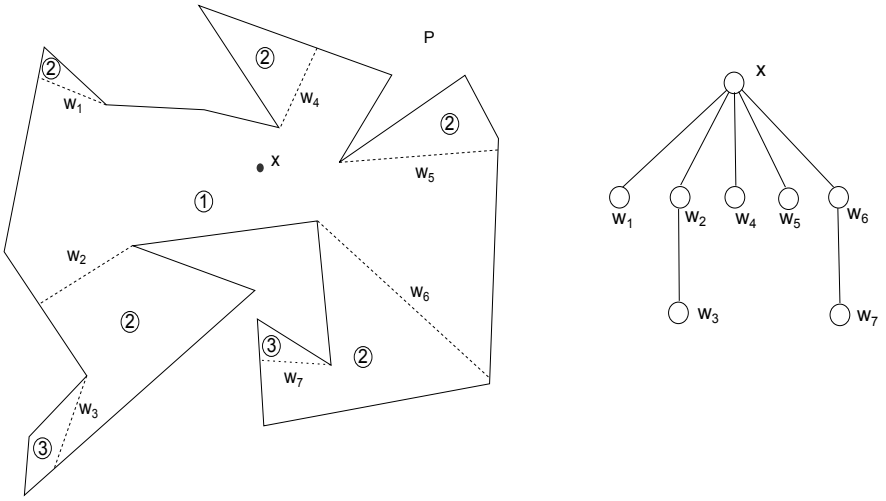


Figura 4.3: Árbol de ventanas del polígono P.

Cada nodo de $W(x)$ es etiquetado con la ventana que genera la región correspondiente a dicho nodo, y la raíz de $W(x)$ es etiquetada por x. En la Figura 4.3 se muestra el árbol de ventanas correspondiente a la partición del polígono P de la Figura 4.2.
Sea c una cuerda y x un segmento de línea o un punto en P, denotaremos como $P[c; x]$ al subpolígono que no contiene a x. Véase la Figura 4.4.

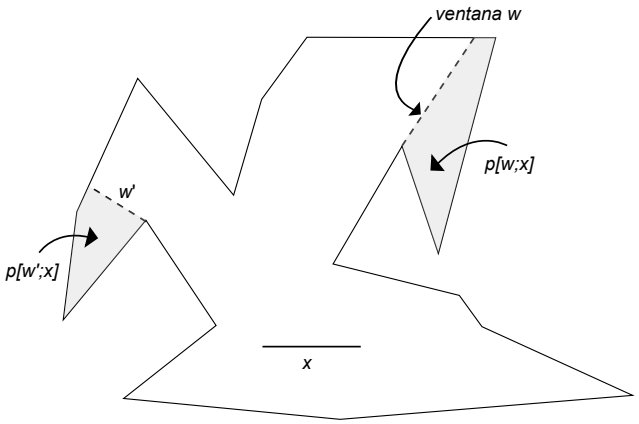


Figura 4.4: Definiciones

Sea $x \in P$ un punto. El algoritmo 4.1 es un procedimiento recursivo que construye la partición por ventanas de P con respecto a x. El algoritmo 4.1 regresa una colección de regiones que en conjunto son la partición de P.

Algoritmo 4.1 Particion-ventana(x,P)

Entrada: Un polígono P y un punto o segmento de linea $x \in P$.
Salida: Partición por ventanas de P con respecto de x.

1: Calcular el polígono de visibilidad de x en P, $V_p(x)$
2: Sean $w_1, w_2, ..., w_k$ las ventanas de $V_p(x)$
3: **para** $i \leftarrow 1$ **hasta** k **hacer**
4: Particion-ventana(w_i, $P\,[w_i; x]$)
5: **fin para**
6: **devolver** $R(x) := V_p(x)$

El algoritmo 4.1 requiere de un algoritmo para calcular la visibilidad desde un punto y la visibilidad débil desde un segmento de línea en P. La visibilidad desde un punto puede calcularse en tiempo $\Theta(n)$ con los algoritmos de ElGindy y Avis [11] o Lee [13]. El polígono de visibilidad débil desde un segmento de línea en P también puede ser calculado en tiempo $\Theta(n)$, siempre que se calcule previamente la triangulación de P. La triangulación de un polígono P puede ser calculada en $\Theta(n)$ también, entonces cada llamada recursiva al procedimiento *Particion-ventana*(,) del algoritmo 4.1 toma tiempo $\Theta(n)$. En la Figura 4.5 se muestra un polígono de n vértices cuya partición en ventanas genera $n/2$ regiones. Entonces, en el peor caso, el algoritmo 4.1 hace $n/2$ llamadas recursivas al procedimiento *Particion-ventana*(,). Por lo tanto en el peor caso, el algoritmo 4.1 toma tiempo $\Omega(n^2)$.

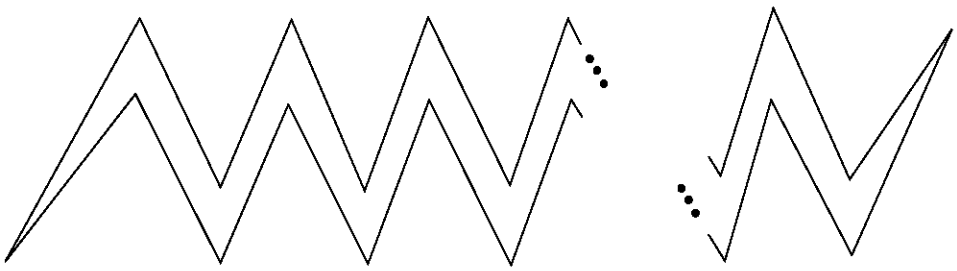


Figura 4.5: Un polígono cuya partición en ventanas tiene $n/2$ regiones.

En lo que sigue se mostrara que esta cota puede ser mejorada mostrando una forma cuidadosa para calcular un polígono de visibilidad débil que dara como resultado una complejidad en tiempo $O(n)$ en el peor caso para el algoritmo 4.1.

La idea principal de esta mejora se basa en hacer ver que cada llamada recursiva a *Particion-ventana*(,) se calcula esencialmente en una región diferente del polígono P. Por lo anterior, dada una triangulación G en P, en la mejora se calculará el polígono de visibilidad débil, de tal forma que solamente aquellos triángulos que pertenecen, total o parcialmente, al polígono de visibilidad son procesados. En particular, si e es una arista (o una cuerda) de P, entonces el algoritmo calculara $V(e)$ en tiempo $O(k_e)$, donde k_e es el número de triángulos de G que son intersectados por $V(e)$. Más adelante demostraremos que cada triangulo en G intersecta a lo más un número constante de regiones de la partición. Combinando estos dos resultados obtendremos que la complejidad en tiempo del algoritmo 4.1 es $O(n)$. A continuación mostraremos el algoritmo para calcular $V(e)$ en tiempo $O(k_e)$, dada una triangulación G de P.

4.3. Algoritmo para calcular visibilidad débil en tiempo $O(k_e)$

El algoritmo a continuación lo describiremos para una arista $e \in P$. Sea G una triangulación dada de P, y sean a, b los puntos finales de la arista e y cd una diagonal de P. El algoritmo recursivo 4.2 calcula el polígono de visibilidad de e. Este algoritmo recibe como entrada una diagonal cd de P y la arista e para la cual se calcula el polígono de visibilidad. El algoritmo 4.2 regresa una secuencia de aristas de $V(e)$, dadas en el sentido de las manecillas del reloj, donde las aristas pertenecen al subpolígono $P\,[cd; e]$.

Sea Δabu el único triángulo de G que tiene ab como una arista. Entonces $V(e)$ es justamente la concatenación de *Visibilidad(au, e)* y *Visibilidad(ub, e)*. Ver la Figura 4.6 para el siguiente procedimiento.

Algoritmo 4.2 Visibilidad(cd, e)

Entrada: Una diagonal $cd \in G$ y una arista $e \in P$.
Salida: Poligono de visibilidad $V(e)$.

1: Sean xy, wz las tangentes internas del par geodésico $\pi_E(a, c)$ y $\pi_E(b, d)$, donde x, z caen sobre $\pi_E(a, c)$ y w, y caen sobre $\pi_E(b, d)$.
2: Sea Δcdg el único triángulo en G que tiene a cd como una arista y que no ha sido procesado aún
3: **si** $\pi_E(a, c)$ y $\pi_E(b, g)$ son ambas externo-convexas **entonces**
4: **si** cg es una arista de P **entonces**
5: Sean l_1 y l_2 los puntos donde cg intersecta las líneas wz y xy, respectivamente;
6: **si** l_2 no se encuentra definida **entonces**
7: *Visibilidad(cg; e)* := $\{zl_1, l_1g\}$;
8: **si no**
9: *Visibilidad(cg; e)* := $\{zl_1, l_1l_2, l_2y\}$;
10: **fin si**
11: **si no** cg es una diagonal de P usada por G
12: Recursivamente llamar Visibilidad(cg; e)
13: **fin si**
14: **fin si**
15: **si** $\pi_E(a, g)$ y $\pi_E(b, d)$ son ambas externo-convexas **entonces**
16: **si** gd es una arista de P **entonces**
17: Sean r_1 y r_2 los puntos donde gd intersecta las líneas wz y xy, respectivamente;
18: **si** r_1 no se encuentra definida **entonces**
19: *Visibilidad(gd; e)* := $\{gr_2, r_2y\}$;
20: **si no**
21: *Visibilidad(gd; e)* := $\{zr_1, r_1r_2, r_2y\}$;
22: **fin si**
23: **si no** gd es una diagonal de P usada por G
24: Recursivamente llamar Visibilidad(gd; e)
25: **fin si**
26: **fin si**
27: *Visibilidad(cd, e)* := *Visibilidad(cg, e)* $\cup$ *Visibilidad(gd, e)*

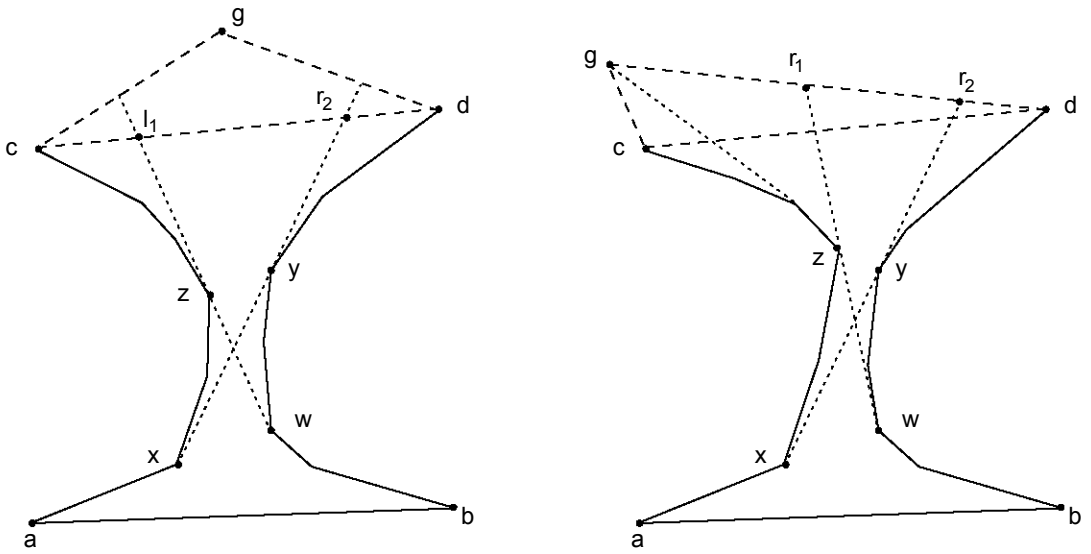


Figura 4.6: Algoritmo *Visibilidad*(cd; e)

El algoritmo 4.2 funciona como sigue. Sea Δcdg el único triángulo que tiene a cd como arista. Lo primero que realiza el algoritmo 4.2 es comprobar si cg es al menos parcialmente visible desde e, es decir, comprueba que $\pi_E(a, c)$ y $\pi_E(b, g)$ sean ambos externo-convexos (condición 1). Si esta condición 1 se cumple, lo siguiente es saber si cg es una arista del polígono, si este es el caso, se verifican los puntos en donde las tangentes xy y wz (definidas en el algoritmo 4.2) intersectan con cg etiquetando a dichos puntos l_1 y l_2 respectivamente. Si l_2 no se encuentra definido, el algoritmo regresa como salida *Visibilidad(cg; e)* := $\{zl_1, l_1g\}$; si l_2 esta definido el algoritmo regresa *Visibilidad(cg; e)* := $\{zl_1, l_1l_2, l_2y\}$. Si pasa que cg no es una arista de P entonces se llama recursivamente al algoritmo 4.2 con cg y e como entrada. Si la condición 1 no se cumple cg no es procesada. De forma análoga se procesa gd, verificando en un principio si gd es al menos parcialmente visible desde e, tal y como se muestra en el algoritmo 4.2, la explicación de esta parte es omitida.

A continuación se muestra un ejemplo de la ejecución del algoritmo 4.2.

