UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

COBERTURA POR BEACONS EN GRÁFICAS GEOMÉTRICAS

TESIS
QUE PARA OPTAR POR EL GRADO DE
MASTER EN CIENCIA DE LA COMPUTACIÓN

PRESENTA
LEANDRO CASUSO MONTERO

TUTOR
DR. JORGE URRUTIA GALICIA
INSTITUTO DE MATEMÁTICAS

CIUDAD UNIVERSITARIA, CD. MX. AGOSTO 2017

Índice general

Capítulo 1

Introducción y Trabajo Previo

1.1. Introducción

En esta tesis se estudian las propiedades y el alcance de la atracción ejercida por *beacons* en gráficas geométricas. Un *beacon* (en español faro o baliza) es un dispositivo que atrae objetos dentro de un dominio dado, ejerciendo sobre estos una atracción magnética que los induce a moverse continuamente hacia él intentando seguir la línea recta que los une, disminuyéndose de manera monótona y *greedy* la distancia Euclidiana entre ambos. Los objetos sobre los que se ejerce la atracción desconocen su entorno y solo se dejan arrastrar por el efecto magnético de la atracción, deslizándose hasta llegar al beacon o hasta quedar atascados en un punto donde localmente no haya dirección en la que se disminuya de manera greedy su distancia Euclidiana al beacon.

En este caso el dominio serán las gráficas geométricas de líneas rectas sobre el plano, donde los vértices tendrán coordenadas cartesianas y las aristas representarán segmentos de líneas rectas que unen vértices. Los beacons podrán ser colocados solo sobre los vértices de la gráfica, y los objetos a atraer podrán colocarse en: solo vértices (variante de vértices), o tanto en vértices como en cualquier punto interior de las aristas (variante de vértices y aristas). Durante su recorrido de atracción, los objetos a atraer deben permanecer siempre sobre la gráfica, deslizándose continuamente sobre las aristas y vértices de esta.

El modelo estudiado en esta tesis es una versión distinta del modelo original planteado por Biro en [2], en el cual el dominio son los polígonos simples. Ambos modelos están motivados por el movimiento de objetos autónomos, como agentes móviles o robots, que siguen una señal dada sin conocer su entorno. Otras posibles aplicaciones mencionadas en [2] están relacionadas con

el ruteo en redes, donde el ruteo geográfico greedy es un método utilizado en la práctica. En todo caso, la mayor ventaja que ofrecen estos modelos es la no necesidad de conocimiento global del entorno o dominio, pudiéndose lograr el ruteo con solo poco conocimiento local de este, lo cual implica una disminución radical de la memoria necesitada y brinda la seguridad de que si la información del objeto es extraída malintencionadamente, esta sea vana respecto al dominio. Específicamente para el modelo de las gráficas geométricas, esto significa que los objetos a atraer solo necesitarán conocer la posición del beacon que los atrae.

Desafortunadamente, durante su recorrido por la gráfica, un objeto sobre el que un beacon ejerce atracción puede quedar atascado en un punto donde localmente no haya dirección en la que se disminuya de manera greedy su distancia Euclidiana al beacon, no pudiendo ser atraído por este. En los ejemplos de aplicaciones mencionados esto implicaría un fallo dada la imposibilidad de alcanzar el destino fijado. Esto hace necesario el estudio de las propiedades de este tipo de atracción para obtener bajo qué condiciones un objeto sobre el que se ejerce atracción llegará al beacon que lo atrae, o para obtener cuál es el área que puede atraer (dominar) un beacon dado, u otras incógnitas relacionadas. La siguiente figura muestra un ejemplo del efecto de atracción en dicho modelo.

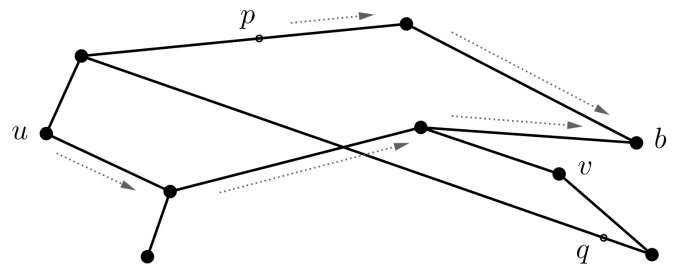


Figura 1.1: Con flechas punteadas están los caminos que tomarían el vértice u y el punto p al ejercer el beacon b atracción sobre ellos. El vértice v y el punto q no tienen dirección hacia donde se disminuya la distancia Euclidiana al beacon b, por lo que quedan estáticos.

Los capítulos siguientes estudian algunas de estas incógnitas. En el Capítulo 2 se establece el modelo de manera más formal definiendo el dominio junto con el tipo de atracción y las variantes, además de dar una serie de proposiciones básicas y establecer la nomenclatura a utilizar. El Capítulo 3 muestra distintos algoritmos para hallar los Conjuntos Atraídos desde un beacon. El Capítulo 4 trata la determinación del Kernel mediante varios algoritmos. El Capítulo 5 trata la Cobertura estableciendo la complejidad y dando algunas

cotas. Para los capítulos 3, 4 y 5 se exponen resultados para el dominio en las dos variantes (vértices, vértices y aristas).

1.2. Trabajo Previo

Este tipo de problema entra en la clasificación de problemas de dominación o cobertura, extensamente estudiados tanto Geometría Computacional como en Teoría de Gráficas.

Respecto a Geometría Computacional, la cobertura por beacons puede ser vista como una extensión de la cobertura por visibilidad, donde los beacons dominan o cubren puntos más allá de la frontera de lo visible. En los problemas de visibilidad la cobertura desde un punto está dada por el conjunto de puntos que pueden ser unidos a este mediante una línea recta contenida completamente dentro del dominio (polígono, poliedro, etc.). Un ejemplo clásico de problema de cobertura por visibilidad es el *Problema de la Galería de Arte*, el cual consiste básicamente en encontrar un conjunto mínimo de guardias que logren vigilar (cubrir, dominar) un polígono. Este problema es NP-difícil como se muestra en [9] y ha sido estudiado en numerosas variantes. En nuestro caso, un beacon puede cubrir o dominar toda la zona que es capaz de atraer, reemplazándose la visibilidad por atracción.

Respecto a Teoría de Gráficas, la cobertura por beacons puede ser vista como una extensión de los problemas de dominación y cobertura de gráficas, donde se dominan o cubren vértices o aristas más allá de la vecindad de un vértice. El *Problema de Dominación* en gráficas consiste en encontrar un conjunto mínimo de vértices de la gráfica tal que todo otro vértice sea vecino de al menos uno de ellos. Este problema es un clásico problema NP-difícil [7] y es muy utilizado para reducciones en pruebas de complejidad de otros problemas. En nuestro caso, un beacon puede dominar todo el conjunto de vértices que puede atraer, reemplazándose la vecindad por atracción.

El modelo de cobertura por beacons es relativamente reciente; fue planteado de manera extensa en el 2013 por Biro en [2] como trabajo de tesis de doctorado, aunque ya se habían presentado trabajos previos como [3] en el 2011. Biro trabajó en su tesis sobre la cobertura y ruteo en polígonos simples, incluyendo los polígonos simples con hoyos y polígonos simples ortogonales. Se estableció allí la "dificultad" de los problemas de cobertura en dichos dominios, y se mostraron variados algoritmos para problemas como: hallar las áreas cubiertas por beacons, hallar el Kernel de los polígonos, entre otros. En esta tesis no se incluyó trabajo en problemas de ruteo, centrándonos solo en la cobertura.

A partir del trabajo de Biro se comenzó el estudio de este modelo. Los

artículos [4] y [5] son resultados del mismo grupo de trabajo de Biro y son derivados de su tesis. Otros autores se han enfocado en mejorar algunas cotas establecidas centrándose en algún subdominio, como polígonos simples ortogonales, tal es el caso de los artículos [1] y [10].

Capítulo 2

Preliminares

2.1. Introducción

En este capítulo estableceremos de manera formal el modelo, definiendo el dominio, los objetos a atraer, los beacons, la atracción y las variantes de problemas. Paulatinamente se introducirá la notación a utilizar y algunos resultados básicos.

2.2. Geometría

En esta sección comenzaremos mencionando algunos resultados de geometría elemental, luego introduciremos algunas notaciones y teoremas básicos de geometría que servirán de apoyo para analizar la atracción.

A continuación mencionamos algunos resultados de geometría elemental que serán utilizados:

- Dada una recta l y un punto b perteneciente o no a esta, existe una y solo una recta perpendicular a l que pasa por b.
- Dada una recta l y un punto b, el punto en l más cercano a b es, o el propio b si $b \in l$, o es el punto b' tal que $\overleftrightarrow{bb'} \perp l$.
- Los ángulos adyacentes suman 180 grados.
- La suma de los ángulos interiores de un triángulo es igual a 180 grados.
- En un triángulo rectángulo, la suma de los cuadrados de los catetos es igual al cuadrado de la hipotenusa (Teorema de Pitágoras).

La siguientes notaciones geométricas serán utilizadas durante el trabajo:
Sean dos puntos distintos p y q.

- Por pq nos referiremos al segmento cerrado entre p y q.
- Por (pq) nos referiremos al segmento abierto entre p y q.
- Por $\overrightarrow{pq}$ al rayo infinito que parte desde p y pasa por q.
- Por $\overleftrightarrow{pq}$ a la recta que pasa por p y q.

Por simplicidad, siempre que se haga referencia a un segmento u otro elemento geométrico definido por dos puntos, asumiremos que estos son distintos. En caso de ser necesario se aclarará la posibilidad de que sean iguales.

Sean tres puntos distintos p, q y b. Por $\sphericalangle pqb$ nos referiremos a la amplitud del ángulo menor de los dos delimitados por pq y qb, por lo cual siempre se cumplirá que $0 \leq \sphericalangle pqb \leq 180$. Por $\measuredangle pqb$ nos referiremos a la amplitud del ángulo formado por qb partiendo de pq girando en sentido contrario a las manecillas del reloj, por lo que su amplitud sí puede ser mayor a 180; sus valores estarán entre $0 \leq \measuredangle pqb < 360$.

Intentando hacer más compacto el lenguaje utilizado, a partir de ahora nos referiremos simplemente por *distancia* a la distancia Euclidiana, y utilizaremos la función $d(p, b)$ para obtener su valor entre dos puntos p y b.

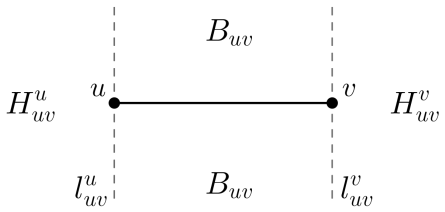
Respecto a un segmento uv, definiremos a continuación algunas rectas y áreas notables:

Definición 2.2.1. Sea uv un segmento. Las *rectas soporte perpendiculares* l^u_{uv} y l^v_{uv}, de uv, son las rectas perpendiculares a $\overleftrightarrow{uv}$ que pasan por u y v respectivamente.

Definición 2.2.2. Sea uv un segmento. Llamaremos H^u_{uv} (respectivamente H^v_{uv}), al semiplano que parte inclusivamente desde l^u_{uv} (respectivamente l^v_{uv}) en dirección contraria a uv.

Definición 2.2.3. Sea uv una arista. Llamaremos *banda* B_{uv} de uv, al área situada estrictamente entre las rectas soporte perpendiculares de uv.

El lector notará que respecto a uv, las áreas formadas por H^u_{uv}, H^v_{uv} y la banda respectiva B_{uv}, son ajenas y particionan el plano Euclidiano.



Los siguientes resultados servirán de apoyo geométrico para analizar la atracción:

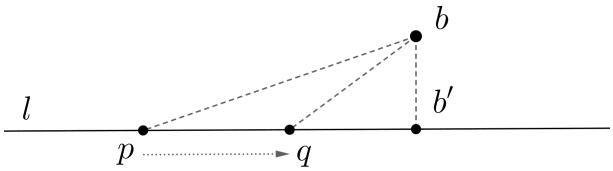
Proposición 2.2.4. *Sea l una recta y b un punto. Sea b' el punto de l más cercano al punto b. La función distancia hacia b, $d(x) = d(x, b)$, de los puntos $x \in l$, es monótona decreciente en la medida que x se acerque al punto b'.*

Demostración. Sean p y q dos puntos distintos de l tal que q está más cerca de b' que p, es decir $d(q, b') < d(p, b')$. Como b' es el punto de l más cercano a b, entonces $b' = b \in l$ o $\overleftrightarrow{bb'} \perp l$. Si $b' = b \in l$, tomaremos $\overleftrightarrow{bb'}$ como una recta vacía la cual cumple $\overleftrightarrow{bb'} \perp l$.