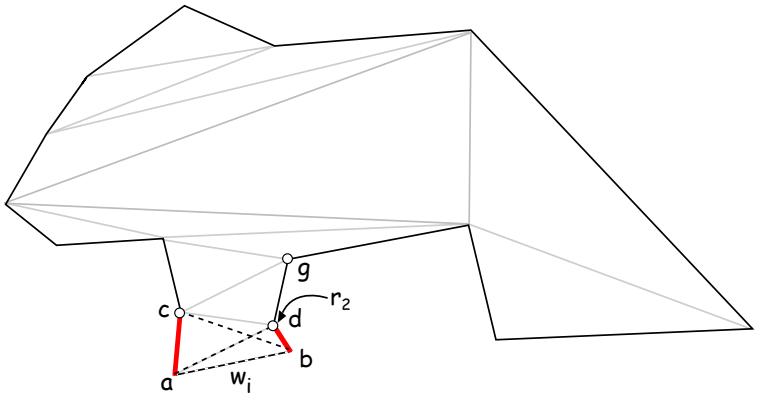


Figura 4.7: Ejecución del algoritmo *Visibilidad*(cd, w_i).

Supongamos que el algoritmo ha sido ejecutado y nos fijamos en el momento en el cual se tiene la entrada *Visibilidad*(cd, w_i), como se muestra en la Figura 4.7 (las aristas mas gruesas forman parte del polígono de visibilidad de w_i en $P[w_i, x]$, donde x es el punto o la arista que genero a w_i) . Sea Δcgd el triángulo a analizar en este momento, la condición del paso 3 del algoritmo *Visibilidad*(cd, w_i) nos dice que si $\pi_E(a, c)$ y $\pi_E(b, g)$ son ambas externo-convexas se tiene que cg es total o parcialmente visible desde w_i, como cg es una diagonal de P, el paso 11 se cumple y el algoritmo procede recursivamente con la entrada *Visibilidad*(cg, w_i). La condición del paso 15 nos dice que si $\pi_E(a, g)$ y $\pi_E(b, d)$ son ambas externo-convexas se tiene que gd es total o parcialmente visible desde w_i, como gd es una arista en P, por el paso 16 se definen r_1 y r_2 como los puntos de intersección de las tangentes con la arista gd. Observemos que r_1 no esta definida (gd solo intersecta con xy), y dado que r_2 y y son el mismo vértice d, por el paso 19 gd es añadida al polígono de visibilidad de w_i en $P[w_i, x]$.

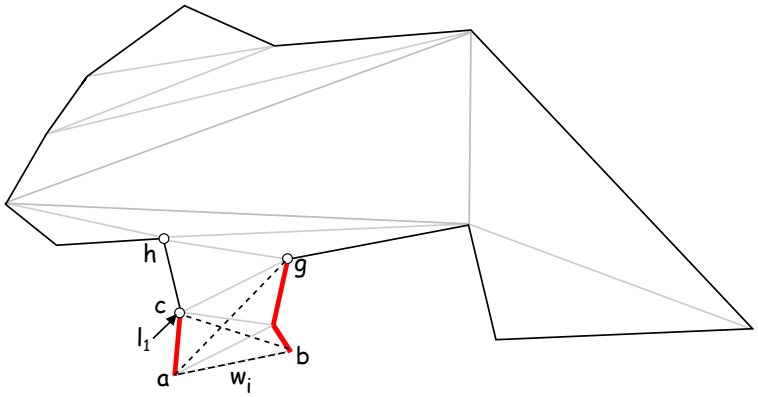


Figura 4.8: Ejecución del algoritmo *Visibilidad*(cg, w_i).

Aún no se completa el calculo del polígono de visibilidad de w_i en $P[w_i, x]$ ya que el algoritmo ahora debe calcular *Visibilidad*(cg, w_i) recursivamente, como se muestra en la

Figura 4.8. Sea Δchg el triángulo a analizar en esta ejecución del algoritmo, la condición del paso 3 del algoritmo se cumple y ch es una arista de P, como l_2 no esta definido y l_1 es justamente el vértice c, entonces ch es añadida al polígono de visibilidad de w_i en $P\,[w_i, x]$. La condición del paso 15 se cumple, y se tiene que hg no es una arista en P, entonces se llama recursivamente a $Visibilidad(hg, w_i)$.

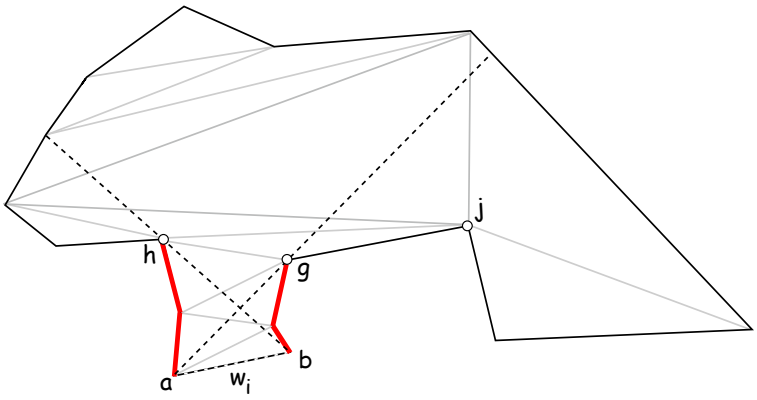


Figura 4.9: Ejecución del algoritmo $Visibilidad(hg, w_i)$.

Sea Δhjg el triángulo a analizar en $Visibilidad(hg, w_i)$ (ver Figura 4.9), la condición del paso 3 del algoritmo se cumple y como hj no es una arista de P, el algoritmo se ejecuta ahora con $Visibilidad(hj, w_i)$. La condición del paso 15 no se cumple, y el algoritmo $Visibilidad(hg, w_i)$ se detiene.

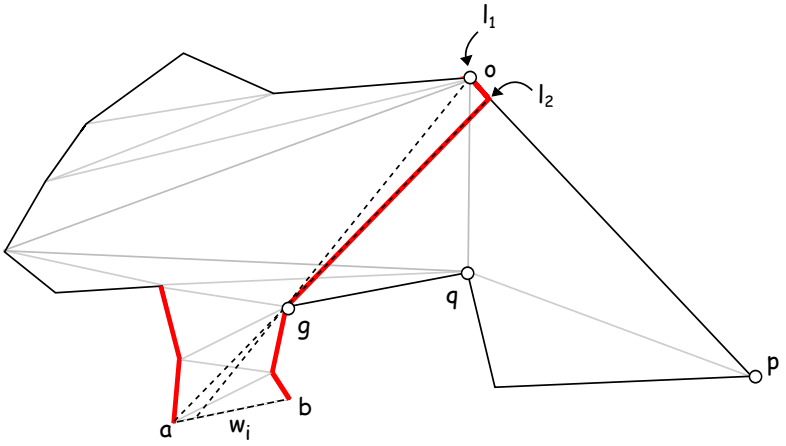


Figura 4.10: Ejecución del algoritmo $Visibilidad(oq, w_i)$.

Sea Δopq el triángulo a analizar en la *i-ésima* iteración, con la entrada $Visibilidad(oq, w_i)$, como se muestra en la Figura 4.10. La condición del paso 3 del algoritmo se cumple y op es una arista de P, como l_1 y l_2 están ambos definidos, entonces las aristas

ol_2 y l_2g son añadidas al polígono de visibilidad de w_i en $P[w_i, x]$.

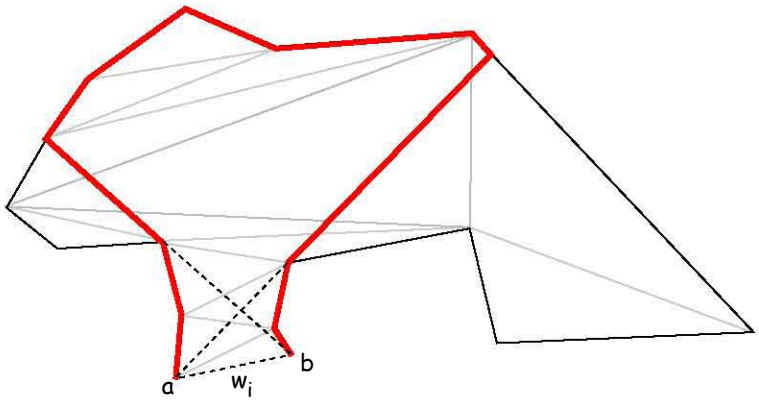


Figura 4.11: Polígono de visibilidad de w_i en $P[w_i, x]$.

Ejecutando así cada una de las llamadas recursivas al algoritmo se obtiene entonces el polígono de visibilidad de w_i en $P[w_i, x]$, el cual es mostrado en la Figura 4.11 representado con lineas más gruesas.

Sea Δabu el único triángulo de G que tiene $e = ab$ como una arista. Sin perdida de generalidad, suponer que ambos au y bu son diagonales de P. Entonces, para calcular $V(e)$, se llama recursivamente a $Visibilidad(au, e)$ y $Visibilidad(ub, e)$, y se concatenan los resultados. Sea k_e el número de triángulos de G que son intersectados por $V(e)$. Para mostrar que cada una de las llamadas recursivas del procedimiento anterior toma tiempo $O(k_e)$, observemos que el algoritmo procesa solamente aquellos triángulos que son por lo menos parcialmente visibles a e, y que cada uno de dichos triángulos es procesado exactamente una vez. Por lo tanto, el costo total es acotado por el costo necesario en construir y mantener árboles de trayectoria mínima de ambos puntos finales de e, a los vértices de k_e triángulos. Por el algoritmo en tiempo lineal de Guibas et al. [15], este costo es acotado por $O(k_e)$. Lo anterior es resumido en el siguiente Lema.

Lema 4.3.1. Sea P un polígono simple de n vértices y sea G una triangulación dada de P. El polígono de visibilidad de una arista e de P puede ser calculada en tiempo $O(k_e)$, donde k_e es el número de triángulos de G intersectados por $V(e)$.

Se mostrará ahora como obtener un resultado similar al Lema 4.3.1 para calcular el polígono de visibilidad desde una cuerda de P. Las cuerdas son elemento muy importante, ya que en el algoritmo $Particion\text{-}ventana(,)$, los polígonos de visibilidad son calculados desde ventanas de otros polígonos de visibilidad en P, y cada una de esas ventanas son una cuerda de P. Sea s una cuerda en P. Supóngase que s divide a P en dos subpolígonos, P_1 y P_2. Sin perdida de generalidad, calcular $V_{P_1}(s)$, el cual es el polígono de visibilidad de s en P_1. Obviamente s es una arista de P_1 y, por lo tanto, el Lema 4.3.1 se mantiene para P_1 y s si P_1 estuviera triangulado. Desafortunadamente, restringiendo

la triangulación dada G (de P) a P_1 no necesariamente da una triangulación valida de P_1, y esto es debido a que s puede cruzar varias diagonales de G. En lo siguiente, se mostrara como modificar localmente a G para obtener una triangulación de P_1 en tiempo proporcional al número de triángulos de G intersectados por $V_{P_1}(s)$. Se procede como sigue.

Sea $P(s) \subseteq P$ el subpolígono que consiste de aquellos triángulos de G cuyo interior intersecta con s y sea $P_1(s) = P_1 \cap P(s)$. Claramente podemos observar que $P_1(s)$ es completamente visible por s. Usando este hecho, se puede triangular a $P_1(s)$ en tiempo lineal, por el método de Avis y Toussaint [14]. Dado que G restringido a $P_1 - P_1(s)$ es una triangulación valida, junto con la triangulación de $P_1(s)$ encontrada anteriormente, se obtiene una triangulación de P_1. El costo incurrido es proporcional al número de triángulos de G intersectados por s, el cual claramente no es mayor que el número de triángulos intersectados por $V_{P_1}(s)$. Con esto se obtiene el siguiente corolario del Lema 4.3.1.

Corolario 4.3.1. Sea P un polígono simple de n vértices y sea G una triangulación dada de P. Sea s una cuerda de P y sea $P_1 \subseteq P$ uno de los dos subpolígonos que resultan de partir P por s. Entonces, el polígono de visibilidad de s en P_1 puede ser calculado en tiempo proporcional al número de triángulos de G que son intersectados por el polígono de visibilidad.

4.4. Complejidad en tiempo para calcular una partición por ventanas

En esta sección se mostrara que el Lema 4.3.1 y el Corolario 4.3.1 son suficientes para acotar la complejidad en tiempo de *Particion-ventana*(x, P) por $O(n)$. Se procede como sigue.

En un principio, se debe de calcular el polígono de visibilidad de x en P, es decir calcular $V(x)$, en tiempo $O(n)$. Para las llamadas restantes al procedimiento *Particion-ventana*(,), los polígonos de visibilidad serán siempre calculados desde cuerdas de P, los cuales serán calculados usando el algoritmo *Visibilidad*(,) diseñado anteriormente. Por el Corolario 4.3.1, la complejidad en tiempo de calcular cada uno de dichos polígonos de visibilidad es proporcional al número de triángulos de G que son intersectados por el polígono de visibilidad en cuestión. Se puede observar entonces que cada polígono de visibilidad que es calculado define una región en la partición por ventanas de P. Por lo tanto, la cota de $O(n)$ para *Particion-ventana*(x,P) es conseguida si se puede mostrar que cada triángulo de G intersecta solamente un número constante de regiones en la partición. Se afirmara que un triángulo de G intersecta a lo más tres regiones de la partición en ventanas de P con respecto de un punto x. Para probar esta afirmación se argumentara lo siguiente (ver Figura 4.12).

Sea *Particion-ventana*$(w_i, P[w_i; w_j])$ la primera vez que un triángulo $t \in G$ es usado, esto es, el polígono de visibilidad de w_i calculado en $P[w_i; w_j]$ intersecta el interior de t. Si t se encuentra totalmente contenido en $V_{P[w_i;w_j]}(w_i)$, entonces t no puede pertenecer a cualquier otra región de la partición, y obviamente la afirmación se mantiene. De otra manera $t \cap V_{P[w_i;w_j]}(w_i)$ es un subconjunto conectado de P y es acotado por a lo más dos ventanas de $V_{P[w_i;w_j]}(w_i)$, por decir, w_{i1} y w_{i2}; ver Figura 4.12. Supóngase que la ventana w'_{i2} de la Figura 4.12 es generada a partir de w_{i2}, entonces la parte del triángulo que esta contenida en $P[w'_{i2}; w_{i2}]$ no es vista por w_{i2}, pero se tiene el hecho de que al dividir una región convexa por medio de una línea recta, el resultado son dos regiones convexas, por lo tanto la parte del triángulo en $P[w_{i2}; w_i]$ es totalmente visible desde w_{i2} y entonces w'_{i2} no es generado por w_{i2}. Supóngase ahora que la ventana w'_{i2} es generada por w_i, se puede ver claramente que esto no puede ocurrir ya que al generar la ventana w_{i2}, la región $P[w_{i2}; w_i]$ no es visible desde w_i, y entonces w'_{i2} no puede ser una ventana de w_i, se concluye entonces que w'_{i2} no puede existir. Para w_i1 el caso es simétrico, por lo tanto no sera demostrado. Sean a y b los puntos extremos de w_i, sean a_1 y b_1, y a_2 y b_2 los puntos extremos de w_{i1} y w_{i2} respectivamente. Por el algoritmo *Visibilidad*(,) se tiene que las trayectorias $\pi_E(a, a_1) \cup a_1b_1$ y $\pi_E(b, a_2) \cup a_2b_2$ son ambas convexas hacia fuera, lo cual indica que la parte de t que se encuentra entre w_{i1} y w_{i2} se encuentra totalmente contenida en $V_{P[w_i;w_j]}(w_i)$. Se puede ver entonces que t se encuentra totalmente 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 $P[w_{i2}; w_i]$, y la afirmación se mantiene.

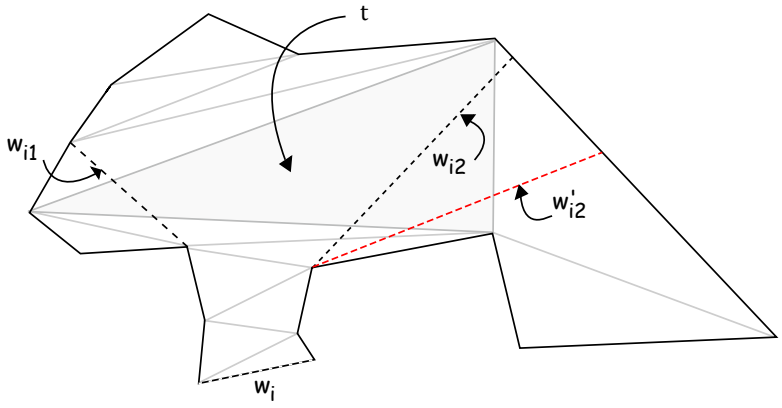


Figura 4.12: Intersección de un triángulo con tres regiones.

En resumen, cada región de la partición en ventanas de P puede ser calculada en tiempo proporcional al número de triángulos de G intersectados por este, y ningún triángulo es intersectado por más de tres regiones. Esto prueba el siguiente teorema.

Teorema 4.4.1. Sea P el polígono triangulado de n vértices. Sea $x$un punto fijo o segmento de línea en P. La partición en ventanas de P con respecto de x puede ser construida en tiempo y espacio óptimos $\Theta(n)$.

4.5. Asociando la partición por ventanas y árboles de ventanas

En esta sección se describirá como asociar una partición por ventanas de P para responder eficientemente consultas, usando algunas técnicas conocidas. Se tiene principal interés en localizar rápidamente un punto de consulta en una subdivisión planar inducida por una partición en ventanas y encontrar de manera rápida, la intersección del contorno de un polígono con un rayo de consulta.

Para el problema de localización de puntos, observe que sobreponiendo la partición en ventanas de P sobre la triangulación dada G, lo cual lleva a una subdivisión de P en $O(n)$ triángulos disjuntos y cuadriláteros convexos. Con esto se puede utilizar el algoritmo en tiempo lineal de Kirkpatrick [16] o el algoritmo de Edelsbrunner, Guibas y Stolfi [12] para preprocesar esta subdivisión y así obtener una estructura de datos, que dado un punto de consulta p, se pueda determinar en tiempo $O(\log n)$ la única región de la subdivisión que contiene a p.

Para responder eficientemente ciertas consultas sobre intersección y visibilidad, se puede observar que cada región $R(w)$ de una partición en ventanas de P es débilmente visible con respecto a su correspondiente ventana w. Usando el método de Avis y Toussaint [14], $R(w)$ puede ser triangulada en tiempo lineal. Una vez triangulado, se puede utilizar la técnica de Guibas et al.[15] para preprocesar a $R(w)$ en tiempo lineal adicional para obtener una estructura de datos que responda a las siguientes consultas en tiempo $O(\log m)$ cada una, donde $m \leq n$ es el tamaño de $R(w)$.

(Q_1) Dado un rayo r que emana de algún punto de w, encontrar el primer punto sobre el contorno de $R(w)$ tocado por r.

(Q_2) Dado un punto p en $R(w)$, calcular el subsegmento de w visible desde p.

(Q_3) Dado un punto p y una dirección u, encontrar el primer punto que toca el contorno de $R(w)$ por el rayo de p con dirección u.

El costo total de preprocesar todas las regiones de una partición en ventanas de P de la forma apenas descrita es solamente $O(n)$. También se mantiene la siguiente información con un árbol de ventanas. Para cada vértice v de $R(w)$, se tiene un punto p_v de w tal que p_v y v son mutuamente visibles. Para hacer esto, se calcula el árbol de trayectorias mínimas de $R(w)$ desde uno de los puntos finales de w, por decir a. Sea u el vértice de $\Phi_E(a, v)$ que precede inmediatamente a v. Sea p_v el punto donde la línea uv intersecta w. Es fácil ver que p_v y v son mutuamente visibles. Dado que todas las regiones de la partición en ventanas pueden ser trianguladas en tiempo lineal, el costo de este procesamiento es $O(n)$.

Por ultimo, con cada nodo del árbol de ventanas, se guarda su profundidad, donde la profundidad de un nodo en el árbol es su distancia (en aristas) entre la raíz y el nodo. En

conclusión, un árbol de ventanas de P puede mejorarse en tiempo $O(n)$ para responder las siguientes consultas.

(E_1) Un punto de consulta $p \in P$ puede ser localizado en una única región $R(w)$ de la partición en ventanas en tiempo $O(\log n)$.

(E_2) Dado un nodo arbitrario $w \in W(x)$, la profundidad de w puede ser encontrada en tiempo $O(1)$.

(E_3) Las consultas de intersección, (Q_1), (Q_2) y (Q_3), pueden ser contestadas en tiempo $O(\log n)$ por consulta.

(E_4) Dado un nodo $w \in W(x)$ y un vértice v en este, un punto $p_v \in w$ que es visible desde v puede ser encontrado en tiempo $O(1)$.

4.6. Árbol de ventana y distancia en aristas

Sea T un árbol que tiene un número finito de nodos. Sea r la raíz de T.

Definición 4.5. La *distancia en un árbol* entre dos nodos $x, y \in T$ es el número de aristas en el árbol de la única trayectoria más corta que conecta a x y y en T.

Definición 4.6. La *altura* del árbol T es la distancia de árbol máxima de la raíz r a cualquiera de las hojas de T. La *profundidad* de un nodo $x \in T$ es la distancia de árbol de x hacia la raíz r.

Definición 4.7. El *subárbol de T con raíz en x*, es obtenido eliminando de T todos los nodos que no son descendientes de x.

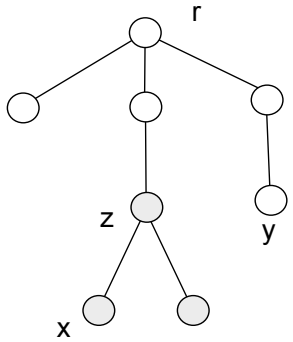


Figura 4.13: Definiciones. La altura del árbol es tres, la distancia entre x y y es cinco y el subárbol con raíz en z es denotado por los nodos sombreados.

Sea x un punto o un segmento de línea en P. El siguiente lema establece que la profundidad de un nodo en $W(x)$ es igual a la distancia en aristas de la ventana correspondiente de x. Si w es un nodo de $W(x)$.

Lema 4.6.1. Sea w un nodo arbitrario de $W(x)$ y sea k la profundidad de w en T. Entonces $d_L(x, w) = k$

Demostración. Sea $(w_0, w_1, ..., w_k)$ la lista ordenada de nodos en la trayectoria de x hacia w en $W(x)$, donde $w_0 = x$ y $w_k = w$, ver Figura 4.14. Para mostrar que $d_L(x, w) \geq k$, considerar una trayectoria mínima en aristas $\pi_L(x, w)$ conectando a x y w en P. Es fácil ver que $\pi_L(x, w)$ atraviesa cada una de las ventanas $w_1, ..., w_{k-1}$. Dado que w_i no puede ver algún punto de $R(w_j)$, para $0 \leq i < j \leq k$, se sigue que $\pi_L(x, w)$ debe de tener por lo menos un vértice en cada $R(w_i)$, para $i = 0, 1, ..., k-2$ lo que implica que $d_L(x, w) \geq k$.

Por otro lado, sea $x_i \in w_i$ el punto de intersección entre la línea determinada por w_{i+1} y w_i, para $0 \leq i \leq k-1$, y sea x_k un punto arbitrario de $w = w_k$. Entonces, $(w_0, w_1, ..., w_k)$ es una trayectoria de k-aristas conectando a x y w, lo cual implica que $d_L(x, w) \leq k$. Lo anterior termina la prueba.

□

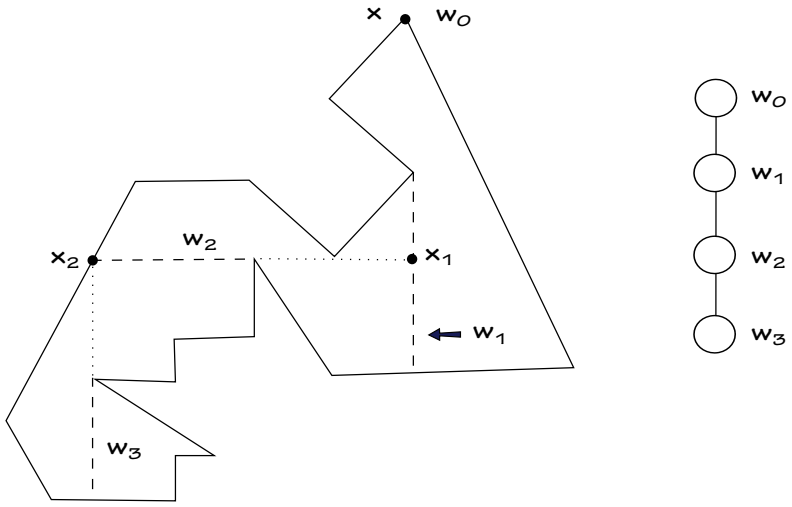


Figura 4.14: Prueba para el Lema 4.6.1.

Corolario 4.6.1. Sea $w \in W(x)$ un nodo arbitrario. Una trayectoria mínima en aristas entre x y w puede ser encontrada en tiempo $O(k)$, donde k es la profundidad de w en $W(x)$.

La siguiente definición asocia, con cada punto de P, una única región en la partición por ventanas de P.

Definición: Sea p un punto arbitrario de P. se definirá como $wmate(p)$ el único nodo del árbol de ventana $W(x)$ tal que p se encuentra en $R(wmate(p))$. Entonces, $wmate(p)$ es llamada la ventana compañero de p.

Dado que p es (débilmente) visible desde una cuerda $wmate(p)$, el siguiente lema se sigue del Lema 4.6.1; por lo tanto la prueba es omitida.

Lema 4.6.2. Sea p un punto arbitrario de P y sea $wmate(p)$ la ventana compañero de p en $W(x)$. Entonces, $d_L(x,p) = k_p + 1$, donde k_p es la profundidad de $wmate(p)$ en $W(x)$.

4.7. Algoritmo para resolver trayectorias mínimas en aristas

El algoritmo que se presenta en esta sección para resolver trayectorias minimas en aristas entre dos puntos, por decir x y y, dentro de un polígono simple corre en tiempo $O(k_y + \log n)$. A continuación se muestra dicho algoritmo.

Algoritmo 4.3 Link-Distance(x,y,P)

Entrada: Un polígono P y un par de puntos $x, y \in P$.
Salida: Trayectoria mínima en aristas entre x y y dentro de P.