Entonces los triángulos $bb'p$ y $bb'q$ (posiblemente nulos si $b' = b \in l$) son rectángulos en b', por lo que aplicando Pitágoras

$$d(p, b) = \sqrt{d(p, b')^2 + d(b', b)^2}.$$
$$d(q, b) = \sqrt{d(q, b')^2 + d(b', b)^2}.$$

Entonces como $d(q, b') < d(p, b')$ y ambas son no negativas, $d(q, b')^2 < d(p, b')^2$. Luego, $\sqrt{d(q, b')^2 + d(b', b)^2} < \sqrt{d(p, b')^2 + d(b', b)^2}$, por lo que $d(q, b) < d(p, b)$. Luego para cualquiera dos puntos p y q sobre l, con q más cercano a b' que p, $d(q, b) < d(p, b)$, por lo que $d(x) = d(x, b)$ es monótona decreciente en la medida que x se acerque a b'. □



Corolario 2.2.5. *Sea l una recta y b un punto. Sea b' el punto de l más cercano al punto b y sea pq un segmento en l tal que q está más cerca de b' que p. La función distancia hacia b, $d(x) = d(x, b)$, de los puntos $x \in pq$, es monótona decreciente en la medida que x se acerque al punto q.*

Demostración. Como q está entre cualquier punto $x \in pq$ y b', en la medida que x se acerque a q entonces se acercará a b', por lo que $d(x)$ será monótona decreciente en el intervalo. □

Proposición 2.2.6. *Sea uv un segmento y b un punto. Sea b' el punto de $\overleftrightarrow{uv}$ más cercano al punto b. Entonces se cumple que b' está en la misma área que b de las definidas por uv, es decir:*
Si $b \in H^u_{uv}$, entonces $b' \in H^u_{uv}$.
Si $b \in H^v_{uv}$, entonces $b' \in H^v_{uv}$.
Si $b \in B_{uv}$, entonces $b' \in B_{uv}$.

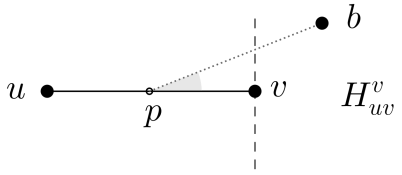
Demostración. Si $b \in \overleftrightarrow{uv}$, entonces como $b' = b \in \overleftrightarrow{uv}$, trivialmente b' estará en la misma área que b.

Como b' es el punto de $\overleftrightarrow{uv}$ más cercano a b, entonces $\overleftrightarrow{bb'} \perp \overleftrightarrow{uv}$. Por definición, tenemos que las rectas soporte perpendiculares de uv, l^u_{uv} y l^v_{uv}, cumplen que $l^u_{uv} \perp \overleftrightarrow{uv}$ y $l^v_{uv} \perp \overleftrightarrow{uv}$. Entonces tiene que cumplirse que $l^u_{uv} \parallel \overleftrightarrow{bb'} \parallel l^v_{uv}$, pues todas son perpendiculares a $\overleftrightarrow{uv}$. Si b' estuviese en un área distinta que b, entonces bb' cortaría alguna de las rectas l^u_{uv} o l^v_{uv} por estar en semiplanos distintos, por lo que entonces sería imposible que $\overleftrightarrow{bb'}$ fuese paralela a estas. Luego, b y b' tiene que estar en la misma área de las definidas por uv. □

Proposición 2.2.7. *Sea uv un segmento y $b \in H^v_{uv}$ un punto. Para todo punto p en el interior de uv, es decir $p \in (uv)$, se cumple que $\sphericalangle bpv < \sphericalangle bpu$.*

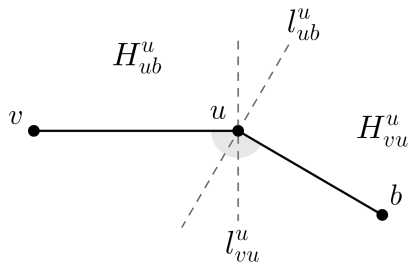
Demostración. Sea l_p la recta en p tal que $l_p \perp \overleftrightarrow{uv}$. Como $p \in (uv)$ y $l_p \perp \overleftrightarrow{uv}$, entonces $p \in B_{uv}$ y $l_p \in B_{uv}$. Como b y v están en H^v_{uv}, y $l_p \in B_{uv}$, entonces b y v están estrictamente en el mismo semiplano abierto de los definidos por l_p, por lo que $\sphericalangle bpv < 90$.

Luego como $\angle bpv$ y $\angle bpu$ son adyacentes y $\sphericalangle bpv < 90$, entonces $\sphericalangle bpu > 90$. Luego $\sphericalangle bpv < \sphericalangle bpu$. □



Proposición 2.2.8. *Sean vu y ub dos segmentos consecutivos en el punto u. Entonces $b \in H^u_{vu} \iff v \in H^u_{ub}$*

Demostración. Si $b \in H^u_{vu}$ entonces b y v están en semiplanos distintos de los definidos por l^u_{vu} y $\sphericalangle buv \geq 90$. Entonces respecto a l^u_{ub}, b y v tienen que estar en semiplanos distintos, pues si estuviesen en el mismo semiplano $\sphericalangle buv < 90$. Luego como respecto a l^u_{ub}, b y v están en semiplanos distintos, y $b \notin H^u_{ub}$, entonces $v \in H^u_{ub}$. Homólogamente si se parte de que $v \in H^u_{ub}$. □



2.3. Dominio

El dominio general sobre el que se trabajará serán: gráficas simples conexas geométricas de líneas rectas sobre el plano.

Mucha de la escritura de este trabajo se apoya en conceptos de gráficas, y se utilizarán sin definir conceptos comunes de Teoría de Gráficas como: conjunto de vértices y aristas, adyacencia, arco, camino, camino simple o trayectoria, ciclo, conexidad, componente conexa y otros, así como algunos resultados básicos de esta teoría. Como referencia para el lector, se puede utilizar [6] para encontrar estos conceptos y resultados. Aun así, estableceremos la nomenclatura general a utilizar y los tipos de gráficas sobre las que trabajaremos para dejar claro el contexto.

Definición 2.3.1. Una *gráfica* $G = (V, E)$ es un par ordenado (V, E) donde V es un conjunto de elementos llamados vértices y E es un conjunto de pares no ordenados $\{u, v\}$ de elementos de V llamados aristas.

Por simplicidad, para referirnos a una arista $\{u, v\}$ utilizaremos la notación uv. Cuando se necesite diferenciar los conjuntos de vértices y aristas de una gráfica G respecto a otra, utilizaremos la notación $V(G)$ y $E(G)$ para referirnos a dichos conjuntos. Estableciendo un lenguaje más algorítmico, llamaremos $Adj[u]$ a la lista (conjunto) de vértices adyacentes a un vértice u. Llamaremos $d(u)$ al grado de u en G, por lo que $d(u) = |Adj[u]|$. En el contexto de los vértices adyacentes a u, llamaremos v_i^u con $1 \leq i \leq d(u)$ al i-ésimo vértice en la lista de adyacencia de u. Al analizar el tiempo de ejecución de un algoritmo sobre G tomaremos en cuenta su número de vértices $|V|$ y de aristas $|E|$, pero para simplificar su escritura adoptaremos la convención de referirnos a estos solo por V y E dentro de la notación O (*big-O*) y en los análisis de complejidad.

Las gráficas sobre las que trabajaremos serán siempre simples, es decir, no tendrán lazos (aristas desde un vértice a sí mismo) ni aristas paralelas (dos aristas entre dos mismos vértices). Por la naturaleza del problema es conveniente enfocarnos solo en gráficas conexas, ya que un beacon no podrá nunca atraer a un objeto situado en otra componente conexa. En caso de no conexidad, los resultados hallados serían aplicables a cada componente conexa por separado. A partir de ahora toda referencia a la gráfica implicará que esta es simple y conexa.

Definición 2.3.2. Una *gráfica geométrica*, sobre el plano, es una gráfica $G = (V, E)$ donde los vértices tienen coordenadas cartesianas en el plano Euclidiano y las aristas representan segmentos de líneas rectas que unen vértices.

Como la gráfica y el dibujo asociado a esta en el plano son íntegramente correspondientes, nos referiremos por vértice indistintamente al vértice de la gráfica como al punto en el plano correspondiente a sus coordenadas; de igual forma nos referiremos por arista indistintamente a la arista de la gráfica como a los segmentos de recta que estas representan. Por simplicidad para nuestro trabajo, nos referiremos a estas solo como gráficas.

Se pedirá que no dos vértices ocupen la misma posición en el plano y que no tres vértices sean colineales. En el sentido general y a menos que se aclare lo contrario, la gráfica admitirá cruces (i.e aristas que se cortan) sin que el punto de cruce sea necesariamente un vértice de la gráfica.

Usualmente durante este trabajo hablaremos de *gráficas densas* y *gráficas dispersas*. Por gráficas densas nos estaremos refiriendo a gráficas donde el número de aristas es cercano al número de máximo de aristas, i.e $|E|$ es cercano a $|V|(|V|-1)/2$, y representaremos esto abusando de la notación O como $E \in O(V^2)$. Por gráficas dispersas nos estaremos refiriendo a gráficas donde el número de aristas es cercano al número de vértices, y representaremos esto abusando de la notación O como $E \in O(V)$.

A partir de este punto, en las subsecuentes definiciones, teoremas, y resultados en general, asumiremos siempre que los vértices, aristas, puntos y beacons mencionados, pertenecen a una gráfica geométrica $G = (V, E)$.

2.4. Objetos

Los objetos a atraer no tienen forma y pueden ser vistos como puntos que serán situados sobre la gráfica; pueden ser colocados tanto en vértices como en cualquier punto interior de las aristas. Durante su recorrido de atracción los objetos deben permanecer siempre sobre la gráfica, deslizándose continuamente sobre las aristas y vértices de esta. Además desconocen la forma de la gráfica y no tienen memoria, por lo que no son capaces de recordar ningún dato o punto visitado de la gráfica.

Bajo la acción de un beacon los objetos solo se dejan arrastrar por el efecto magnético de la atracción, hasta llegar al beacon o hasta quedar atascados en un punto donde no haya dirección en la que se disminuya de manera greedy su distancia al beacon. La velocidad con que los objetos se mueven por la gráfica no es relevante para nuestro modelo, solo diremos aquí que estos se mueven con alguna velocidad positiva y constante, aunque bastaría exigir que esta fuera positiva y no infinitesimal.

Fuera de estas consideraciones los objetos a atraer no tienen ninguna otra característica y durante la escritura serán vistos como puntos propios de la gráfica.

2.5. Beacons y Atracción

Los beacons son dispositivos sin forma y pueden ser vistos como puntos que serán situados sobre vértices de la gráfica; su colocación sobre puntos interiores de las aristas no está permitido. Al ser activados, los beacons ejercen una atracción magnética sobre algunos objetos colocados en la gráfica, induciéndolos a moverse continuamente hacia él intentando seguir la línea recta que los une, disminuyéndose de manera monótona y greedy la distancia entre ambos. En este capítulo se darán una serie de definiciones respecto a los beacons y la atracción ejercida por estos.

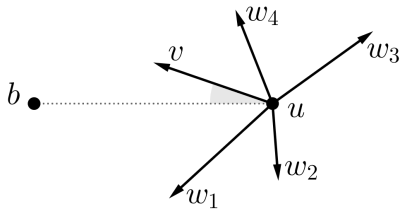
Definición 2.5.1. Un *beacon* en una gráfica geométrica G, es un vértice de G que puede ser activado para ejercer atracción (acción) sobre objetos puntuales colocados en G.

La atracción (acción) ejercida por un beacon, sobre un objeto, lo induce a moverse continuamente hacia él intentando seguir la línea recta que los une. Si el objeto puede moverse en dirección al beacon sobre dicha línea recta lo hará, pero en caso de que no sea posible entonces el objeto será inducido a tomar una dirección lo más cercana angularmente posible a esta.