1: Calcular la partición en ventanas de P con respecto de x con el algoritmo *Particion-ventana*(x, P).
2: Calcular el árbol de ventanas $W(x)$.
3: Localizar $wmate(y)$
4: Encontrar un punto $z \in wmate(y)$ tal que y sea visible desde z.
5: Calcular una trayectoria minima en aristas entre x y $wmate(y)$, es decir $\pi_L(x, wmate(y))$.
6: **devolver** $\pi_L(x,y) := \pi_L(x, wmate(y)) \cup zy$

El algoritmo 4.3 funciona como sigue. Se calcula la partición en ventanas de P con respecto de x, ya que esta será necesaria para poder construir el árbol de ventanas $W(x)$. Después se localiza el $wmate(y)$, lo cual puede hacerse en tiempo $O(\log n)$ por (E_1) . Utilizando la consulta Q_2 mencionada anteriormente se puede encontrar un punto z dentro de $wmate(y)$ que puede ver a y, en tiempo $O(\log n)$. Por el Corolario 4.6.1 una trayectoria mínima en aristas entre x y $wmate(y)$, $\pi_L(x, wmate(y))$, puede ser calculada en tiempo $O(k_y)$, donde k_y es la profundidad de $wmate(y)$ en $W(x)$, para encontrar dicha trayectoria. Una trayectoria mínima en aristas de y a x es obtenida simplemente añadiendo el segmento zy a la trayectoria $\pi_L(x, wmate(y))$, es decir

$$\pi_L(x,y) = \pi_L(x, wmate(y)) \cup zy$$

La complejidad en tiempo del algoritmo anterior es $O(k_y + \log n)$.

Capítulo 5

Complejidad de incrustar un árbol de Steiner al interior de un polígono simple

5.1. Reducción del problema 3-SAT al problema ISP

Definición del problema ISP

El problema principal de esta tesis es el de incrustar un árbol de Steiner al interior de un polígono simple con puntos en su interior, demostrando que dicho problema pertenece a la clase de problemas no deterministas polínomiales. La complejidad computacional de este problema radica principalmente en la colocación de los vértices Steiner dentro del polígono simple, ya que se tienen infinitas posibilidades para colocarlos. Se hará la restricción de que el conjunto de puntos acotados por el polígono sea un subconjunto de vértices del mismo. A continuación se define formalmente el problema.

Problema de incrustar un árbol de Steiner dentro de un polígono simple (ISP)

Instancia: Un árbol de Steiner $T = (V_T, E_T)$, $|V_T| = p + q$ donde p es el número de vértices terminales y q es el número de vértices Steiner, y un conjunto $D = \{d_1, d_2, ..., d_p\}$ de puntos al interior de un polígono simple P con n vértices.

Pregunta: ¿Existe una biyección entre los vértices terminales de T y los puntos del conjunto D, tal que toda arista de T puede ser construida al interior de P como un segmento rectilineo ?

Teorema 5.1.1. El problema de incrustar un árbol de Steiner al interior de un polígono simple pertenece a la clase de problemas NP-Completos.

Para poder mostrar la NP-Completéz de este problema, se reducirá el problema de 3-Satisfacibilidad booleana, el cual es NP-Completo, al problema de incrustar un árbol

de Steiner dentro de un polígono simple. Se utilizará el termino ISP para referirnos a este último problema a lo largo de este capítulo. Veamos la definición del problema de *3-Satisfacibilidad booleana*

3-Satisfacibilidad booleana (3-SAT)

Instancia: Un conjunto $U = \{u_1, u_2, ..., u_h\}$ de variables booleanas y una colección $C = \{c_1, c_2, ..., c_m\}$ de cláusulas sobre U tal que $c_i \in C$ es una disyunción de precisamente tres literales.

Pregunta: ¿Existe una valuación de las variables en C, tal que, la formula $I = c_1 \wedge c_2 \wedge c_3 \wedge ... \wedge c_m$ es verdadera?

Mostraremos ahora que el problema *3-SAT* se puede transformar en tiempo polinomial al problema *ISP*. Es decir, dada una instancia de *3-SAT* como entrada, se construirá en tiempo polinomial un árbol de Steiner T y una región poligonal P simplemente conectada que encierre a un conjunto de puntos D, tal que T se puede incrustar en D si y solo si la instancia de *3-SAT* es satisfacible. Dividimos la construcción de la instancia de *ISP*, a partir de una instancia de *3-SAT*, en dos partes: la construcción del árbol de Steiner y la construcción del polígono simple.

Construcción del árbol de Steiner

Dada una instancia I de 3-SAT, definimos al conjunto de cláusulas como $C_I = \{c_1, c_2, ..., c_m\}$ y al conjunto de variables como $U_I = \{u_1, u_2, ..., u_h\}$. Sea l una literal. Por cada literal l_{ij}, $i = \{1, 2, 3\}$, que pertenece a una cláusula $c_j \in C_I$, con $j = \{1, 2, ..., m\}$, crear una gráfica $K_{ij} = (V_{K_{ij}}, E_{K_{ij}})$, cuyo conjunto de vértices $V_{K_{ij}}$ sea un vértice Steiner s_{ij} y dos vértices terminales r_{ij} y r'_{ij}; y su conjunto de aristas $E_{K_{ij}}$ esta formado por las aristas $e_{ij} = (s_{ij}, r_{ij})$ y $e'_{ij} = (s_{ij}, r'_{ij})$.

Por cada cláusula c_j, con $j = \{1, 2, ..., m\}$, crear la gráfica $G_j = (V_j, E_j)$, donde V_j es la unión entre el conjunto de vértices $V_{K_{ij}}$, para toda $i = \{1, 2, 3\}$ y un vértice terminal r_{c_j}; y E_j es la unión entre el conjunto de aristas $E_{K_{ij}}$, para toda $i = \{1, 2, 3\}$ y una arista e_{c_j} que conecta al vértice r_{c_j} con algún vértice Steiner $s_{ij} \in \bigcup_i V_{K_{ij}}$.

Sea u_f una variable booleana en U_I. Por cada variable u_f, con $f = \{1, 2, ..., h\}$, crear una gráfica $Q_f = (V_{u_f}, E_{u_f})$ cuyo conjunto de vértices V_{u_f} sea un vértice Steiner s_{u_f} y vértices terminales $r_{u_f z}$, con $z = \{1, 2, ..., q_f\}$,donde q_f es el número de cláusulas en las que aparece la variable u_f y un vértice terminal más $r_{u_f w}$. Crear una arista que vaya del vértice Steiner s_{u_f} a cada uno de los vértices terminales $r_{u_f z}$ y $r_{u_f w}$, para así formar el conjunto E_{u_v}.

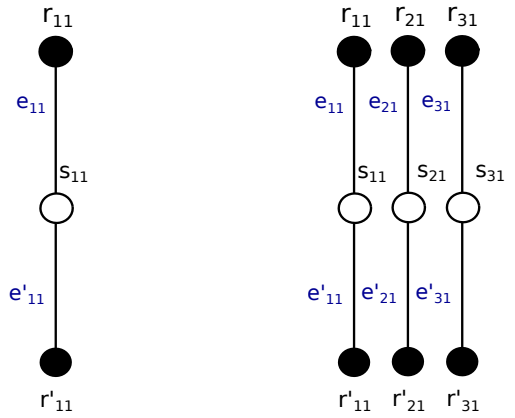
Con lo anterior crear la gráfica $T = (V, E)$, donde V es la unión de los vértices Steiner

y terminales de Q_f y los vértices V_j en G_j, para toda $j = \{1, 2, ..., m\}$ y $f = \{1, 2, ..., h\}$; y E es la unión de las aristas de Q_f y las aristas E_j en G_j, para toda $j = \{1, 2, ..., m\}$ y $f = \{1, 2, ..., h\}$ además de que al conjunto E se le agregan las aristas siguientes: por cada variable u_f, con $f = \{1, 2, ..., h\}$, se creará una arista que vaya del vértice Steiner s_{u_f} hacia cualquier vértice Steiner s_{ij}, con $i = \{1, 2, 3\}$, si y solo si la variable u_f se encuentra presente en la cláusula c_j. Con las restricciones adicionales de que cada vértice s_{u_f} es adyacente a lo más a un vértice s_{ij} de la claúsula c_j y además un vértice s_{ij} no puede ser adyacente a más de un vértice s_{u_f}.

Creamos una gráfica más $A = (V_a, E_a)$, tal que V_a este compuesta por un vértice Steiner s_a y h vértices terminales r_a, y E_a contiene a las aristas que conectan al vértice s_a con cada uno de los h terminales.

Para finalizar, unimos la gráfica A con la gráfica T haciendo $T = (V_a \cup V, E_a \cup E)$ y creamos las aristas $e_{u_f} = (s_a, s_{u_f})$ para toda $f = \{1, 2, ..., h\}$. Nótese que la gráfica T es un árbol y el número total de vértices Steiner que contiene es $3m + h + 1$.

Para ejemplificar esta construcción, se utilizará la siguiente instancia I de *3-SAT*, $I = (\overline{u_1} \vee \overline{u_2} \vee u_3) \wedge (\overline{u_1} \vee u_2 \vee u_3) \wedge (u_1 \vee \overline{u_2} \vee \overline{u_3})$. Comenzaremos por definir a los conjuntos U_I y C_I como sigue: las variables que participan en la intancia I son u_1, u_2 y u_3 por lo tanto $U_I = \{u_1, u_2, u_3\}$, y dado que el número de cláusulas que tiene la instancia I es tres, entonces el conjunto C_I queda como $C_I = \{c_1, c_2, c_3\}$.



(a) Gráfica K_{11}

(b) Gráficas K_{11}, K_{21} y K_{31}

Figura 5.1: Gráficas K para las literales de la cláusula c_1

Para la construcción de la gráfica K_{ij} se debe tener en cuenta que por definición cada cláusula esta formada por tres literales, por lo tanto se tendrán para este ejemplo 9 gráficas K_{ij}. La gráfica K_{11} quedara definida como $K_{11} = (V_{K_{11}}, E_{K_{11}})$, $V_{K_{11}} = \{s_{11}, r_{11}, r'_{11}\}$ donde s_{11} es el vértice Steiner y r_{11} y r'_{11} son los vértices terminales; y $E_{K_{11}} = \{e_{11}, e'_{11}\}$

donde $e_{11} = (r_{11}, s_{11})$ y $e'_{11} = (s_{11}, r'_{11})$. En la Figura 5.1(a) se puede observar esta construcción. Para las 8 gráficas K_{ij} restantes el procedimiento es similar, lo único que cambia es la etiqueta de los vértices. En la Figura 5.1(b) se pueden observar las tres gráficas K_{ij} para las literales de la cláusula c_1.

Para la cláusula c_1 se creará la gráfica $G_1 = (V_1, E_1)$, tal que el conjunto $V_1 = V_{K_{11}} \cup V_{K_{21}} \cup V_{K_{31}} \cup \{r_{c1}\} = \{r_{11}, s_{11}, r'_{11}, r_{21}, s_{21}, r'_{21}, r_{31}, s_{31}, r'_{31}, r_{c1}\}$, donde r_{c1} es un nuevo vértice terminal; y $E_1 = \{e_{11}, e'_{11}, e_{21}, e'_{21}, e_{31}, e'_{31}, e_{c1}\}$ donde e_{c1} es una arista que conecta al vértice r_{c1} con cualquiera de los vértices Steiner s_{11}, s_{21} o s_{31}, para este ejemplo se tomará a s_{11} de tal forma que $e_{c1} = (r_{c1}, s_{11})$. Para las demás cláusulas el procedimiento es el mismo. En la Figura 5.2 se puede observar la construcción de las gráficas G para las cláusulas c_1, c_2 y c_3.

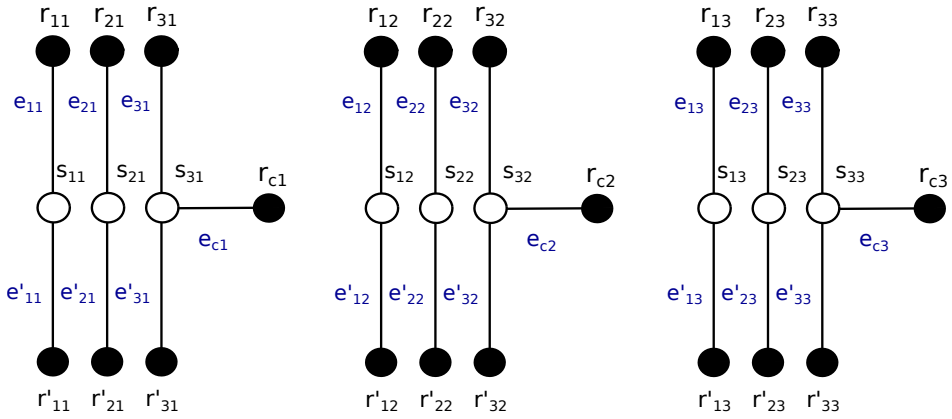


Figura 5.2: Gráficas G_j de las cláusulas c_1, c_2 y c_3

Para la variable u_1 se creará la gráfica $Q_1 = (V_{u_1}, E_{u_1})$, tal que el conjunto $V_{u_1} = \{s_{u_1}, r_{u_1}1, r_{u_1}2, r_{u_1}3, r_{u_1}w\}$, donde s_{u_1} es el vértice Steiner y los vértices r_{u_1} son vértices terminales. En este ejemplo q_i es 3 para toda $\{i = 1, ..., f\}$. El conjunto $E_{u_1} = \{e_{u_1}1, e_{u_1}2, e_{u_1}3, e_{u_1}w\}$ donde $e_{u_1}1 = (s_{u_1}, r_{u_1}1)$, $e_{u_1}2 = (s_{u_1}, r_{u_1}2)$, $e_{u_1}3 = (s_{u_1}, r_{u_1}3)$, y $e_{u_1}w = (s_{u_1}, r_{u_1}w)$. Para las demás variables el procedimiento es el mismo. En la Figura 5.3 se muestra la construcción de las gráficas Q_f para las variables u_1, u_2 y u_3.