Note que este fenómeno ocurre interviniendo solo aspectos locales (direcciones desde el objeto) y sin conocerse qué pudiera ocurrir más adelante al tomar una u otra dirección, seleccionándose simplemente una especie de mínimo local respecto a la dirección de acercamiento. Es por ello que para referirnos a la dirección elegida le añadiremos el adjetivo de greedy.

Para modelar formalmente este fenómeno enunciamos entonces la siguiente definición:

Definición 2.5.2. Sea b un beacon, u un punto con $b \neq u$, y R un conjunto de rayos direccionales que parten desde u. La dirección $\overrightarrow{uv} \in R$ es *greedy* respecto a b si, $\forall\ \overrightarrow{uw} \in R$ con $\overrightarrow{uw} \neq \overrightarrow{uv}$, $\sphericalangle buv < \sphericalangle buw$.



De la definición anterior podemos inferir que solo puede haber una dirección greedy, dado que la condición de comparación entre los ángulos formados es estricta. Si no existe una dirección que cumpla estrictamente la comparación con todas las demás, entonces el objeto quedará atascado en el punto actual debido a la indeterminación.

Luego de tener definido el aspecto dirección, es también necesario definir el desplazamiento y el tipo de desplazamiento inducido por la atracción. Como planteamos anteriormente, no es relevante para nuestro modelo la velocidad del movimiento de los objetos, por lo que omitiremos referirnos a esta en las definiciones.

Definición 2.5.3. Sea uv un segmento. El *desplazamiento* uv, es un movimiento continuo de un objeto puntual que va desde u hasta v manteniéndose siempre sobre uv.

Definición 2.5.4. Sea b un beacon y uv un segmento. El desplazamiento uv es *monótono* respecto a b si, la función de distancia a b, $d(x) = d(x, b)$, de los puntos $x \in uv$, es monótona decreciente en la medida que x se acerque a v.

Definición 2.5.5. Sea b un beacon y uv un segmento. El desplazamiento uv es *monótono greedy* respecto a b si, es monótono respecto a b, y para cada punto $p \neq v$ en uv, la dirección $\overrightarrow{pv}$ es greedy respecto a b entre todas las direcciones posibles que puede tomar p en G.

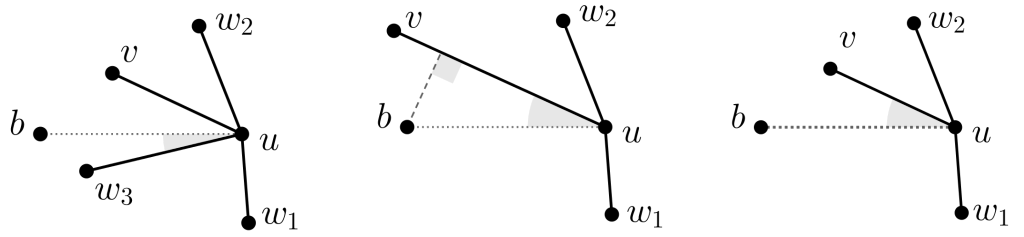


Figura 2.1: Desplazamiento uv respecto a b: en la figura (a) uv es monótono pero no greedy, en la figura (b) uv es greedy pero no monótono, en la figura (c) uv es monótono greedy.

Luego de tener definidos los aspectos de dirección y desplazamiento, podemos dar ya una definición concisa de la atracción:

Definición 2.5.6. La *atracción* (*acción*) ejercida por un beacon b sobre un objeto puntual p, en una gráfica geométrica G, induce a p a moverse sobre G mediante desplazamientos monótonos greedy respecto a b.

Desafortunadamente, pueden existir en la gráfica puntos desde los cuales no existan desplazamientos monótonos greedy respecto a un beacon, por lo que quedarán atascados en su posición bajo la acción de este. A dichos puntos los llamaremos puntos muertos como enuncia la siguiente definición:

Definición 2.5.7. Un punto d es un *punto muerto* (*dead point*) respecto a un beacon b si, $d \neq b$, y bajo la acción de b un objeto colocado en d permanece estacionario, es decir, $\nexists$ segmento $dv \in G$ tal que el desplazamiento dv sea monótono greedy respecto a b.

Como podemos deducir de la definición, un punto puede ser punto muerto respecto a un beacon por dos razones: o desde el punto no existe dirección greedy (una estricta) respecto al beacon, o tomando la dirección greedy no existe desplazamiento monótono (de disminución) respecto al beacon.

En la Figura 2.2 podemos observar que el vértice v y el punto q son puntos muertos respecto a b. El vértice v no tiene desplazamiento monótono. El punto q está justo sobre la perpendicular a b, por lo que no tiene dirección greedy (estricta) ni tampoco desplazamiento monótono.

Continuaremos esta sección enunciaremos algunas definiciones respecto al efecto de atracción ejercido por un beacon:

Definición 2.5.8. Un punto p, bajo la acción de un beacon b, *alcanza* un punto q si, un objeto colocado en p se mueve por G mediante desplazamientos monótonos greedy hasta que eventualmente su distancia a q sea 0.

Definición 2.5.9. Un beacon b *atrae* un punto p si, bajo la acción de b, p alcanza b. Decimos también que en tal caso p es *atraído* por b.

Evidentemente de la definición anterior inferimos que un vértice v es atraído por un beacon b si b atrae el punto que representa v.

Una de las proposiciones que demostraremos posteriormente en esta tesis es que la atracción ejercida por un beacon sobre puntos interiores de una arista conlleva a un resultado equivalente para todos ellos, es decir, todos finalmente alcanzarán el mismo punto en G. Esto implica que el interior de la arista puede ser visto como un todo respecto a la atracción y que el tratamiento por regiones no es necesario, por lo que hablaremos en términos de aristas y vértices atraídos, y no en términos de conjunto de puntos atraídos.

Definición 2.5.10. Una arista uv, bajo la acción de un beacon b, *alcanza* un punto p si, todos sus puntos interiores alcanzan p.

Definición 2.5.11. Un beacon b *atrae* una arista uv si, uv alcanza b. Decimos también que en tal caso uv es atraída por b.

Respecto a la atracción ejercida por un beacon sobre un vértice y a la arista (dirección) tomada por este, exponemos las siguientes definiciones:

Definición 2.5.12. Sea b un beacon. La arista uv es *de atracción* desde u respecto a b si, bajo la acción de b, el vértice u alcanza v mediante uv, es decir, el desplazamiento uv es monótono greedy.

Definición 2.5.13. Sea b un beacon. La arista uv es *de atracción* respecto a b si, es de atracción desde u o desde v respecto a b.

En la Figura 2.1c la arista uv es de atracción de u a b, puesto que el desplazamiento uv es monótono greedy respecto a b.

Queremos enfatizar que la gráfica geométrica nunca cambiará su forma (posición en el plano), ya que el efecto de atracción solo afecta realmente a objetos que se mueven sobre esta. Hacemos la aclaración de manera explícita para evitar que alguna afirmación sobre la atracción de un vértice o arista haga pensar al lector que la gráfica cambia su forma.

2.6. Propiedades de la Atracción

En esta sección enunciaremos algunas propiedades de la atracción que nos darán las herramientas necesarias para el estudio de problemas relacionados con el modelo. Nuestro objetivo aquí es poder llegar a resultados que nos permitan discretizar el efecto de la atracción.

Proposición 2.6.1. *Sea b un beacon, entonces b atrae al vértice que representa el propio b en la gráfica geométrica.*

Demostración. Trivialmente, un objeto que parte desde el punto b ya se encuentra sobre b, por lo que ya ha logrado alcanzar al beacon. □

Proposición 2.6.2. *Sea b un beacon, entonces b atrae a toda arista incidente en b y a todo vértice adyacente a b.*

Demostración. Sea una arista ub incidente en b. Para todo punto interior p de ub, la dirección $\overrightarrow{pb}$ estaría disponible en G, por lo que un objeto colocado en p se deslizaría por el segmento de línea recta que los une hasta alcanzar b.

Para un vértice u adyacente a b, de igual forma la dirección en línea recta estaría disponible, por lo que u alcanzaría b. Como no se admite que tres vértices sean colineales, no puede existir otra arista de u que vaya en línea recta hacia b, por lo que no hay ambigüedad en la elección. □

Proposición 2.6.3. *Si bajo la acción de un beacon b, un punto p alcanza un punto q, entonces p correrá la misma suerte que q bajo la acción de b.*

Demostración. Al alcanzar p el punto q bajo la acción de b, entonces podemos ver ya a p como q, por lo que correrá la misma suerte que q respecto a b. □

Corolario 2.6.4. *Si bajo la acción de un beacon b, un punto p alcanza un punto muerto d respecto a b, entonces p permanecerá estacionario en d.*

Demostración. Directamente de la proposición anterior. Al alcanzar p el punto d bajo la acción de b, entonces correrá la misma suerte que d respecto a b, por lo que permanecerá estacionario en d. □

Corolario 2.6.5. *Si bajo la acción de un beacon b, un punto p alcanza un punto q atraído por b, entonces p es atraído por b.*

Demostración. Directamente de la proposición anterior. Al alcanzar p el punto q bajo la acción de b, entonces correrá la misma suerte que q respecto a b, por lo que será atraído. □

Corolario 2.6.6. *Si un beacon b atrae un punto p, entonces todo punto q, alcanzado por p bajo la acción de b, es también atraído por b.*

Demostración. Directamente de la proposición anterior. Al alcanzar p el punto q bajo la acción de b, entonces correrá la misma suerte que q respecto a b, por lo que si q no fuese atraído entonces sería imposible que p lo fuese. □

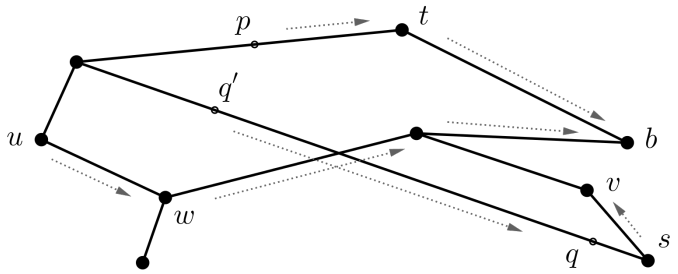


Figura 2.2: El punto p alcanza t bajo la acción de b. Como t es atraído entonces p también lo es. Igualmente el vértice u alcanza w que es atraído, por lo que u también es atraído. El vértice v y el punto q son puntos muertos respecto a b. El punto q' alcanza q bajo la acción de b, por lo que quedará estático en ese punto. Igualmente el vértice s alcanza v, por lo que quedará estático en v.

Nos centraremos ahora en la atracción ejercida por un beacon en el contexto de un segmento y en específico de una arista, para mostrar el comportamiento equivalente de todos sus puntos respecto al beacon.

Teorema 2.6.7. *Sea uv un segmento y b un beacon. Si $b \in H^v_{uv}$, y si para todo punto $p \in (uv)$, p solo tiene disponibles las direcciones $\overrightarrow{pu}$ y $\overrightarrow{pv}$, entonces todo punto en (uv) alcanzará v. Si además, $\overrightarrow{uv}$ es la dirección greedy desde u respecto a b, entonces también u alcanzará v.*

Demostración. Sea b' el punto de $\overleftrightarrow{uv}$ más cercano a b. Como $b \in H^v_{uv}$, entonces $b' \in H^v_{uv}$ por 2.2.6. Entonces como $b' \in H^v_{uv}$, y u, v, b' están sobre $\overleftrightarrow{uv}$, se cumple que v está más cerca de b' que u. Entonces la función distancia hacia b, $d(x) = d(x, b)$, de los puntos $x \in uv$, es monótona decreciente en la medida que x se acerque a v por 2.2.5. Luego, los desplazamientos hacia v son monótonos respecto a b.

Para todo punto $p \in (uv)$, como $b \in H^v_{uv}$, entonces $\sphericalangle bpv < \sphericalangle bpu$ por 2.2.7. Entonces, la dirección pv es greedy respecto b entre las dos posibles $\overrightarrow{pu}$ y $\overrightarrow{pv}$. Luego, para todo punto en (uv) la dirección hacia v es greedy.

Como en (uv) los desplazamientos hacia v son monótonos, y para cada punto la dirección hacia v es greedy, los desplazamientos son monótonos greedy. Luego, todo punto en (uv) alcanzará v.

Como el desplazamiento uv es monótono, si la dirección $\overrightarrow{uv}$ es greedy, el desplazamiento uv sería monótono greedy. Luego, u alcanzaría v. □

Corolario 2.6.8. *Sea b un beacon y uv una arista. La atracción ejercida por b sobre puntos interiores de uv, conlleva a un resultado equivalente para todos ellos, ocurriendo una de las siguientes variantes (ver Figura 2.3):*

Si $b \in H^u_{uv}$, entonces todos alcanzarán u.

Si $b \in H^v_{uv}$, entonces todos alcanzarán v.

Si $b \in B_{uv}$, entonces $\exists$ un punto muerto en (uv) respecto a b y todos lo alcanzarán.

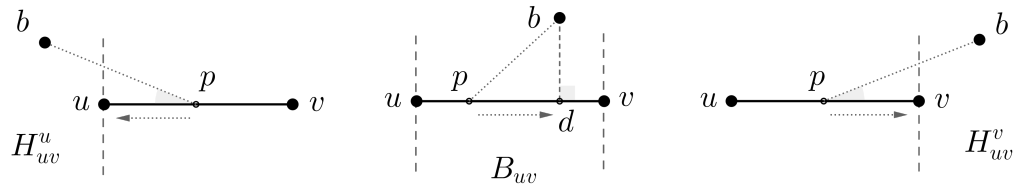
Demostración. Para todo punto interior p de uv, las únicas direcciones disponibles en G son $\overrightarrow{pu}$ y $\overrightarrow{pv}$, puesto que p solo se puede desplazar sobre uv.

Si $b \in H^u_{uv}$, entonces por el Teorema 2.6.7 todos alcanzarán u.

Si $b \in H^v_{uv}$, entonces por el Teorema 2.6.7 todos alcanzarán v.

Si $b \in B_{uv}$, entonces el punto b' en l más cercano a b está también en B_{uv} por 2.2.6. Entonces el punto b' no tiene dirección greedy en G respecto a b, puesto que las dos dirección posibles $\overrightarrow{b'u}$ y $\overrightarrow{b'v}$, generan ángulos iguales respecto a b ($\sphericalangle bb'u = 90 = \sphericalangle bb'v$) por $\overleftrightarrow{bb'} \perp \overleftrightarrow{uv}$. Luego, b' sería un punto muerto respecto a b.

Dividamos el segmento uv por el punto b' en dos segmentos ub' y $b'v$. El beacon b se encuentra entonces en los $H_{b'}$ respectivos de los segmentos ub' y $b'v$, por estar para ambos en las respectivas $l_{b'}$. Entonces, aplicando el Teorema 2.6.7 a cada segmento, todos sus puntos interiores alcanzarían b'. □



(a) (b) (c)

Figura 2.3: La figura (a) corresponde a cuando $b \in H^u_{uv}$. La figura (b) corresponde a cuando $b \in B_{uv}$. La figura (c) corresponde a cuando $b \in H^v_{uv}$.

Corolario 2.6.9. *Sea b un beacon y uv una arista, y sea $\overrightarrow{uv}$ la dirección greedy desde u respecto a b. La atracción ejercida por b sobre u lleva una de las siguientes variantes:*

Si $b \in H^u_{uv}$, entonces u queda en u por ser punto muerto respecto a b.

Si $b \in H^v_{uv}$, entonces u alcanzará v.

Si $b \in B_{uv}$, entonces $\exists$ un punto muerto en (uv) respecto a b y u lo alcanzará.

Demostración. Si $b \in H^u_{uv}$, entonces por el teorema anterior todos los puntos interiores de uv alcanzarán u. Luego, los desplazamientos hacia u desde uv son monótonos, por lo que es imposible desde u haya desplazamiento monótono sobre uv. Como la dirección $\overrightarrow{uv}$ es la dirección greedy respecto a b desde u, y no existe desplazamiento monótono sobre esta, entonces u queda atascado siendo punto muerto respecto a b. Luego, todos los puntos interiores de uv y u quedan en u.

Si $b \in H^v_{uv}$, entonces por el Teorema 2.6.7 u alcanzará v.

Si $b \in B_{uv}$, entonces por el mismo razonamiento del corolario anterior, $\exists$ un punto muerto en (uv) respecto a b y u lo alcanzará. □

Corolario 2.6.10. *Sea b un beacon y uv una arista. La arista uv alcanza v (respectivamente u) $\iff$ $b \in H^v_{uv}$ (respectivamente $b \in H^v_{uv}$).*

Demostración. Si $b \in H^v_{uv}$ (respectivamente $b \in H^v_{uv}$) entonces todos sus puntos interiores alcanzan v (respectivamente u). En otro caso los puntos interiores alcanzarán un punto muerto en (uv) o alcanzarán u (respectivamente v). □

Corolario 2.6.11. *Sea b un beacon y uv una arista. La arista uv alcanza v (respectivamente u) $\iff$ $\sphericalangle uvb \geq 90$ (respectivamente $\sphericalangle vub \geq 90$).*

Demostración. Si $\sphericalangle uvb \geq 90$ (respectivamente $\sphericalangle vub \geq 90$) entonces $b \in H^v_{uv}$ (respectivamente $b \in H^v_{uv}$) por lo que todos sus puntos interiores alcanzan v

(respectivamente u). En otro caso $b \notin H^v_{uv}$ (respectivamente $b \notin H^v_{uv}$) por lo que los puntos interiores alcanzarán un punto muerto en (uv) o alcanzarán u (respectivamente v). □

2.7. Variantes

Todos los resultados de este trabajo están divididos en dos variantes de problemas: variante de *vértices*, y variante de *vértices y aristas*.

La variante de vértices se refiere a la atracción ejercida inicialmente solo sobre vértices, por lo que los algoritmos y demás resultados se enfocan solo en la atracción de los vértices de la gráfica.

La variante de vértices-aristas se refiere a la atracción ejercida inicialmente tanto sobre vértices como sobre aristas, por lo que los algoritmos y demás resultados se enfocan en la atracción de ambos.

El lector seguramente intuirá que la segunda variante es más difícil computacionalmente que la primera. En algunos casos ambas variantes presentan complejidades similares, sobre todo cuando las gráficas son dispersas, pero en general la segunda es efectivamente más compleja.

Capítulo 3

Conjuntos Atraídos

3.1. Introducción

En este capítulo expondremos algoritmos para encontrar el conjunto de vértices y el conjunto de vértices y aristas atraídos por un beacon. Ambos conjuntos serán definidos en la sección siguiente, junto con algunas otras definiciones y resultados sobre los elementos involucrados y el aspecto de la región atraída por un beacon. En las secciones posteriores expondremos los algoritmos para las dos variantes tratadas, mostrando la correctitud de estos y sus complejidades computacionales. Enfatizamos que en todas las definiciones y resultados mencionados estaremos haciendo referencia a una gráfica geométrica $G = (V, E)$.

3.2. Preliminares

Definición 3.2.1. Sea b un beacon. Mediante $A_V(b)$ denotaremos el conjunto de *vértices atraídos* por b.

Definición 3.2.2. Sea b un beacon. Mediante $A_E(b)$ denotaremos el conjunto de *aristas atraídas* por b.

Definición 3.2.3. Sea b un beacon. Mediante $A(b)$ denotaremos el conjunto de *vértices y aristas atraídos* por b, es decir, $A(b) = A_V(b) \cup A_E(b)$.

Además de las definiciones respecto a los conjuntos atraídos, definiremos otros elementos para mostrar algunos resultados sobre el aspecto de la región atraída por un beacon.

Definición 3.2.4. Sea b un beacon. Mediante E_b denotaremos al conjunto de *aristas de atracción* respecto a b.

Nótese que en la definición anterior no se pide que las aristas en E_b sean atraídas por el beacon, solo que sean desplazamientos monótonos greedy en algún sentido. Esto asegura que bajo la acción de b, para cada arista $uv \in E_b$, o u alcanza v, o v alcanza u, siempre mediante uv.

Definición 3.2.5. Sea b un beacon. La *orientación por atracción* de una arista $uv \in E_b$, es la orientación de uv que la convierte en un arco (arista dirigida) donde el vértice destino es el vértice alcanzado bajo la acción de b.

Definición 3.2.6. Sea b un beacon. La *subgráfica abarcadora de atracción* F_b, está formada por los vértices de G y las aristas de E_b.

La subgráfica abarcadora de atracción F_b contiene justamente las aristas importantes para la atracción sobre vértices, puesto que una arista que no esté en F_b no es monótona greedy, y por tanto nunca será tomada como camino al beacon desde un vértice. El siguiente resultado caracteriza la estructura de la subgráfica abarcadora de atracción F_b.

Teorema 3.2.7. *Sea b un beacon. La subgráfica abarcadora de atracción F_b es un bosque (forest), es decir, F_b es acíclica.*

Demostración. Supongamos que F_b contiene un ciclo C. Como cada arista en C pertenece a F_b entonces todas pertenecen a E_b, por lo que son desplazamientos monótonos greedy respecto a b. Sea C' la digráfica con los vértices y aristas de C, orientando por atracción las aristas.

Partiendo desde un vértice inicial v, recorramos tanto como sea posible los arcos de C' en sentido contrario a su orientación, es decir, de destino a fuente. El recorrido se hará mientras sea posible ir de destino a fuente y hasta alcanzar nuevamente v.

Si el vértice actual v_i no tiene arco donde v_i sea destino, entonces v_i es fuente de dos arcos en C', pues en C tenía grado dos y ningún arco es de destino en C'. Entonces desde v_i partirían dos aristas monótonas greedy en G, y esto no es posible pues la dirección greedy es única desde cada punto.

Si se alcanzó nuevamente v, entonces podemos partir desde v y recorrer C' en el sentido corriente de la orientación (de fuente a destino) y llegar nuevamente a v. Entonces como cada arco es una arista monótona greedy en G, partiríamos desde v disminuyendo monótonamente la distancia a b hasta alcanzar el mismo v, y esto no puede ser posible.

Luego, toda orientación de atracción de las aristas de C conllevaría a una contradicción, por lo que no es posible que C exista. Luego, F_b es acíclica por lo que es un bosque. □

Dado que ya sabemos que F_b es un bosque, entonces particularmente nos interesa el árbol donde se encuentra el beacon b. La siguiente definición enuncia dicho árbol y el teorema a continuación establece la relación de este con el conjunto de vértices atraídos por b.

Definición 3.2.8. Sea b un beacon. El *árbol de atracción* T_b de b, es el árbol de F_b donde se encuentra b.

Teorema 3.2.9. *Sea b un beacon. Sea T_b el árbol de atracción de b y sea $V(T_b)$ el conjunto de vértices de T_b. Entonces $A_V(b) = V(T_b)$, es decir, el conjunto de vértices atraídos por b es el conjunto de vértices del árbol de atracción de b.*

Demostración. Mostremos primero que $V(T_b) \subseteq A_V(b)$:

El vértice b en T_b es trivialmente atraído por b, por lo que $b \in A_V(b)$. Sea un vértice $v \neq b$ en T_b y sea P el camino que une v y b en T_b. Dado que T_b es un árbol entonces P existe y es único. Como cada arista en P pertenece a T_b y por tanto a F_b, entonces todas pertenecen a E_b, por lo que son desplazamientos monótonos greedy respecto a b. Sea P' la digráfica con los vértices y aristas de P, orientando por atracción las aristas.

Partiendo desde b, recorramos tanto como sea posible los arcos de P' en sentido contrario a su orientación, es decir, de destino a fuente. El recorrido se hará mientras sea posible ir de destino a fuente y hasta alcanzar v.

Si el vértice actual v_i no tiene arco donde v_i sea destino, entonces separemos en otros dos casos. Si $v_i = b$ entonces existiría un arco cuya fuente sería b, y esto es imposible porque desde b no puede haber desplazamiento. Si $v_i \neq b$ entonces v_i sería fuente de dos arcos en P', pues en P tenía grado dos por ser nodo interno. Entonces desde v_i partirían dos aristas monótonas greedy en G, y esto no es posible pues la dirección greedy es única desde cada punto.

Si se alcanzó v, entonces podemos partir desde v y recorrer P' en el sentido corriente de la orientación (de fuente a destino) y llegar a b. Entonces como cada arco es una arista monótona greedy en G, partiendo desde v alcanzamos b, por lo que $v \in A_V(b)$.