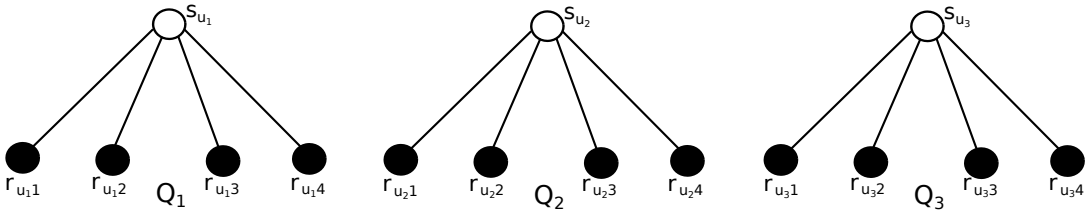


Figura 5.3: Gráficas Q_f de las variables u_1, u_2 y u_3

Para la construcción de la gráfica $T = (V, E)$, se uniran las gráficas G_j, con $j = \{1, 2, 3\}$, y Q_f, con $f = \{1, 2, 3\}$, creadas anteriormente, por lo que la gráfica T tendrá como conjunto de vértices a $V = \{V_1 \cup V_2 \cup V_3 \cup V_{u_1} \cup V_{u_2} \cup V_{u_3}\} = \{r_{11}, ..., r_{33}, r'_{11}, ..., r'_{33}, r_{c1}, r_{c2}, r_{c3}, s_{11}, ..., s_{33}, r_{u_1}1, ..., r_{u_3}4, s_{u_1}, s_{u_2}, s_{u_3}\}$ y como conjunto de aristas a $E = \{E_1 \cup E_2 \cup E_3 \cup E_{u_1} \cup E_{u_2} \cup E_{u_3}\} = \{e_{11}, ..., e_{33}, e'_{11}, ..., e'_{33}, e_{c_1}, e_{c_2}, e_{c_3}, e_{u_1}1, ..., e_{u_3}3\}$. Dado que cada variable se encuentra presente en todas las claúsulas de este ejemplo, entonces se creará una arista que vaya de s_{u_1} hacia alguno de los vértices s_{i1}, una de s_{u_1} hacia alguno de s_{i2} y una más de s_{u_1} hacia alguno de s_{i3}, para $i = \{1, 2, 3\}$, la construcción para las aristas de s_{u_2} y s_{u_3} es similar, de tal forma que la arista no sea adyacente a un mismo vértice s_{i1}, s_{i2}, s_{i3}, para este ejemplo las aristas creadas son las siguientes: (s_{u_1}, s_{11}), (s_{u_1}, s_{12}), (s_{u_1}, s_{33}), (s_{u_2}, s_{21}), (s_{u_2}, s_{22}), (s_{u_2}, s_{13}), (s_{u_3}, s_{31}), (s_{u_3}, s_{32}), (s_{u_3}, s_{23}). En la Figura 5.4 se puede observar esta construcción.

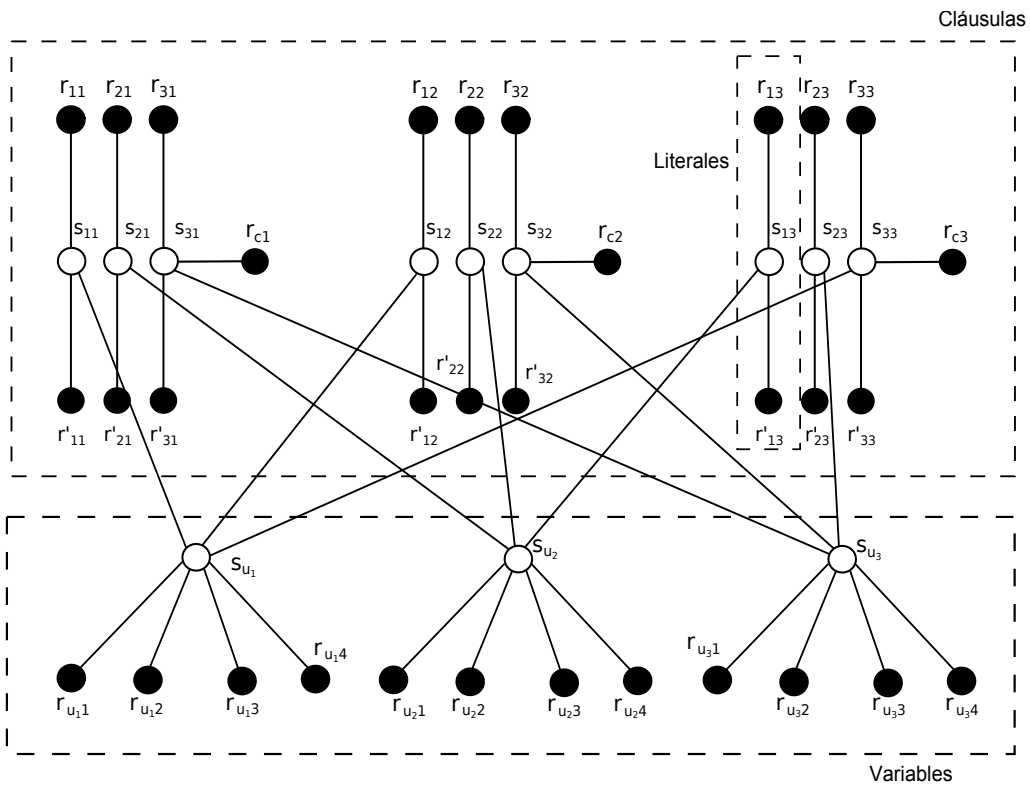


Figura 5.4: Gráfica T.

Se construirá ahora la gráfica $A = (V_a, E_a)$, cuyo conjunto de vértices será $V_a = \{s_a, r_{a1}, r_{a2}, r_{a3}\}$, donde s_a es el vértice Steiner, y su conjunto de aristas $E_a = \{(s_a, r_{a1}), ..., (s_a, r_{a3})\}$.

Se creará una nueva gráfica T_s uniendo la gráfica A a la gráfica T, y se añadiran las

aristas $(s_a, s_{u_1}), (s_a, s_{u_2})$ y (s_a, s_{u_3}). La gráfica T_s es el árbol de Steiner para la instancia I. En la Figura 5.5 se puede observar el árbol T_s.

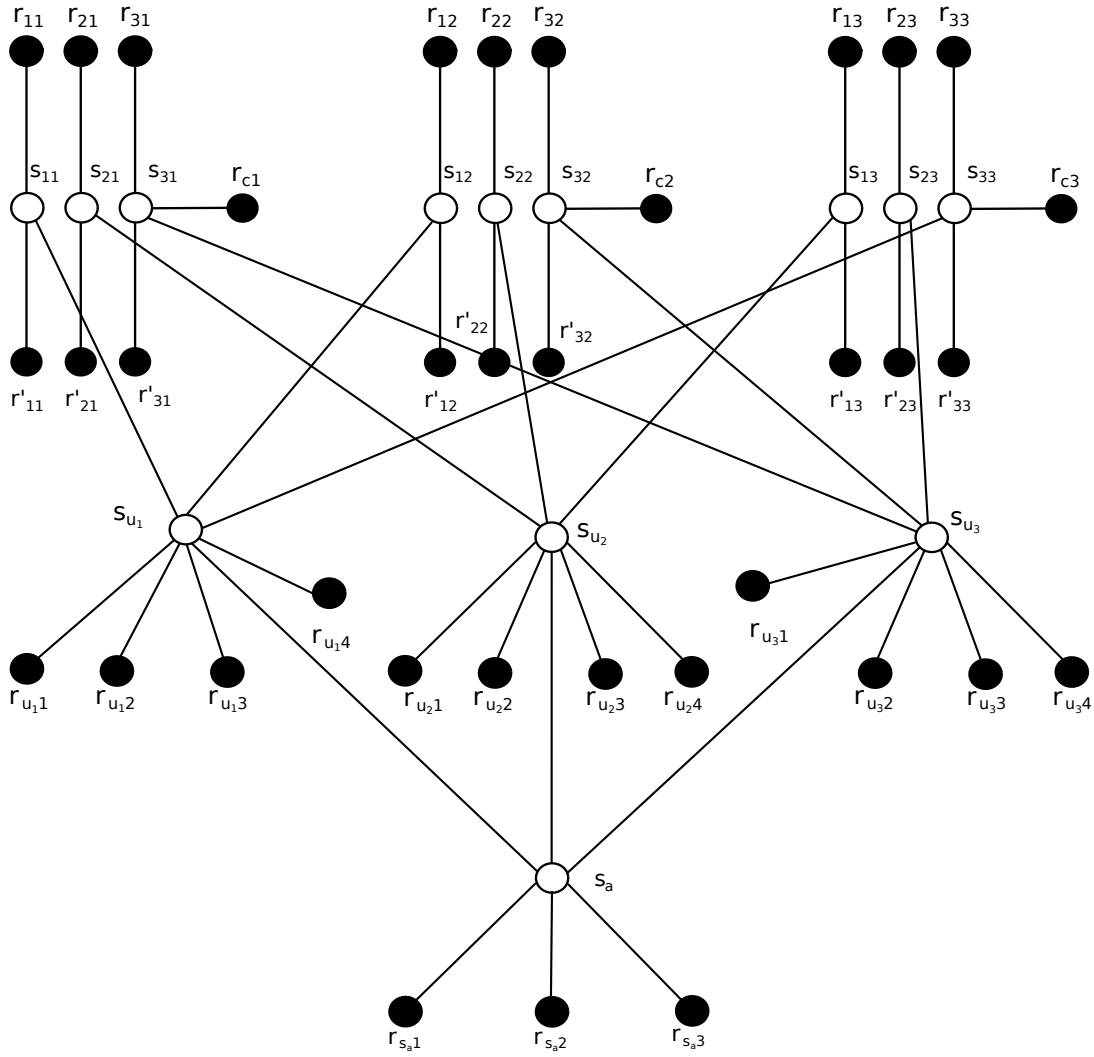


Figura 5.5: Construcción completa del árbol de Steiner T para la instancia I: $(\overline{u_1} \vee \overline{u_2} \vee u_3) \wedge (\overline{u_1} \vee u_2 \vee u_3) \wedge (u_1 \vee \overline{u_2} \vee \overline{u_3})$.

Construcción del polígono

La construcción del polígono esta basada en la construcción realizada por D.T. Lee en [6] para la demostración de la NP-Completéz del problema de la galeria de arte en polígonos simples.

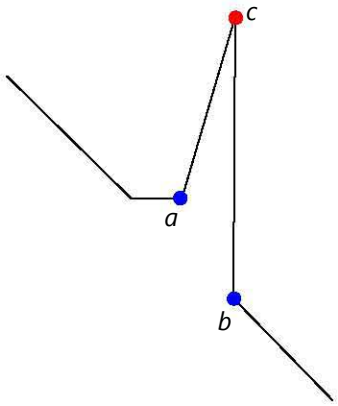


Figura 5.6: Patrón de la literal

Patrón de las literales: El patrón para las literales es mostrado en la Figura 5.6. La principal característica que tiene este patrón es que únicamente los vértices a y b pueden ver al vértice c. Tres de estos patrones existirán por cláusula en la construcción final del polígono.

Patrón de las cláusulas: Sin perdida de generalidad, considérese la cláusula $C_h = A \vee B \vee D$, donde $A \in \{u_i, \overline{u_i}\}$, $B \in \{u_j, \overline{u_j}\}$ y $D \in \{u_k, \overline{u_k}\}$ son literales y u_i, u_j y u_k son variables en U. El patrón básico para la cláusula C_h es mostrado en la Figura 5.7.

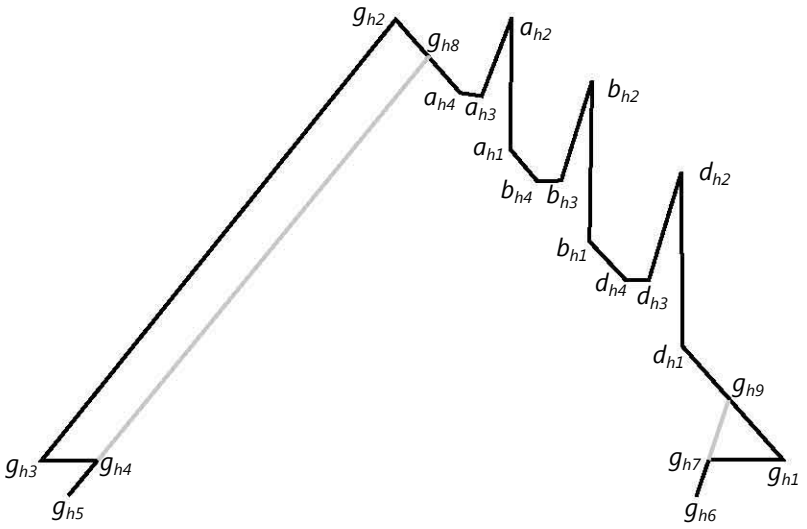


Figura 5.7: Patrón de las cláusulas C_h

Definición 5.1. Dos o más puntos son *colineales* si dichos puntos pertenecen a una misma línea recta.