Luego, para todo vértice $v \in V(T_b)$, entonces $v \in A_V(b)$, por lo que $V(T_b) \subseteq A_V(b)$.

Mostraremos ahora que $A_V(b)$ no contiene ningún otro vértice de G:

Supongamos que existe un vértice $v \in A_V(b)$ tal que $v \notin V(T_b)$. Entonces existiría un camino P en G desde v hasta b tal que todas sus aristas serían desplazamientos monótonos greedy respecto a b. Entonces todas las aristas de P pertenecerían a E_b, por lo que pertenecerían a F_b. Entonces como existiría

un camino en F_b que conecta v con b, ambos tendrían que estar en el mismo árbol de F_b, pero b está en $V(T_b)$ y asumimos que $v \notin V(T_b)$. Luego, llegamos a una contradicción por lo que $A_V(b)$ no puede contener ningún otro vértice de G.

Luego, como $V(T_b) \subseteq A_V(b)$ y $A_V(b)$ no puede contener ningún otro vértice de G, entonces $A_V(b) = V(T_b)$. □

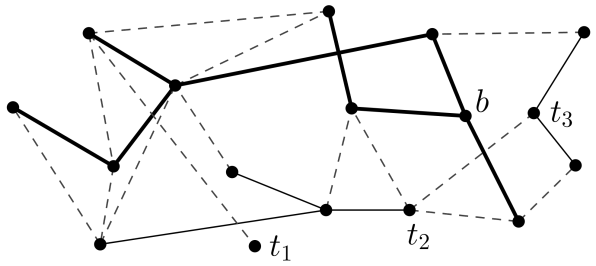


Figura 3.1: La figura contiene las gráficas G, F_b y T_b. La gráfica G está conformada por todos los vértices y todas las aristas (discontinuas, finas y gruesas). El bosque F_b está conformado por todos los vértices y las aristas no discontinuas. El árbol T_b está conformado por las aristas gruesas y los vértices incidentes en estas. Las aristas discontinuas no quedaron en F_b por no ser de atracción. El lector podrá notar que F_b contiene cuatro árboles, en los cuales la atracción conduce respectivamente a los vértices b, t_1, t_2 y t_3. También se puede notar que los vértices de T_b son los atraídos por b, es decir, que $A_V(b) = V(T_b)$.

El teorema anterior indica implícitamente una forma de determinar el conjunto de vértices atraídos por un beacon utilizando su árbol de atracción, pero necesitamos también obtener resultados en cuanto a las aristas atraídas para poder hallarlas. El siguiente teorema establece la relación entre las aristas atraídas y el conjunto de vértices atraídos.

Teorema 3.2.10. *Sea b un beacon y uv una arista. Entonces $uv \in A_E(b)$ $\iff$ uv alcanza v y $v \in A_V(b)$, o uv alcanza u y $u \in A_V(b)$.*

Demostración. Si uv alcanza v entonces todo punto en (uv) alcanza v. Entonces por Proposición 2.6.3 los puntos de (uv) correrán la misma suerte que v bajo la acción de b. Luego, si $v \in A_V(b)$ entonces uv es atraída por b, y si $v \notin A_V(b)$ entonces uv no es atraída por b.

Homólogamente, si uv alcanza u entonces tenemos que, si $u \in A_V(b)$ entonces uv es atraída por b, y si $u \notin A_V(b)$ entonces uv no es atraída por b.

Si uv no alcanza v ni u, entonces por 2.6.8 todos los puntos en (uv) alcanzarán un punto muerto en (uv), por lo que uv no es atraída por b. □

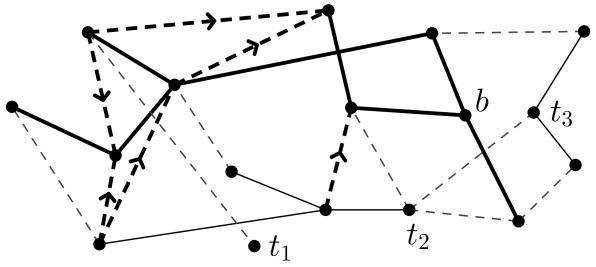
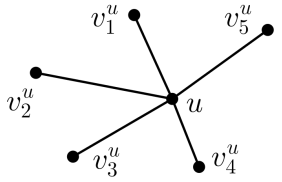


Figura 3.2: Las aristas de T_b serán atraídas por b por ser de atracción y siempre alcanzar un vértice de T_b. Las aristas discontinuas gruesas son las restantes aristas atraídas por b. El lector puede notar que algunas inciden en dos vértices de T_b mientras que otras solo inciden en uno. Las que inciden solo en uno tienen su orientación hacia el vértice de T_b.

El teorema anterior también indica implícitamente como obtener las aristas atraídas por un beacon, en este caso utilizando el conjunto de vértices atraídos por el beacon.

Un aspecto importante, y que hace la diferencia entre muchos algoritmos de esta tesis, es la ordenación circular de las listas de adyacencia de los vértices. Supongamos que para cada vértice u, tomamos como referencia el primer vértice v_1^u de su lista de adyacencia. Para todo vértice v_i^u adyacente a u, podemos obtener $\measuredangle v_1^u u v_i^u$, el ángulo formado por $v_i^u u$ partiendo de $v_1^u u$ girando en sentido contrario a las manecillas del reloj. Recordamos que en este caso $\measuredangle v_1^u u v_i^u$ sí puede ser mayor a 180 y que para el caso del propio v_1 decimos que $\measuredangle v_1^u u v_1^u = 0$. Enunciamos entonces la siguiente definición:

Definición 3.2.11. Sea u un vértice. Diremos que la lista de adyacencia de u está *ordenada circularmente* si, la secuencia de ángulos respecto a la primera arista es no decreciente, es decir si, $\forall\ v_i^u,\ v_j^u \in Adj[u]$ con $1 \leq i < j \leq d(u)$, se cumple que $\measuredangle v_1^u u v_i^u \leq \measuredangle v_1^u u v_j^u$.

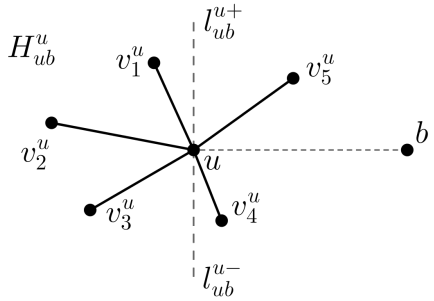


Nótese que en realidad la secuencia de ángulos será estrictamente creciente dado que se exige posición general. Muchos algoritmos descritos posteriormente asumen que los vértices de la gráfica tienen sus listas de adyacencia ordenadas circularmente. Para los análisis de correctitud y complejidad temporal de dichos algoritmos utilizaremos los siguientes teoremas y definiciones:

Teorema 3.2.12. *Sea u un vértice y b un beacon. Sea ub el segmento que une u y b sin que necesariamente ub sea una arista de la gráfica. Entonces el conjunto de aristas vu de u, que lo alcanzan bajo la acción de b, son aquellas tales que $v \in H^u_{ub}$.*

Demostración. Sea una arista vu de u. Entonces los segmentos vu y ub son consecutivos en u, por lo que aplicando la Proposición 2.2.8, como $v \in H^u_{ub}$ entonces $b \in H^u_{vu}$. Luego, como $b \in H^u_{vu}$ entonces por Corolario 2.6.10 se cumple que vu alcanza u. □

Definición 3.2.13. Sea u un vértice y b un beacon. Sea ub el segmento que une u y b sin que necesariamente ub sea una arista de la gráfica. Llamaremos l^{u+}_{ub} al rayo que parte desde u sobre l^u_{ub}, tal que H^u_{uv} se encuentra inmediatamente girando a partir de l^{u+}_{ub} en contra de las manecillas del reloj. Llamaremos l^{u-}_{ub} entonces al rayo opuesto que parte desde u sobre l^u_{ub}.



Teorema 3.2.14. *Sea b un beacon y u un vértice cuya lista de adyacencia $Adj[u]$ está ordenada circularmente. Entonces el conjunto de aristas de u que lo alcanzan bajo la acción de b, son consecutivas circularmente en $Adj[u]$.*

Demostración. Sea ub el segmento que une u y b sin que necesariamente ub sea una arista de la gráfica. Partiendo desde l^{u+}_{ub}, comencemos a girar en contra de las manecillas del reloj hasta alcanzar l^{u-}_{ub}.

Claramente, las aristas vu encontradas durante este recorrido son las que pertenecen a H^u_{ub} por definición de l^{u+}_{ub}. Entonces, estas son las que alcanzan u bajo la acción de b por Teorema 3.2.12.

Luego, todas son consecutivas circularmente en $Adj[u]$ por seguir también $Adj[u]$ un ordenamiento circular en contra de las manecillas del reloj. □

Teorema 3.2.15. *Sean $d_1, d_2, ..., d_V$, V números no negativos. Entonces se cumple que, la suma de los logaritmos de los V números es menor o igual que la suma de los logaritmos de su media aritmética, es decir:*

$$\sum_{i=1}^{V} \log d_i \leq \sum_{j=1}^{V} \log \frac{\sum_{i=1}^{V} d_i}{V}$$

Demostración. El miembro derecho de la desigualdad suma V veces el mismo valor, por lo que podemos cambiar la sumatoria por multiplicación, quedando:

$$\sum_{i=1}^{V} \log d_i \leq V \log \frac{\sum_{i=1}^{V} d_i}{V}$$

Por propiedades de logaritmos tenemos que:

$$\sum_{i=1}^{V} \log d_i = \log \prod_{i=1}^{V} d_i \quad \text{y} \quad V \log \frac{\sum_{i=1}^{V} d_i}{V} = \log \left(\frac{\sum_{i=1}^{V} d_i}{V} \right)^V$$

Luego debemos demostrar que:

$$\log \prod_{i=1}^{V} d_i \leq \log \left(\frac{\sum_{i=1}^{V} d_i}{V} \right)^V$$

La función logaritmo es monótona creciente, luego la desigualdad anterior se cumple si los argumentos de los logaritmos la cumplen, es decir, si:

$$\prod_{i=1}^{V} d_i \leq \left(\frac{\sum_{i=1}^{V} d_i}{V} \right)^V$$

Finalmente, pasando la potencia como raíz al miembro izquierdo, queda:

$$\sqrt[V]{\prod_{i=1}^{V} d_i} \leq \frac{\sum_{i=1}^{V} d_i}{V}$$

Para mejorar la comprensión de la última desigualdad, expandiremos la multiplicatoria y la sumatoria, quedando:

$$\sqrt[V]{d_1 \cdot d_2 \cdot ... \cdot d_V} \leq \frac{d_1 + d_2 + ... + d_V}{V}$$

Como el lector podrá notar en la desigualdad, el miembro izquierdo es la media geométrica y el miembro derecho es la media aritmética. Esta desigualdad entre ambas medias es conocida y está demostrada su veracidad cuando todos los valores son no negativos. Luego, como los V valores son no negativos, la desigualdad inicial se cumple. □

Respecto a la complejidad temporal de los algoritmos, también es importante mencionar que como la gráfica es conexa entonces $V = O(E)$, por lo que en los análisis de complejidad se afirmará que $O(V + E) = O(E)$.

Luego de tener estos resultados estamos en condiciones de enunciar algoritmos para hallar los conjuntos atraídos por un beacon. Los algoritmos presentados siguen básicamente la misma línea expuesta por los resultados anteriores: Dada G y un beacon b, primero se halla la subgráfica abarcadora de atracción F_b, donde implícitamente se encuentra el árbol de atracción T_b, luego se obtiene $A_V(b)$ a partir de un recorrido por T_b. Finalmente, para el caso de las aristas, a partir de $A_V(b)$ obtenemos $A_E(b)$.

3.3. Vértices Atraídos - 1

El algoritmo descrito en esta sección halla el conjunto de vértices atraídos $A_V(b)$ por un beacon b en $O(V + E) = O(E)$. El algoritmo sigue la línea expuesta por los resultados preliminares, hallando la subgráfica abarcadora de atracción F_b donde implícitamente se encuentra el árbol de atracción T_b, para luego obtener $A_V(b)$ a partir de un recorrido en profundidad por T_b.