A los vértices a_{h2}, b_{h2}, d_{h2} y g_{h1} les llamaremos vértices clave.Sean $a_1, a_2, ..., a_h$

puntos en el plano, escribiremos $col(a_1, a_2, ..., a_h)$ cuando los puntos son colineales. En el patrón de la Figura 5.7 se tiene entonces $col(g_{h2}, g_{h8}, a_{h4}, a_{h1}, b_{h4}, b_{h1}, d_{h4}, d_{h1}, g_{h9}, g_{h1})$, $col(g_{h3}, g_{h4}, g_{h7}, g_{h1})$, $col(g_{h8}, g_{h4}, g_{h5})$ y $col(g_{h9}, g_{h7}, g_{h6})$.

Observación 5.1.1. Ninguno de los g_{hi}, con $i = \{1, 2, 3, ..., 7\}$, puede ver a los vértices a_{h2}, b_{h2} y d_{h2} en el patrón para C_h. Además cualesquiera dos vértices de cada patrón de literales en la cláusula, no son suficientes para ver a los vértices clave del patrón C_h. Por lo tanto al menos tres vértices de C_h son requeridos para ver a los vértices clave en el patrón C_h.

Lema 5.1.1. Solamente existen siete configuraciones de tres vértices, tal que estos pueden ver a los vértices clave en el patrón C_h.

Demostración. Ninguno de los vértices a_{h4}, b_{h4}, d_{h4}, y g_{hi}, con $i = \{1, 2, ..., 7\}$, puede ver a cualquiera de los tres vertices clave de las literales, por lo que unicamente consideramemos configuraciones de $H_i = \{a_{hi}, b_{hi}, d_{hi}\}$, $i = \{1, 2, 3\}$. Como H_2 forma parte de los vértices clave, este conjunto no puede ser tomado. Vértices a_{h1} y a_{h3} no podrían ser seleccionados al mismo tiempo. Si dichos vértices fueran seleccionados, se podrían necesitar dos vértices más para ver a los vértices b_{h2} y d_{h2}. Esto incluso se cumple para b_{h1} y b_{h3}, y d_{h1} y d_{h3}. Esto significa que ocho posibles configuraciones de tres vértices existen de los conjuntos H_1 y H_3 de vértices para ver a todos los vértices clave del patrón de cláusula C_h. Si se tomaran los tres vértices en H_3, estos no podrian ver a g_{h1}, por lo tanto quedan siete posibles configuraciones con tres vértices. Es fácil comprobar que cada una de estas siete configuraciones con tres vértices $\{a_{h1}, b_{h1}, d_{h1}\}$, $\{a_{h1}, b_{h1}, d_{h3}\}$, $\{a_{h1}, b_{h3}, d_{h1}\}$, $\{a_{h1}, b_{h3}, d_{h3}\}$, $\{a_{h3}, b_{h1}, d_{h1}\}$, $\{a_{h3}, b_{h1}, d_{h3}\}$ y $\{a_{h3}, b_{h3}, d_{h1}\}$, pueden ver a los vértices clave del patrón C_h.

□

Etiquetando a los vértices: Etiquetaremos a los vértices de $H_1 \cup H_3$ con la función de etiquetamiento $l : H_1 \cup H_3 \to \{t, f\}$ (t = verdadero y f = falso) definida como sigue: sea $W \in \{A, B, D\}$ una literal, si $W = u$, $l(w_{h1}) = t$, $l(w_{h3}) = f$, y si $W = \overline{u}$ entonces $l(w_{h1}) = f$ y $l(w_{h3}) = t$. Cabe señalar que esta función de etiquetamiento representa el valor de verdad de las variables.

Diremos que el asignamiento de las variables u_i, u_j y u_k en C_h, es un *asignamiento verdadero* si $A \vee B \vee D =$*verdadero*, y un *asignamiento falso* de cualquier otra manera.

Lema 5.1.2. Sean α, β y γ tres vértices del patrón de cláusula C_h. Los vértices α, β y γ pueden ver a los vértices clave de C_h si y solo si $C_h = A \vee B \vee D =$ *verdadero*.

Demostración. C_h tiene un valor de verdad *falso* si y solo si los valores de verdad de las literales A, B y D son *falso*. Del Lema 5.1.1 tenemos que si $\alpha = a_{h3}$, $\beta = b_{h3}$ y $\gamma = d_{h3}$,

estos vértices no podrian ver a todos los vértices clave del patrón de cláusula C_h, por lo que al menos uno de $\{a_{h1}, b_{h1}, d_{h1}\}$ es necesario, observemos lo siguiente: si $A = u$, donde u es una variable de la fórmula, entonces $l(a_{h1}) = t$, haciendo $A = t$, y si $A = \overline{u}$ se tiene que $l(a_{h1}) = f$, haciendo $A = t$, lo cual también ocurre para las otras literales. Sabemos que para que la cláusula $C_h = A \vee B \vee D$ sea verdadera, al menos una literal tiene que ser verdadera, por lo observado anteriormente esto es logrado si uno de α, β y γ es asignado a alguno de $\{a_{h1}, b_{h1}, d_{h1}\}$. Por lo tanto tomando cualquiera de las siete asignaciones mostradas en el Lema 5.1.1 la cláusula $C_h = A \vee B \vee D$ toma valor de verdad *verdadero*.

Supongamos ahora que se toma una configuración la cual no puede conectar a todos los vértices clave del patrón de cláusula para C_h, por ejemplo $\alpha = a_{h3}$, $\beta = b_{h3}$ y $\gamma = d_{h3}$, por la observación anterior, ninguna de las literales A, B o D toma valor de verdad verdadero, haciendo que la cláusula $C_h = A \vee B \vee D$ tome valor de verdad *falso*.

□

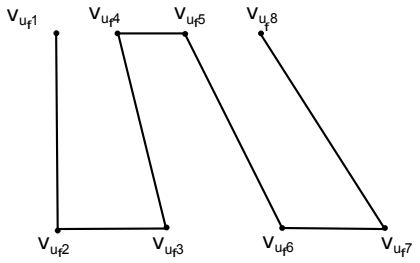


Figura 5.8: Patrón de las variables

Patrón de las variables: Sea u_f una variable en U. El patrón de variable para la variable u_f se muestra en la Figura 5.8, el propósito de este patrón es el de hacer consistentes los valores de verdad asignados a las literales de una variable, es decir que si la literal $W = u_f$ tiene valor de verdad *verdadero*, entonces $W = \overline{u_f}$ tendrá como valor de verdad *falso* y viceversa, en todas las cláusulas.

El patrón de variable lo describiremos a continuación para u_f, entendiendose que el procedimiento es igual para las demás variables. El patrón de variable para u_f consiste de dos cavidades (regiones rectangulares) las cuales tendran vértices etiquetados en su parte superior e inferior por medio de la función de etiquetamiento ζ. Sea $V_{u_f} = \{v_{u_f1}, ..., v_{u_f8}\}$ el conjunto de vértices del patrón de variable de u_f, definimos a la función de etiquetamiento $\zeta : V_{u_f} \to \{F_f, V_f.w_f, w'_f\}$, como sigue: $\zeta(v_{u_f3}) = w_f$, $\zeta(v_{u_f4}) = F_f$, $\zeta(v_{u_f7}) = w'_f$, $\zeta(v_{u_f8}) = T_f$. Las etiquetas $\zeta(v_{u_f4})$ y $\zeta(v_{u_f8})$ representan los valores de verdad *falso* y *verdadero* para la variable u_f, respectivamente. Los vértices v_{u_f3} y v_{u_f7} son vértices clave para este patrón.

Construcción completa: Una vez definidos los patrones de variables y de cláusulas, es

necesario relacionar dichos patrones. Sea u_f una variable con vértices etiquetados como T_f y F_f en su patrón de variable, y sea $W \in \{u, \overline{u}\}$ una literal con vértices etiquetados como t y f en su patrón de literal. Para lograr la relación entre el patrón de variables y el patrón de cláusulas, es necesario agregar a cada cavidad del patrón de variables q_f puntas, donde q_f es el número de cláusulas en las que la variable u_f participa, de tal forma que dichas puntas esten alineadas a los vértices F_f y T_f del patrón de variable y a los vértices f y t de cada patrón de literal de las cláusulas a las que pertenece u_f.

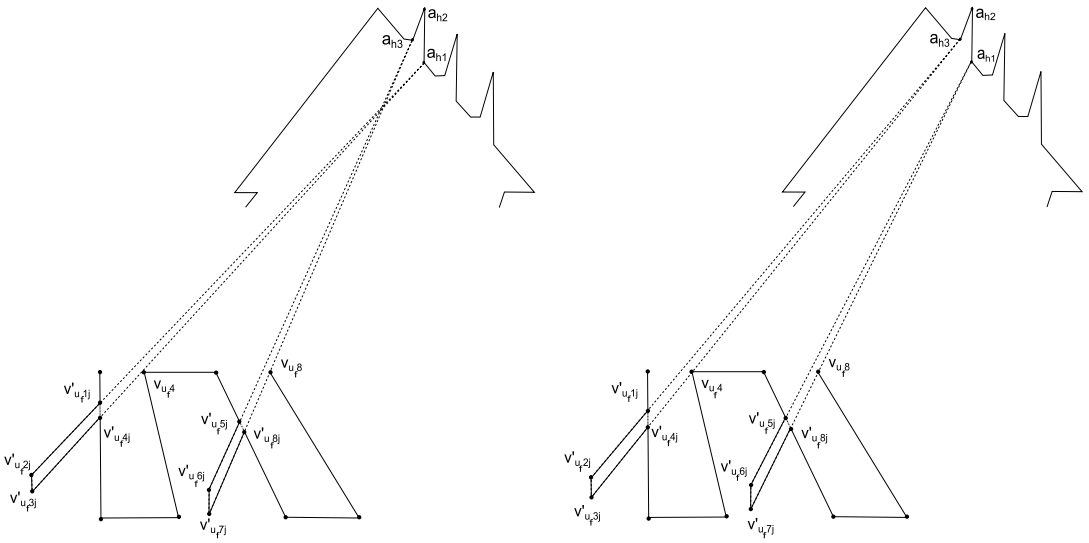
La cavidades del patrón para las variables son construidas a partir del vértice w mostrado en la Figura 5.10, de tal forma que se cumpla que $col(w, v_{u_f1}, v_{u_f2})$, $col(w, v_{u_f4}, v_{u_f3})$, $col(w, v_{u_f5}, v_{u_f6})$, $col(w, v_{u_f8}, v_{u_f7})$, y que $col(v_{u_f1}, v_{u_f4}, v_{u_f5}, v_{u_f8})$ y $col(v_{u_f2}, v_{u_f3}, v_{u_f6}, v_{u_f7})$ son paralelos. Podemos observar que el vértice w puede ver a los vertices v_{u_f3} y v_{u_f7}.

Mostraremos ahora como construir las puntas del patrón de variables. Supongamos que u_f aparece en la cláusula C_j y $A \in \{u_f, \overline{u_f}\}$ es una literal en C_j. Una punta en la cavidad izquierda del patrón de variable para u_f es contruida agregando el conjunto de vértices $\left\{v'_{u_f1j}, ..., v'_{u_f4j}\right\}$ y el conjunto de aristas $\left\{(v'_{u_f1j}, v'_{u_f2j}), (v'_{u_f2j}, v'_{u_f3j}), (v'_{u_f3j}, v'_{u_f4j})\right\}$ de tal forma que $\text{col}(a_{h_1}, v_{u_f4}, v'_{u_f4j}, v'_{u_f3j})$, $\text{col}(a_{h_1}, v'_{u_f1j}, v'_{u_f2j})$ y $col(v'_{u_f2j}, v'_{u_f3j})$, si $l = u_f$, y si $l = \overline{u_f}$ el conjunto de aristas es $\left\{(v'_{u_f1j}, v'_{u_f2j}), (v'_{u_f2j}, v'_{u_f3j}), (v'_{u_f3j}, v'_{u_f4j})\right\}$ tal que $\text{col}(a_{h_3}, v_{u_f4}, v'_{u_f4j}, v'_{u_f3j})$, $\text{col}(a_{h_3}, v'_{u_f1j}, v'_{u_f2j})$ y $col(v'_{u_f2j}, v'_{u_f3j})$. El segmento (v'_{u_f2j}, v'_{u_f3j}) es eliminado del polígono en ambos casos, y el vértice v'_{u_f3j} es un vértice clave.

Una punta en la cavidad derecha del patrón de variable para u_f es contruida agregando el conjunto de vértices $\left\{v'_{u_f5j}, ..., v'_{u_f8j}\right\}$ y el conjunto de aristas $\left\{, (v'_{u_f5j}, v'_{u_f6j}), (v'_{u_f6j}, v'_{u_f7j}), (v'_{u_f7j}, v'_{u_f8j})\right\}$ de tal forma que $\text{col}(a_{h_1}, v_{u_f8}, v'_{u_f8j}, v'_{u_f7j})$, $\text{col}(a_{h_1}, v'_{u_f5j}, v'_{u_f6j})$ y $col(v'_{u_f6j}, v'_{u_f7j})$, si $l = u_f$, y si $l = \overline{u_f}$ el conjunto de aristas es $\left\{, (v'_{u_f5j}, v'_{u_f6j}), (v'_{u_f6j}, v'_{u_f7j}), (v'_{u_f7j}, v'_{u_f8j})\right\}$ tal que $\text{col}(a_{h_3}, v_{u_f8}, v'_{u_f8j}, v'_{u_f7j})$, $\text{col}(a_{h_3}, v'_{u_f5j}, v'_{u_f6j})$ y $col(v'_{u_f6j}, v'_{u_f7j})$. El segmento (v'_{u_f6j}, v'_{u_f7j}) es eliminado del polígono en ambos casos y el vértice v'_{u_f7j} es un vértice clave.