Algoritmo 3.3.1 $O(V + E) = O(E)$

```
1: procedure VérticesAtraídos1(b, G)
2:     F_b ← (V(G), ∅)

3:     for all u ∈ V(G) do
4:         v ← BúsquedaLinealAristaGreedy(u, b, G)
5:         if v ≠ null and EsDesplazamientoMonotono(uv, b) then
6:             F_b.Adj[v] ← F_b.Adj[v] ∪ {u}
7:             F_b.Adj[u] ← F_b.Adj[u] ∪ {v}
8:         end if
9:     end for

10:    A_V = DFS-Vértices(b, F_b)
11:    return A_V.
12: end procedure
```

El método "BúsquedaLinealAristaGreedy(u, b, G)" recorre las aristas de u para encontrar la arista greedy. Si existe (una con ángulo menor estricto al beacon) entonces devuelve su otro vértice incidente. Si no existe entonces devuelve *null*. Su orden es entonces $O(d(u))$ por recorrer las aristas de u.

El método "EsDesplazamientoMonotono(uv, b)" devuelve "verdadero" si y solo si uv es desplazamiento monótono respecto a b, comprobando si b está en H^v_{uv}. Su orden es entonces $O(1)$.

El método "DFS-Vértices(b, F_b)" hace un recorrido en profundidad por el árbol de F_b que contiene a b, es decir, por T_b. Durante el recorrido llena una lista con todos los vértices visitados y al terminar devuelve dicha lista. Su orden es entonces $O(V)$ por ser T_b un árbol con a lo más $V(T_b) = V(G)$.

Teorema 3.3.2. *Sea b un beacon en una gráfica geométrica G. El Algoritmo 3.3.1 computa correctamente el conjunto de vértices atraídos por b en G.*

Demostración. La línea 1 del algoritmo inicializa una subgráfica abarcadora F_b de G con los vértices de G pero sin ninguna arista.

El ciclo siguiente detecta para cada vértice u de G si este tiene arista de atracción respecto a b: Primero busca la arista greedy respecto a b con el método "BúsquedaLinealAristaGreedy(u, b, G)", y en caso de existir además pregunta si esta es un desplazamiento monótono respecto a b utilizando el método "EsDesplazamientoMonotono(uv, b)".

En caso afirmativo entonces existe arista de atracción de u respecto a b, y en las líneas 6 y 7 esta se incluye en F_b. En caso negativo entones ninguna arista de u es de atracción respecto a b y podemos pasar a revisar las aristas de otro vértice.

Al concluir el ciclo entonces F_b contiene todos los vértices de G y las aristas de atracción respecto a b, por lo que por definición F_b es la subgráfica abarcadora de atracción de G. Luego F_b es un bosque por el Teorema 3.2.7, y contiene al árbol T_b donde se encuentra b por definición.

Finalmente el método "DFS-Vértices(b, F_b)" obtiene los vértices del árbol que contiene a b en F_b, es decir, obtiene $V(T_b)$, y por Teorema 3.2.9 $V(T_b) = A_V(b)$, por lo que obtenemos correctamente $A_V(b)$ para retornarlo. □

Teorema 3.3.3. *Sea b un beacon en una gráfica geométrica $G = (V, E)$. El Algoritmo 3.3.1 computa el conjunto de vértices atraídos por b en G en tiempo $O(V + E) = O(E)$.*

Demostración. La línea 1 del algoritmo inicializa una subgráfica abarcadora F_b de G con los vértices de G. Entonces su orden es $O(V)$.

El orden del recorrido "DFS-Vértices" de la línea 10 es $O(V)$, por ser T_b un árbol con a lo más $|V(T_b)| = |V(G)|$ vértices.

El ciclo de la línea 2 itera por todos los vértices de G y para cada uno invoca al método "BúsquedaLinealAristaGreedy" que itera por todas sus aristas, por lo que el orden amortizado es $O(V + E) = O(E)$. El resto de las instrucciones dentro del ciclo son $O(1)$ por lo que no afectan el orden.

Luego, el algoritmo es $O(V + E) = O(E)$. □

Como nota final de esta sección quisiéramos dejarle al lector una pregunta respecto al Algoritmo 3.3.1 ¿Es la línea 7 del algoritmo necesaria?

3.4. Vértices Atraídos - 2

El algoritmo descrito en esta sección, halla el conjunto de vértices atraídos $A_V(b)$ por un beacon b, asumiendo que los vértices de la gráfica tienen sus listas de adyacencia ordenadas circularmente. Su tiempo de ejecución es $O(V \log E/V)$, lo cual mejora sustancialmente el orden $O(V+E) = O(E)$ del Algoritmo 3.3.1 en la medida que la gráfica sea más densa. La mejora del algoritmo consiste en encontrar la arista greedy desde un vértice de forma más rápida, haciendo búsqueda binaria sobre las aristas del vértice. El resto del método es igual al Algoritmo 3.3.1.

Asumamos entonces que el vértice u tiene su lista de adyacencia ordenada circularmente, y que dado un beacon b queremos determinar la arista greedy desde u si existe. Sea $\measuredangle v_1^u ub$ el ángulo formado por bu partiendo de $v_1^u u$ girando en sentido contrario a las manecillas del reloj. Entonces podemos hacer búsqueda binaria de $\measuredangle v_1^u ub$ respecto a los ángulos de la lista de adyacencia de u, dado que esta está ordenada respecto a aquellos.

La búsqueda binaria nos arrojaría el vértice v_i^u tal que $\measuredangle v_1^u u v_i^u < \measuredangle v_1^u ub$ y $\measuredangle v_1^u u v_{i+1}^u > \measuredangle v_1^u ub$ si $i < d(u)$, es decir, tal que $\measuredangle v_1^u u v_i^u$ es el mayor de los menores. Luego las aristas candidatas a ser arista greedy serían uv_i^u y la próxima circularmente $uv^u_{(i \mod d(u))+1}$, por lo que podemos hacer la comprobación de menor ángulo estricto respecto a b solo sobre estas dos.

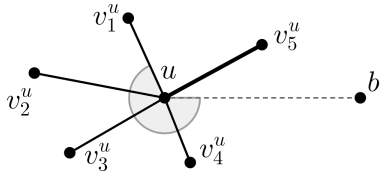


Figura 3.3: En la figura $\measuredangle v_1^u ub$ está resaltado. La búsqueda binaria nos arrojaría el vértice v_4^u por ser $\measuredangle v_1^u u v_4^u$ el mayor de los menores de $\measuredangle v_1^u ub$. Entonces se escogería la arista greedy entre $v_4^u u$ y $v_5^u u$. En este caso $\sphericalangle v_5^u ub < \sphericalangle v_4^u ub$, por lo que se escoge como arista de atracción $v_5^u u$ (arista gruesa).

Estableceremos que el método "BúsquedaBinariaAristaGreedy(u, b, G)" hace la búsqueda binaria descrita sobre las aristas de u para encontrar la arista greedy. Si existe (una con ángulo menor estricto al beacon) entonces devuelve su otro vértice incidente. Si no existe entonces devuelve *null*. Su orden es $O(\log d(u))$ por hacer búsqueda binaria sobre las aristas de u.

A continuación enunciamos entonces el segundo algoritmo para hallar el conjunto de vértices atraídos por un beacon. El lector notará que es prácticamente igual al Algoritmo 3.3.1, solo cambiando en la línea 4 el llamado a "BúsquedaLinealAristaGreedy" por "BúsquedaBinariaAristaGreedy". El algoritmo asume que las aristas de cada vértice están ordenadas circularmente.

Algoritmo 3.4.1 $O(V \log E/V)$ Asume ordenamiento circular

```
1: procedure VérticesAtraídos2(b, G)
2:     F_b ← (V(G), ∅)
3:     for all u ∈ V(G) do
4:         v ← BúsquedaBinariaAristaGreedy(u, b, G)
5:         if v ≠ null and EsDesplazamientoMonotono(uv, b) then
6:             F_b.Adj[v] ← F_b.Adj[v] ∪ {u}
7:             F_b.Adj[u] ← F_b.Adj[u] ∪ {v}
8:         end if
9:     end for
10:    A_V = DFS-Vértices(b, F_b)
11:    return A_V.
12: end procedure
```

Teorema 3.4.2. *Sea b un beacon en una gráfica geométrica G donde todos sus vértices tienen las listas de adyacencia ordenadas circularmente. El Algoritmo 3.4.1 computa correctamente el conjunto de vértices atraídos por b en G.*

Demostración. El método "BúsquedaBinariaAristaGreedy" difiere del método "BúsquedaLinealAristaGreedy" en su funcionamiento interno, pero ambos retornan la misma salida para una misma entrada, por lo que a los efectos del Algoritmo 3.4.1 son iguales.

Entonces podemos sustituir el método "BúsquedaBinariaAristaGreedy" por "BúsquedaLinealAristaGreedy" en el Algoritmo 3.4.1, quedando este igual al Algoritmo 3.3.1. Luego, por Teorema 3.3.2 el Algoritmo 3.3.1 es correcto, por lo que el Algoritmo 3.4.1 también lo es. □

Teorema 3.4.3. *Sea b un beacon en una gráfica geométrica $G = (V, E)$ donde todos sus vértices tienen las listas de adyacencia ordenadas circularmente. El Algoritmo 3.4.1 computa el conjunto de vértices atraídos por b en G en tiempo $O(V \log E/V)$.*

Demostración. La línea 1 del algoritmo inicializa una subgráfica abarcadora F_b de G con los vértices de G. Entonces su orden es $O(V)$.

El orden del recorrido "DFS-Vértices" de la línea 10 es $O(V)$, por ser T_b un árbol con a lo más $V(T_b) = V(G)$.

El ciclo de la línea 2 itera por todos los vértices u de G y para cada uno invoca al método "BúsquedaBinariaAristaGreedy" que es de orden $O(\log d(u))$. El resto de las instrucciones dentro del ciclo son $O(1)$ por lo que no afectan

el orden. Entonces el orden amortizado del ciclo es la sumatoria de los logaritmos de los grados de los vértices de G, es decir, $O(\sum_{i=1}^{V} \log d(u_i))$.

Dado que los grados de los vértices son no negativos, por Teorema 3.2.15 tenemos entonces que:

$$\sum_{i=1}^{V} \log d(u_i) \leq \sum_{j=1}^{V} \log \frac{\sum_{i=1}^{V} d(u_i)}{V}$$

Como el miembro derecho de la desigualdad suma V veces el mismo valor, podemos cambiar la sumatoria por multiplicación, quedando:

$$\sum_{i=1}^{V} \log d(u_i) \leq V \log \frac{\sum_{i=1}^{V} d(u_i)}{V}$$

Sustituyendo la suma de grados del miembro derecho por $2E$ tenemos:

$$\sum_{i=1}^{V} \log d(u_i) \leq V \log \frac{2E}{V}$$

Entonces, el orden amortizado del ciclo es $O(V \log E/V)$.

Luego, el orden del algoritmo es $O(V \log E/V)$. □

El peor caso del Algoritmo 3.4.1 es bajo la distribución equitativa de grados sobre los vértices $(2E/V)$. Para hacer una breve comparación entre los tiempos de ejecución del Algoritmo 3.4.1 y del Algoritmo 3.3.1, supongamos que $E = kV$ para un $0 < k \leq (V-1)/2$. Entonces el orden del Algoritmo 3.4.1 quedaría en $O(V \log kV/V) = O(V \log k)$, mientras que el orden del Algoritmo 3.3.1 quedaría en $O(V + kV) = O(Vk)$. Claramente, el Algoritmo 3.4.1 es mejor que el anterior y lo será más en la medida que la gráfica sea más densa.

Como nota final de esta sección y en respuesta de la pregunta propuesta en la sección anterior, afirmamos que las líneas 7 del Algoritmo 3.4.1 y del Algoritmo 3.3.1 no son necesarias. Basta insertar las aristas de atracción como arcos en sentido contrario a la atracción (líneas 6), ya que en el recorrido por el árbol de atracción se desciende, partiendo de b, desde vértices atraídos hacia vértices que alcanzan estos, por lo que basta poner las aristas en ese sentido.

3.5. Vértices y Aristas Atraídos - 1

El algoritmo descrito en esta sección halla el conjunto de vértices y aristas atraídos $A(b)$ por un beacon b en $O(V + E) = O(E)$. El algoritmo sigue la

línea expuesta por los resultados preliminares, hallando primero el conjunto de vértices atraídos $A_V(b)$, para a partir de estos obtener el conjunto de aristas atraídas $A_E(b)$ y finalmente devolver su unión.

Algoritmo 3.5.1 $O(V + E) = O(E)$

```
1: procedure VérticesAristasAtraídos1(b, G)
2:     A_V ← VérticesAtraídos1(b, G)
3:     A_E ← ∅

4:     for all u ∈ A_V do
5:         A_E ← A_E ∪ BusquedaLinealAristasIncidentesAlcanzan(u, b, G)
6:     end for

7:     A ← A_V ∪ A_E
8:     return A
9: end procedure
```

El método "BusquedaLinealAristasIncidentesAlcanzan(u, b, G)", devuelve un conjunto con todas las aristas incidentes a u que lo alcanzan bajo la acción de b. Para ello recorre todas las aristas vu de u y para cada una pregunta si $b \in H^u_{vu}$, es decir, si sus puntos alcanzan u bajo la acción de b. Solo en caso positivo incluye entonces la arista en el conjunto a devolver. Su orden es entonces $O(d(u))$ por recorrer todas las aristas de u.

Teorema 3.5.2. *Sea b un beacon en una gráfica geométrica G. El Algoritmo 3.5.1 computa correctamente el conjunto de vértices y aristas atraído por b en G.*

Demostración. La línea 1 del algoritmo obtiene $A_V = A_V(b)$, el conjunto de vértices atraídos por b. La línea 2 del algoritmo inicializa con vació el conjunto A_E.

El ciclo siguiente obtiene, para cada vértice atraído $u \in A_V = A_V(b)$, las aristas incidentes a u que lo alcanzan bajo la acción de b, almacenándolas en A_E. Entonces al concluir el ciclo, A_E almacena las aristas uv tal que, o uv alcanza $v \in A_V = A_V(b)$, o uv alcanza $u \in A_V = A_V(b)$. Luego, por Teorema 3.2.10, $A_E = A_E(b)$.

En la penúltima línea se crea el conjunto A uniendo los conjuntos $A_V = A_V(b)$ y $A_E = A_E(b)$. Finalmente se devuelve A que por definición es $A(b)$. □

Teorema 3.5.3. *Sea b un beacon en una gráfica geométrica $G = (V, E)$. El Algoritmo 3.5.1 computa el conjunto de vértices y aristas atraídos por b en G en tiempo $O(V + E) = O(E)$.*

Demostración. La línea 1 del algoritmo obtiene el conjunto de vértices atraídos por b, $A_V = A_V(b)$, utilizando el Algoritmo 3.3.1, por lo que queda en $O(V + E) = O(E)$.

La línea 7 es la unión de los conjuntos $A_V = A_V(b)$ y $A_E = A_E(b)$, cuyas cardinalidades cumplen que $|A_V| = |A_V(b)| \leq V$ y $|A_E| = |A_E(b)| \leq E$. Por lo que podemos acotar la unión por $O(V + E) = O(E)$.

El ciclo de la línea 2 itera por todos los vértices de $A_V = A_V(b)$, y para cada uno invoca al método "BusquedaLinealAristasIncidentesAlcanzan", que itera por todas sus aristas. Como $|A_V| = |A_V(b)|$ puede ser V, y se recorren todas las aristas de sus vértices, entonces se pueden recorrer todas las aristas de E. Entonces, el orden amortizado del ciclo es $O(V + E) = O(E)$.

Luego, el algoritmo es $O(V + E) = O(E)$. □

3.6. Vértices y Aristas Atraídos - 2

El algoritmo descrito en esta sección, halla el conjunto de vértices y aristas atraídos $A(b)$ por un beacon b, asumiendo que los vértices de la gráfica tienen sus listas de adyacencia ordenadas circularmente. Su tiempo de ejecución es $O((V \log E/V) + |A_E(b)|)$, el cual no mejora asintóticamente el orden $O(V + E) = O(E)$ del Algoritmo 3.5.1 pero sí pudiera llegar a ser mucho mejor en la práctica. La mejora del algoritmo consiste en que, para un vértice dado se encuentran las aristas de este que lo alcanzan de forma más rápida, haciendo dos búsquedas binarias sobre las aristas del vértice para luego solo recorrer las que lo alcanzan. El resto del método es igual al Algoritmo 3.5.1.

Asumamos entonces que el vértice u tiene su lista de adyacencia ordenada circularmente, y que dado un beacon b queremos obtener el conjunto de aristas de u que lo alcanzan bajo la acción de b:

Sea u^+ un punto en l_{ub}^{u+} distinto de u, y sea u^- un punto en l_{ub}^{u-} distinto de u. Sean entonces $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u u^-$ los ángulos formados por u^+u y u^-u respectivamente partiéndose de $v_1^u u$ girando en sentido contrario a las manecillas del reloj. Entonces podemos hacer búsquedas binarias de $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u u^-$ respecto a los ángulos de la lista de adyacencia de u.

Las búsquedas binarias nos arrojarían los vértices v_i^u y v_j^u respectivamente tales que $\measuredangle v_1^u u v_i^u$ y $\measuredangle v_1^u u v_j^u$ son los mayores de los menores de $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u u^-$ respectivamente. Luego, por Definición 3.2.13 de l_{ub}^{u+} y l_{ub}^{u-}, y por Teorema 3.2.14, las aristas de u que lo alcanzan bajo la acción de b son las que están circularmente partiendo desde v_{i+1}^u (próxima circular) hasta v_j^u. Recorriendo entonces la lista de adyacencia de u circularmente desde v_{i+1}^u hasta v_j^u, obtenemos dichas aristas.

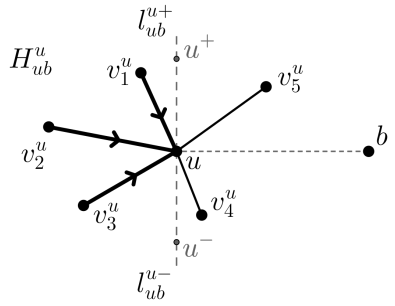


Figura 3.4: Las búsquedas binarias de $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u u^-$ nos arrojarían los vértices v_5^u y v_3^u, por ser $\measuredangle v_1^u u v_5^u$ el mayor de los menores de $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u v_3^u$ el mayor de los menores de $\measuredangle v_1^u u u^-$ respectivamente. El próximo vértice circularmente a v_5^u es v_1^u. Entonces todas las aristas desde v_1^u hasta llegar a v_3^u, en la lista de adyacencia de u, son las que alcanzan u bajo la acción de b. En la figura son las aristas gruesas.

Estableceremos que el método "BúsquedaMejorAristasIncidentesAlcanzan(u, b, G)" hace las dos búsquedas binarias descritas sobre las aristas de u, junto el recorrido descrito para devolver una lista con todas las aristas de u que lo alcanzan bajo la acción de b. Como caso especial, cuando el primer parámetro sea $u = b$, el método simplemente devolverá todas las aristas de b pues trivialmente todas lo alcanzan.

A continuación enunciamos entonces el segundo algoritmo para hallar el conjunto de vértices y aristas atraídos por un beacon. El lector notará que es prácticamente igual al Algoritmo 3.5.1, solo cambiando en la línea 1 el llamado a "VérticesAtraídos1" por "VérticesAtraídos2" y en la línea 5 el llamado a "BúsquedaLinealAristasIncidentesAlcanzan" por "BúsquedaMejorAristasIncidentesAlcanzan". El algoritmo asume que las aristas de cada vértice están ordenadas circularmente.

Algoritmo 3.6.1 $O((V \log E/V) + |A_E(b)|)$ Asume ordenamiento circular

```
1: procedure VérticesAristasAtraídos2(b, G)
2:     A_V ← VérticesAtraídos2(b, G)
3:     A_E ← ∅

4:     for all u ∈ A_V do
5:         A_E ← A_E ∪ BúsquedaMejorAristasIncidentesAlcanzan(u, b, G)
6:     end for

7:     A ← A_V ∪ A_E
8:     return A
9: end procedure
```

Teorema 3.6.2. *Sea b un beacon en una gráfica geométrica G donde todos sus vértices tienen las listas de adyacencia ordenadas circularmente. El Algoritmo 3.6.1 computa correctamente el conjunto de vértices y aristas atraído por b en G.*

Demostración. Los métodos "VérticesAtraídos2" y "BúsquedaMejorAristasIncidentesAlcanzan" difieren respectivamente de los métodos "VérticesAtraídos1" y "BúsquedaLinealAristasIncidentesAlcanzan" en sus funcionamientos internos, pero respectivamente retornan la misma salida para una misma entrada, por lo que a los efectos del Algoritmo 3.6.1 son iguales.

Entonces podemos sustituir los métodos "VérticesAtraídos2" y "BúsquedaMejorAristasIncidentesAlcanzan" por "VérticesAtraídos1" y "BúsquedaLinealAristasIncidentesAlcanzan" en el Algoritmo 3.6.1, quedando este igual al Algoritmo 3.5.1. Luego, por Teorema 3.5.2 el Algoritmo 3.5.1 es correcto, por lo que el Algoritmo 3.6.1 también lo es. □

Teorema 3.6.3. *Sea b un beacon en una gráfica geométrica $G = (V, E)$ donde todos sus vértices tienen las listas de adyacencia ordenadas circularmente. El Algoritmo 3.6.1 computa el conjunto de vértices y aristas atraídos por b en G en tiempo $O((V \log E/V) + |A_E(b)|)$.*

Demostración. La línea 1 del algoritmo obtiene el conjunto de vértices atraídos por b, $A_V = A_V(b)$, utilizando el Algoritmo 3.4.1, por lo que queda en $O(V \log E/V)$.

La línea 7 es la unión de los conjuntos $A_V = A_V(b)$ y $A_E = A_E(b)$, cuyas cardinalidades cumplen que $|A_V| = |A_V(b)| \leq V$ y $|A_E| = |A_E(b)| \leq E$, por lo que queda en $O(|A_V(b)| + |A_E(b)|)$.

El ciclo de la línea 2 itera por todos los vértices de $A_V = A_V(b)$, y para cada uno invoca al método "BúsquedaMejorAristasIncidentesAlcanzan". El método "BúsquedaMejorAristasIncidentesAlcanzan" hace dos procesos: un par de búsquedas binarias, y un recorrido por las aristas que alcanzan el vértice. Separaremos los dos procesos para hallar sus complejidades:

Para cada vértice en $A_V = A_V(b)$ el primer proceso hace dos búsquedas binarias sobre sus aristas. Repitiendo entonces el análisis hecho para hallar la complejidad del Algoritmo 3.4.1, nos queda que el proceso global toma de forma amortizada $O(V \log E/V)$, alcanzándose este bajo la distribución equitativa de grados sobre los vértices.

Para cada vértice en $A_V = A_V(b)$ el segundo proceso hace un recorrido por las arista de este que lo alcanzan. Por Teorema 3.2.10 sabemos que estas conforman $A_E(b)$, por lo que el proceso global toma de forma amortizada $O(|A_E(b)|)$.

Luego, el algoritmo es $O(V \log E/V) + |A_E(b)|)$. □

Capítulo 4

Kernels

4.1. Introducción

En este capítulo expondremos algoritmos para encontrar el kernel de beacons para vértices y el kernel de beacons para vértices y aristas. El kernel de beacons se define como el conjunto de vértices de la gráfica tal que cada uno por sí solo es capaz de atraer todos los elementos; para el caso de la variante de vértices esto significa atraer todos los vértices y para el caso de vértices y aristas significa atraer todos los vértices y aristas. En la próxima sección se definirán formalmente ambos conjuntos y se expondrán algunos resultados que servirán de apoyo para los análisis de los algoritmos. En las secciones posteriores expondremos los algoritmos para las dos variantes tratadas, mostrando la correctitud de estos y sus complejidades computacionales. Enfatizamos que en todas las definiciones y resultados mencionados estaremos haciendo referencia a una gráfica geométrica $G = (V, E)$.

4.2. Preliminares

Definición 4.2.1. El *kernel de beacons* $K_V(G)$, de una gráfica geométrica $G = (V, E)$ en la variante de vértices, es el conjunto de vértices (beacons) de G tal que cada uno es capaz de atraer todos los vértices de la gráfica, es decir, son los $v \in V$ tal que $A_V(v) = V$.

Definición 4.2.2. El *kernel de beacons* $K(G)$, de una gráfica geométrica $G = (V, E)$ en la variante de vértices y aristas, es el conjunto de vértices (beacons) de G tal que cada uno es capaz de atraer todos los vértices y aristas de la gráfica, es decir, son los $v \in V$ tal que $A(v) = V \cup E$.