La consecuencia de estos alineamientos es que el vértice etiquetado como F_f puede ver a todos los vértices clave que se encuentran dentro de las puntas de la cavidad izquierda y el vértice etiquetado como T_f puede ver a todos los vértices clave que se encuentran dentro de las puntas de la cavidad derecha.

Un ejemplo de tres variables y tres cláusulas es mostrado en la Figura 5.10. Cada patrón de variable tiene tres puntas, una por cavidad, por cada literal en la cláusula que usa dicha variable.



(a) Caso cuando $A = u$.

(b) Caso cuando $A = \overline{u}$.

Figura 5.9: Construcción de las puntas del patrón de variable.

Observación 5.1.2. Dado que todas las puntas en la cavidad izquierda están alineadas con $v_{u_f 4}$ y a_{h_1} para todas las literales usando u_f y alineadas con $v_{u_f 4}$ y a_{h_3} para todas las literales usando $\overline{u_f}$, todos los vértices clave dentro de las puntas de la cavidad izquierda podrán ser vistos desde los vértices a_{h_1} y a_{h_3} con los que $v_{u_f 4}$ fue alineado. Esto significa que si las literales que involucran a u_f son asignadas con valores de verdad consistentes con $u_f = \textit{verdadero}$, entonces el vértice $v_{u_f 4}$ del patrón de variable de u_f no será necesario para ver a los vértices clave que se encuentran dentro de las puntas de la cavidad izquierda, pero si será necesario el vértice $v_{u_f 8}$. Una conclusión opuesta es alcanzada con $u_f = \textit{falso}$.

Lema 5.1.3. El mínimo número de vértices que pueden ver a los vértices clave en el polígono son $3m + h + 1$.

Demostración. Por el Lema 5.1.1 se tiene que al menos 3 vértices son necesarios para ver a los vértices clave de cada patrón de cláusula, y por definición se tiene que en total m patrones de cláusulas participan en la construcción del polígono, de aqui se tienen $3m$ vértices necesarios. Por lo mencionado anteriormente, si el asignamiento de los valores de verdad es consistente para las variables, únicamente un vértice etiquetado como F_f o T_f, con $f = \{1, ..., h\}$, será necesario para poder ver a los vértices clave del patrón de variable, es decir h vértices son necesarios. El último vértice necesario es el vértice w. □

Definición 5.2. Sean $v_{c1}, v_{c2}, ..., v_{cp}$ los vértices clave definidos en P, denotaremos como *conjunto de puntos en el plano* $D = \{d_1, d_2, ..., d_p \mid\ d_i = v_{ci}, i = 1, 2, ..., p\}$ al conjunto de

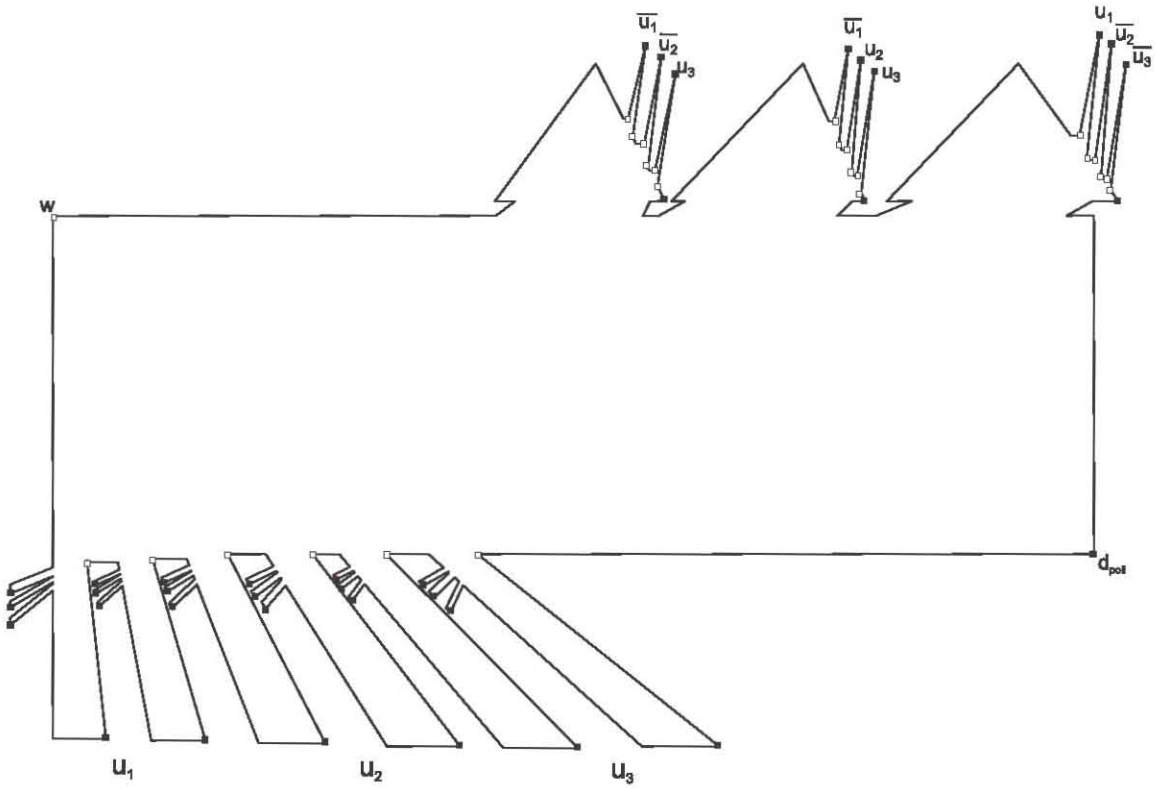


Figura 5.10: Construcción completa del poligono para la fórmula $(\overline{u_1} \vee \overline{u_2} \vee u_3) \wedge (\overline{u_1} \vee u_2 \vee u_3) \wedge (u_1 \vee \overline{u_2} \vee \overline{u_3})$, los cuadros rellenos son el conjunto D y los cuadros vacíos representan los puntos posibles donde insertar un vértice Steiner.

vértices clave del polígono P.

Demostración de la NP-Completitud del problema ISP

Sea I una instancia de 3-SAT, T_I el árbol de Steiner para I y P_I el polígono para I.

Lema 5.1.4. El número de vértices clave en el polígono P_I es igual al número de vértices terminales en el árbol de Steiner T_I.

Demostración. Comenzaremos por contar los vértices clave del polígono P_I. En los patrones de cláusula se tienen los vértices clave $\{a_{j_2}, b_{j_2}, d_{j_2}, g_j\}$, con $j = \{1, ..., m\}$. En los patrones de variable se tienen los vértices clave $\{v_{uf3}, v_{uf7}\}$, con $f = \{1, ..., h\}$. Cada patrón de variable tiene $2q_f$ puntas, donde cada una de ellas tiene un vértice clave, por lo tanto tenemos que por cada patrón de variable se tienen $2q_f$ vértices clave. En total se tiene que el polígono P_I tiene $4m + 2h + 2\sum_{f=1}^{h} q_f$ vértices clave.

Contaremos ahora los vértices terminales del árbol T_I. Cada gráfica $G_j \subset T_I$, con $j = \{1, ..., m\}$, tiene 7 vértices terminales, cada gráfica $Q_f \subset T_I$, con $f = \{1, ..., h\}$, tiene $q_f + 1$ vértices terminales, por último tenemos que la gráfica $A \subset T_I$, tiene h vértices terminales. En total se tienen $7m + \sum_{f=1}^{h} (q_f + 1) + h$ vértices terminales en T_I.

Igualando:

$$4m + 2h + 2\sum_{f=1}^{h} q_f = 7m + \sum_{f=1}^{h} (q_f + 1) + h$$

$$4m + 2h + 2\sum_{f=1}^{h} q_f - (7m + \sum_{f=1}^{h} (q_f + 1) + h) = 0$$

$$4m + 2h + 2\sum_{f=1}^{h} q_f - 7m - \sum_{f=1}^{h} q_f - 2h = 0$$

$$4m - 7m + 2h - 2h + 2\sum_{f=1}^{h} q_f - \sum_{f=1}^{h} q_f = 0$$

$$3m = \sum_{f=1}^{h} q_f$$

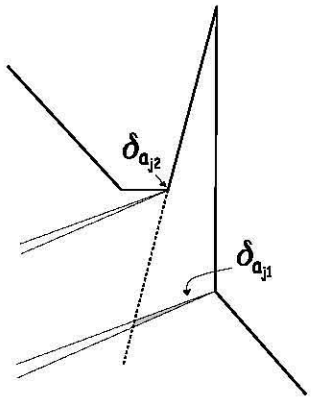
Por la Observación 5.1.2 tenemos que los vértices w_{j_1} o w_{j_3}, con $w \in \{a, b, d\}$ y $j = \{1, ..., m\}$, ven a los vértices clave de una de las dos cavidades de cada patrón de variable, por el Lema 5.1.1 sabemos que solo tres vértices w_{j_1} o w_{j_3} por cláusula son necesarios, de aqui podemos decir que el número de puntas de una de las dos cavidades de cada patrón de variable suman $3m$, por lo tanto el número total de puntas tiene que ser igual a $6m$, es decir $3m = \sum_{f=1}^{h} q_f$. Lo cual concluye la prueba.

□

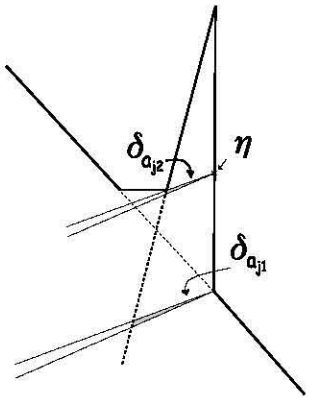
Para la asignación de nodos Steiner es necesario tener en cuenta que dentro del polígono P cualquier punto es un posible candidato para poder incrustar el nodo Steiner. Con las siguientes modificaciones a los patrones, mostraremos que existe una equivalencia entre asignar nodos Steiner a puntos o a vértices.

Modificando el patrón de literales: Sea A una literal de la cláusula C_j. Tomando $\delta_{a_{j1}} = V(a_{j2}) \cap V(g_{j1}) \cap \Delta_{in}(a_{j1})$ como la intersección entre el polígono de visibilidad del vértice clave a_{j2}, el polígono de visibilidad del vértice clave g_{j1} y la punta incidente a a_{j1} (la cual es denotada como $\Delta_{in}(a_{j1})$) obtenemos una región $\delta_{a_{j1}}$ desde la cual cualquier punto perteneciente a dicha región puede conectar a los vértices a_{j2}, g_{j1} y al vértice clave de $\Delta_{in}(a_{j1})$, es decir que se mantienen las propiedades de visibilidad del vértice a_{j1} el cual tambien pertenece a la región. De manera similar definimos la región $\delta_{a_{j2}}$ con la unica diferencia que aqui no se tomara la intersección con el poligono de visibilidad de g_{j1}, ya que dicha intersección seria vacia, por tanto esta región se define como $\delta_{a_{j2}} = V(a_{j2}) \cap \Delta_{in}(a_{j3})$. Notemos que $\delta_{a_{j2}}$ resulta ser el vértice a_{j3} (ver Figura 5.11(a)), supongamos que ahora prolongamos la punta $\Delta_{in}(a_{j3})$, teniendo cuidado de que el vértice a_{j3} se mantenga en la frontera de la punta, hasta intersectar en algún punto η del segmento (a_{j2}, a_{j1}), podemos observar que $\delta_{a_{j2}} = V(a_{j2}) \cap \Delta_{in}(a_{j3})$ ahora es una región que cumple con las propiedades de visibilidad del vértice a_{j3} (ver Figura 5.11(b)).

Modificando el patrón de variable: Este patrón es modificado de manera similar al patrón de literal. Sea $V(w)$ el poligono de visibilidad del vértice w, y Δ_i la *i-esima* punta que es generada en la cavidad izquierda del patrón de variable para u_f. Definimos a la región $\delta_{u_f F} = \bigcap \Delta_i \cap V(w)$ como la intersección de todas las puntas de la cavidad izquierda del patrón para u_f con el polígono de visibilidad de w. Observemos que las propiedades de visibilidad del vértice F se mantienen para cualquier punto dentro de la región $\delta_{u_f F}$. De manera similar definimos $\delta_{u_f T} = \bigcap \Delta_j \cap V(w)$, donde $\bigcap \Delta_j$ es la intersección de todas las puntas de la cavidad derecha del patrón para u_f, de igual forma cualquier vértice dentro de la región $\delta_{u_f T}$ mantiene las propiedades de visibilidad del vértice T. En la Figura 5.12 se muestra esta modificación.



(a) Caso donde $\delta_{a_{j2}}$ es un vértice.



(b) Caso donde $\delta_{a_{j2}}$ es una región.

Figura 5.11: Regiones del patrón de literal para A donde un nodo Steiner puede ser asignado.

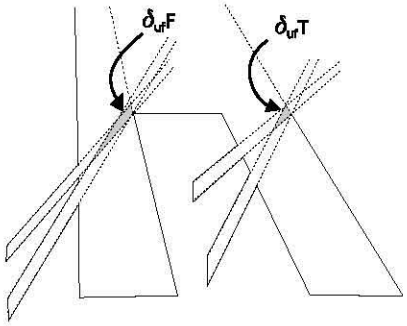


Figura 5.12: Regiones del patrón de variable para u_f donde un nodo Steiner puede ser asignado.

Definición 5.3. Sea $I = C_1 \wedge C_2 \wedge ... \wedge C_m$ una instancia de *3-SAT*, donde $C_j = \{A \vee B \vee D\}$, con $j \in \{1, ..., m\}$, es una cláusula de I; $A \in \{u_i, \overline{u_i}\}$, $B \in \{u_j, \overline{u_j}\}$ y $D \in \{u_k, \overline{u_k}\}$ son literales de C_j; y u_i, u_j y u_k son variables en I; y sean $\delta_{a_{j1}}$ y $\delta_{a_{j2}}$ las regiones donde se puede colocar un nodo Steiner en el patrón de literal para A, $\delta_{b_{j1}}$ y $\delta_{b_{j2}}$ las regiones para B, y $\delta_{c_{j1}}$ y $\delta_{c_{j2}}$ las regiones para C. Si la literal A toma valor de verdad *verdadero* en la instancia I entonces un nodo Steiner es colocado en la región $\delta_{a_{j1}}$, caso contrario, si A toma valor de verdad *falso* en la instancia I entonces un nodo Steiner es colocado en $\delta_{a_{j2}}$. De la misma manera se procede para las literales B y D, y el resto de cláusulas. A esta asignación de nodos Steiner la llamamos *asignación correcta* de nodos Steiner.

Lema 5.1.5. El número de nodos Steiner necesarios para conectar a todos los vértices clave de P_I no aumenta de $3m + h + 1$ si y solo si la asignación de nodos Steiner en el patrón de cláusulas de P_I es una asignación correcta.

Demostración. ($\leftarrow$) Sea μ una asignación correcta de nodos Steiner en el patrón de cláusulas de P_I. Sin perdida de generalidad consideremos que μ conecta a todos los vértices clave de las puntas de la cavidad izquierda. De acuerdo a la Observación 5.1.2 los nodos Steiner deben estar colocados en algun punto dentro de $\delta_{w_{j1}}$ o $\delta_{w_{j2}}$, con $w \in \{a, b, d\}$, de tal forma que un nodo Steiner es requerido en el vértice $v_{u_f 8}$, $f = \{1, ..., m\}$, para cubrir todas las puntas de la cavidad derecha, lo cual no aumenta el número $3m+h+1$ de nodos Steiner necesarios para conectar a los vértices clave de P_I.

($\rightarrow$) Supongamos que existe una asignación ν de nodos Steiner en el patrón de cláusulas de P_I la cual es similar a la asignación μ pero con la diferencia de que uno de los nodos Steiner es colocado dentro de la región $\delta_{w_{j1}}$, con la suposición de que en μ un nodo Steiner fue colocado en $\delta_{w_{j2}}$. Sin perdida de generalidad consideremos que el nodo Steiner colocado en $\delta_{w_{j2}}$ conectaba un vértice clave de la cavidad izquierda del patrón de variable para u_f, donde $f \in \{1, ..., h\}$. De acuerdo a nuestra construcción de las puntas, el nodo Steiner de la asignación ν colocado en $\delta_{w_{j1}}$ podra conectar un vértice clave de la cavidad derecha del patrón de variable para u_f, pero no uno de la cavidad izquierda, dejando asi un vértice clave de la cavidad izquierda sin conectar, de tal forma que para conectar a los vértices clave del patrón de variable para u_f, es necesario colocar un nodo Steiner en $v_{u_f 4}$ y otro en $v_{u_f 8}$, aumentando asi el número de nodos Steiner necesarios a $3m + h + 2$.

□

Teorema 5.1.2. El árbol de Steiner T_I puede ser incrustado en el polígono simple P_I si y solo si la instancia I de *3-SAT* es satisfacible.

Demostración. ($\leftarrow$) Supongamos que la instancia de *3-SAT* es satisfacible, entonces tenemos que existe una configuración de las siete mostradas en el Lema 5.1.1 para colocar los nodos Steiner. Sin perdida de generalidad supongamos que una de ellas considera a d_{j1}, para toda $j = \{1, ..., m\}$. Definiremos la siguiente funcion $\varphi : V_{T_I} \rightarrow V_{P_I}$, como una función de mapeo. Haciendo $\varphi(r_{1j}) = a_{j_2}$, $\varphi(r_{2j}) = b_{j_2}$, $\varphi(r_{3j}) = d_{j_2}$, $\varphi(s_{1j}) = \alpha, \alpha \in \{a_{j_1}, a_{j_3}\}$, $\varphi(s_{2j}) = \beta, \beta \in \{b_{j_1}, b_{j_3}\}$, $\varphi(s_{3j}) = d_{j_1}$, $\varphi(r'_{1j}) = \alpha', \alpha' \in \left\{a'_{j_1}, a'_{j_3}\right\}$, $\varphi(r'_{2j}) = \beta', \beta' \in \left\{b'_{j_1}, b'_{j_3}\right\}$ y $\varphi(r'_{3j}) = d'_{j_1}$ para toda $j = \{1, ..., m\}$. Aseguramos que toda gráfica $G_j \subset T_I$ puede ser incrustada en P_I.

Sea $C = \{C_1, ..., Cm\}$ el conjunto de cláusulas de I. Haciendo la asignación: $\varphi(r_{ufj}) = \gamma, \gamma \in \{a_{j1}, a_{j3}\}$, para toda $j \in \{1, ..., m\}$ tal que $u_f \in C_j$, $\varphi(r_{ufw}) = \Phi, \Phi \in \left\{w_f, w'_f\right\}$, podemos ver que toda gráfica $Q_f \subset T_I$ puede ser incrustada en P_I.

Haciendo la asignación: $\varphi(s_a) = w$, $\varphi(r_{s_f}) = \lambda, \lambda \in \left\{w_f, w'_f\right\} - \Phi$, con $f = \{1, ..., h\}$ aseguramos que la gráfica $A \subset T_I$ puede ser incrustada en P_I.

Dado que $V_{T_I} = \cup_{j=1}^{m} V_{G_j} \cup_{f=1}^{h} V_{Q_f} \cup A$ es el conjunto de vértices de T_I solo hace falta agregar las aristas que conectan a las gráficas G_j, Q_f y A. Dado que $s_a \in V(s_{u_f})$, para toda $f = \{1, ..., h\}$, las aristas (sa, s_{u_f}) pueden ser incrustadas como segmentos rectilineos, uniendo asi las gráficas Q_f con A. Además $s_{u_f} \in V(s_{ij})$, para toda $f = \{1, ..., h\}, i = \{1, 2, 3\}\, yj = \{1, ..., m\}$, por lo que las aristas (s_{u_f}, s_{ij}), para toda $f = \{1, ..., h\}$, $i = \{1, 2, 3\}$ y $j \in \{1, ..., m\}$ tal que $u_f \in C_j$ y $\bigcap(s_{u_f}, s_{ij}) = s_{u_f}$, pueden ser incrustadas como segmentos rectilineos dentro de P_I. Por lo tanto T_I puede ser incrustado en P_I.

($\rightarrow$) Sea C_j, para alguna $j \in \{1, ..., m\}$, una cláusula en I que no es satisfacible. Por el Lema 5.1.2 tenemos que si C_j no es satisfacible entonces tenemos la siguiente asignación $\varphi(s_{1j}) = a_{j3}$, $\varphi(s_{2j}) = b_{j3}$ y $\varphi(s_{3j}) = d_{j3}$, la cual por el Lema 5.1.1 es una asignación que necesita más de tres nodos Steiner para poder conectar a los vértices clave del patrón de la cláusula C_j, aumentando asi el mínimo número de nodos Steiner necesarios para conectar a los vértices clave de P_I. Suponemos que podemos alterar la asignación dada en el Lema 5.1.2 de tal forma que $\varphi(s_{1j}) = a_{j3}$, $\varphi(s_{2j}) = b_{j3}$ y $\varphi(s_{3j}) = d_{j1}$, por el Lema 5.1.5 tenemos que de igual manera hacer esta modificación implica que el mínimo número de nodos Steiner aumente. Por lo tanto el árbol de Steiner T_I no puede ser incrustado dentro del polígono simple P_I. Con esto se concluye la prueba

□

Con la demostración del Teorema anterior se puede concluir entonces que el problema de *3-Satisfacibilidad Booleana* se puede reducir al problema *ISP*. Por lo tanto el Teorema 5.1.1 queda demostrado.

Conclusiones

La finalidad de este trabajo de investigación es la de mostrar que el problema de incrustar un árbol de Steiner al interior de un polígono simple (ISP) pertenece a la clase de problemas NP-Completos. Para lograr esto se hizo uso del método obtenido a partir del Lema 3.1.3 , cuya idea principal es la de reducir un problema perteneciente a la clase de problemas NP-Completos, al problema *ISP*.

Para lograr dicha reducción tomamos el problema de *3-SAT*, el cual es un problema NP-Completo [6], y transformamos la entrada de dicho problema a una entrada aceptada por el problema *ISP*, en este caso a un árbol de Steiner y a un polígono simple con k puntos en su interior. Una vez transformada la entrada del problema *3-SAT* a una entrada aceptada por el problema *ISP*, mostramos que el problema *ISP* tiene solución con la entrada modificada de *3-SAT* si y solo si el problema de *3-SAT* con su entrada original tiene solución. Con esto logramos mostrar que el problema *ISP* pertenece a la clase de problemas NP-Completos.

Este resultado es importante ya que al mostrar que ISP pertenece a la clase de problemas NP-Completos se establece una cota inferior para este problema, con lo cual se da prioridad al desarrollo de algoritmos exponenciales y/o de aproximación, ya que hasta ahora no se conoce algún algoritmo polinomial que pueda resolver con exactitud algún problema en NP-Completo.

Como trabajo futuro, es interesante estudiar la variante en donde el polígono presenta hoyos y establecer la complejidad de esta variante. Otra variación por investigar es la de estudiar el problema en 3 dimensiones en donde en lugar de trabajar con polígonos se trabajaría con poliedros, y el conjunto D estuviera formado por puntos en el espacio.

Referencias

[1] Bagheri, Alireza, and Razzazi, Mohammadreza. *Drawing Planar Graphs Inside Simple Polygons*. LAMBERT Academic Publishing, 2010.

[2] Bagheri, Alireza, and Razzazi, Mohammadreza. *Complexity of Planar Embeddability of Trees inside Simple Polygons*. CoRR, 2009.

[3] Cormen, Tomas H., Leiserson, Charles E., Rivest, Ronald L., and Stein, Cliford. *Introduction to Algorithms*. Second Edition, Mc Graw-Hill, 2001.

[4] O'Rourke, Joseph. *Computational Geometry in C*, Second Edition. Cambridge University Press, 1998.

[5] Ghosh, Subir Kumar. *Visibility Algorithms in the Plane*, First Edition. Cambridge University Press, 2007.

[6] Lee, D.T., and Lin, Arthur K.*Computational Complexity of Art Gallery Problems*, IEEE, Transactions on Information Theory, vol. IT−32, No. 2, 1986.

[7] Promel, H. J., and Steger, Angelika. *The Steiner Tree Problem: A Tour through Graphs, Algorithms, and Complexity.* Vieweg, 2002.

[8] O'Rourke, Joseph. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[9] de Berg, Mark, Cheong, Otfried, van Kreveld, Marc, and Overmars, Mark. *Computational Geometry: Algorithms and Aplications*, Third Edition. Springer, 2008.

[10] Suri, Subhash. *Minimum link paths in polygons and related problems.* (Ph. D. Thesis)-Johns Hopkins University, 1987.

[11] H. ElGindy and D. Avis. *A linear algorithm for computing the visibility polygon from a point*, J. of Algorithms, 1981.

[12] H. Edelsbrunner, L. Guibas and J. Stolfi. *Optimal point location in monotone subdivisions*, SIAM J. Computing, 1986.

[13] D.T. Lee. *Visibility of a simple polygon*, Computer Vision, Graphics and image Processing, 1983.

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

[15] L. Guibas, J. Hershberger, D. Leven, M.Sharir and R. Tarjan. *Linear-time algorithms for visibility and shortest path problems inside a triangulated simple polygon*, Technical Report 218, Computer Science Department, Courant Institute, 1986.

[16] D. Kirkpatrick, *Optimal search in planar subdivisions*, SIAM J. Computing, 1983.

[17] H. ElGindy, *Hierarchical decomposition of polygons with applications*, (PhD thesis)-School of Computer Science, McGill Univ., Montreal, Canada, 1985.

[18] Cortés Valadez, Marcos, *Un algoritmo exponencial para acoplar un árbol de Steiner al interior de un polígono simple*, Tesis de licenciatura, UNAM, Fes Acatlán, México, 2014.