Algunos de los algoritmos que se expondrán posteriormente preprocesan la gráfica ordenando circularmente las listas de adyacencia de todos los vértices. El siguiente teorema expone una cota superior de la complejidad temporal de este preproceso.

Teorema 4.2.3. *Sea $G = (V, E)$ una gráfica geométrica. La complejidad temporal de ordenar circularmente todas las listas de adyacencia de sus vértices es $O(V^2 \log E/V)$.*

Demostración. Utilizando cualquier algoritmo de ordenación de orden $O(n \log n)$, ordenar la lista de adyacencia de un vértice u sería $O(d(u) \log d(u))$. Luego, si este proceso se hace para todos los vértices de G, entonces el orden sería:

$$O(\sum_{i=1}^{V} d(u_i) \log d(u_i))$$

Sustituyamos en la fórmula interna el primer $d(u_i)$ por V, quedando $\sum_{i=1}^{V} V \log d(u_i)$. Como el grado de cada vértice es menor que la cantidad de vértices de la gráfica entonces $d(u_i) < V$, siendo menos estrictos digamos que $d(u_i) \leq V$. Entonces comparando las fórmulas tenemos que:

$$\sum_{i=1}^{V} d(u_i) \log d(u_i) \leq \sum_{i=1}^{V} V \log d(u_i)$$

Como V es una constante dentro de la sumatoria del miembro derecho entonces podemos extraerla, quedando:

$$\sum_{i=1}^{V} d(u_i) \log d(u_i) \leq V \sum_{i=1}^{V} \log d(u_i) \tag{1}$$

Respecto a la sumatoria del miembro derecho, como todos los grados son no negativos, entonces por Teorema 3.2.15 tenemos que:

$$\sum_{i=1}^{V} \log d(u_i) \leq \sum_{j=1}^{V} \log \frac{\sum_{i=1}^{V} d(u_i)}{V} \tag{2}$$

Sustituyendo miembro derecho de la ecuación (2) en el miembro derecho de la ecuación (1), queda:

$$\sum_{i=1}^{V} d(u_i) \log d(u_i) \leq V \sum_{j=1}^{V} \log \frac{\sum_{i=1}^{V} d(u_i)}{V}$$

Como el miembro derecho de la desigualdad suma V veces el mismo valor, podemos cambiar la sumatoria por multiplicación, quedando:

$$\sum_{i=1}^{V} d(u_i) \log d(u_i) \leq VV \log \frac{\sum_{i=1}^{V} d(u_i)}{V}$$

Sustituyendo la suma de grados del miembro derecho por $2E$ tenemos:

$$\sum_{i=1}^{V} d(u_i) \log d(u_i) \leq VV \log \frac{2E}{V}$$

Entonces nos queda:

$$\sum_{i=1}^{V} d(u_i) \log d(u_i) \leq V^2 \log \frac{2E}{V}$$

Luego, finalmente el orden sería $O(V^2 \log E/V)$. □

Aunque esta cota superior es buena, el lector podrá percatarse que se sustituyó implícitamente "$d(u_i) \log d(u_i)$" por "$V \log 2E/V$" para hacer la demostración. Esta sustitución no utiliza exactamente la distribución equitativa de grados sobre los vértices, pues al utilizarla quedaría "$(2E/V) \log 2E/V$". Esto nos hace pensar que la complejidad de la ordenación de todas las listas pudiera ser $O(V\ (E/V) \log E/V) = O(E \log E/V)$, lo cual mejoraría el $O(V^2 \log E/V)$ dado. Aun así, no tenemos demostración de lo anterior por lo que se deja como conjetura.

Conjetura 4.2.4. Sea $G = (V, E)$ una gráfica geométrica. Entonces la complejidad temporal de ordenar circularmente todas las listas de adyacencia de sus vértices es $O(E \log E/V)$.

Cada vértice que pertenezca al kernel de beacons $K(G)$ en la variante de vértices y aristas, debe poder atraer todos los vértices y aristas de la gráfica. Supongamos que un vértice b atrae a todos los vértices de la gráfica, es decir, que $b \in K_V(G)$. Si todas las aristas uv de la gráfica alcanzan alguno de sus extremos bajo la acción de b, entonces ocurrirá también que $b \in K(G)$.

Al asumir que $b \in K_V(G)$ entonces $A_V(b) = V(G)$. Entonces ya tenemos que b atrae todos los vértices de la gráfica, pero si además toda arista uv alcanza algún extremo, digamos v por ejemplo, entonces uv también es atraída por b, dado que alcanza v por lo que correrá la misma suerte que v bajo la acción de b y v es atraído por b. El siguiente teorema enuncia formalmente esta idea.

Teorema 4.2.5. *Sea $G = (V, E)$ una gráfica geométrica. Un vértice $b \in V$ pertenece a $K(G) \iff b \in K_V(G) \wedge \forall uv \in E$ uv alcanza uno de sus extremos bajo la acción de b.*

Demostración. Un vértice $b \in V$ pertenece a $K(G) \iff A(b) = V \cup E$ por definición. Entonces como $A(b) = A_V(b) \cup A_E(b)$ por definición y los conjuntos $A_V(b)$ y $A_E(b)$ son siempre disjuntos, es necesario y suficiente que $A_V(b) = V$ y $A_E(b) = E$.

Si $b \in K(G)$ entonces $A_V(b) = V$, por lo que directamente $b \in K_V(G)$. Además también se cumple que $A_E(b) = E$, por lo que todas las aristas son atraídas por b. Entonces todas las aristas necesitan alcanzar uno de sus vértices extremos bajo la acción de b, puesto que si no sería imposible que alcanzarán vértice alguno, incluido el propio b por lo que no serían atraídas por este.

Si $b \in K_V(G) \wedge \forall uv \in E$ uv alcanza uno de sus extremos bajo la acción de b, entonces ya sabemos directamente que $A_V(b) = V$. Además, como bajo la acción de b toda uv alcanza uno de sus extremos, digamos v, entonces uv es atraída por b puesto que: uv correrá la misma suerte que v bajo la acción de b por Proposición 2.6.3, y v es atraído por b por $A_V(b) = V$. Entonces como toda arista es atraída por b, $A_E(b) = E$. Luego, $A(b) = A_V(b) \cup A_E(b) = V \cup E$, por lo que $b \in K(G)$ por definición. □

4.3. Kernel en Vértices

El algoritmo expuesto en esta sección halla el kernel de beacons $K_V(G)$ en la variante de vértices. El algoritmo comienza preprocesando la gráfica ordenando circularmente las listas de adyacencia de todos los vértices. Luego se itera por todos los vértices de la gráfica y para cada uno se aplica el Algoritmo 3.4.1, comprobando después si el conjunto de vértices atraídos contiene todos los vértices de la gráfica. Su orden es $O(V^2 \log E/V)$.

En caso de que las listas de adyacencia ya estuviesen ordenadas circularmente entonces el preproceso inicial no sería necesario, pero aun así el orden quedaría igualmente $O(V^2 \log E/V)$. Otra opción posible sería inicialmente comprobar si están ordenadas circularmente o no, y ordenarlas después si no lo están. Esta comprobación se haría pasando por todos los vértices y revisando las listas de adyacencia verificando si los ángulos no decrecen (crecen) respecto a la primera arista. La comprobación solo llevaría $O(E)$ así que no afectaría el orden del algoritmo. Por simplicidad preferimos dejar el preproceso sin hacer la comprobación dado que el orden asintótico no se afecta al hacerlo.

Algoritmo 4.3.1 $O(V^2 \log E/V)$

```
1: procedure KernelVértices(G)
2:     for all u ∈ V(G) do
3:         OrdenarCircularmenteAristas(u, G)
4:     end for

5:     K_V ← ∅

6:     for all b ∈ V(G) do
7:         A_V ← VérticesAtraídos2(b, G)
8:         if |A_V| = |V(G)| then
9:             K_V ← K_V ∪ {b}
10:        end if
11:    end for

12:    return K_V.
13: end procedure
```

Teorema 4.3.2. *Sea $G = (V, E)$ una gráfica geométrica. El Algoritmo 4.3.1 computa correctamente $K_V(G)$, el kernel de beacons de G en la variante vértices.*

Demostración. Las líneas 2-4 del algoritmo ordenan circularmente las listas de adyacencia de todos los vértices de G para que después se pueda utilizar el Algoritmo 3.4.1 Vértices Atraídos 2. La línea 5 inicializa el conjunto K_V con el vacío.

El ciclo de la línea 6 itera por todos los vértices b de G, y para cada uno en la línea 7 halla su conjunto de vértices atraídos $A_V(b)$ utilizando el Algoritmo 3.4.1. La línea 8 comprueba si la cardinalidad de $A_V(b)$ es igual a la cardinalidad de V. Solo en caso positivo se añade el vértice b a K_V en la línea 9.

Al terminar el ciclo entonces K_V contendrá los vértices $v \in V$ tal que $|A_V(v)| = |V|$, es decir, tal que $A_V(v) = V$. Luego por Definición 4.2.1 $K_V = K_V(G)$, por lo que se retorna correctamente $K_V(G)$. □

Teorema 4.3.3. *Sea $G = (V, E)$ una gráfica geométrica. El Algoritmo 4.3.1 computa $K_V(G)$, el kernel de beacons de G en la variante vértices, en orden $O(V^2 \log E/V)$.*

Demostración. Las líneas 2-4 del algoritmo ordenan circularmente las listas de adyacencia de todos los vértices de G. Entonces por Teorema 4.2.3, esto

quedaría en $O(V^2 \log E/V)$. En caso de estar ordenadas esto no sería necesario, bastaría una comprobación previa al ordenamiento que quedaría en orden $O(E)$.

El ciclo de la línea 6 itera por todos los vértices de G, y para cada uno halla su conjunto de vértices atraídos utilizando el Algoritmo 3.4.1, el cual es $O(V \log E/V)$. Entonces el orden del ciclo es $O(V^2 \log E/V)$.

El resto de las líneas del algoritmo son $O(1)$ por lo que no afectan la complejidad temporal. Luego, el orden del algoritmo es $O(V^2 \log E/V)$. □

Un algoritmo aun más simple para hallar el kernel en esta variante sería utilizando el Algoritmo 3.3.1 Vértices Atraídos 1, el cual no necesita que las aristas estén ordenadas circularmente. En este caso las líneas 2-4 de ordenamiento no serían necesarias, y en la línea 7 utilizamos entonces el Algoritmo 3.3.1 en vez del Algoritmo 3.4.1. El orden de este algoritmo quedaría entonces en $O(VE)$, puesto que se ejecutaría el Algoritmo 3.3.1 que es $O(E)$ desde cada vértice de V.

Evidentemente este algoritmo sería peor asintóticamente que el Algoritmo 4.3.1, puesto que $O(VE)$ es peor que $O(V^2 \log E/V)$. Si por ejemplo nuestra gráfica fuera densa el orden de este algoritmo sería $O(V^3)$, mientras que el del Algoritmo 4.3.1 sería solo $O(V^2 \log V)$. En caso de ser dispersa ambos algoritmos se comportarían igual.

No se incluyó este algoritmo en el trabajo por su simpleza y por ser superado asintóticamente por otro algoritmo también simple.

4.4. Kernel en Vértices y Aristas - 1

El algoritmo expuesto en esta sección halla el kernel de beacons $K(G)$ en la variante de vértices y aristas. El algoritmo es simple: se itera por todos los vértices de la gráfica y para cada uno se aplica el Algoritmo 3.5.1 Atracción de Vértices y Aristas 1, comprobando después si el conjunto de vértices y aristas atraídos contiene todos los vértices y aristas de la gráfica. Su orden es $O(VE)$.

En caso de que las listas de adyacencia ya estuviesen ordenadas circularmente entonces valdría la pena utilizar el Algoritmo 3.6.1 Atracción de Aristas 2, mejorándose la eficiencia en la práctica aunque quedando igual asintóticamente en $O(VE)$. En este caso el preproceso de ordenamiento no sería tan ventajoso, pues el beneficio de utilizar el Algoritmo 3.6.1 pudiera ser contrarrestado precisamente por el costo de la ordenación, dependería mucho de la estructura y la geometría de la gráfica, así como de los tamaños de los conjuntos de aristas atraídos desde cada vértice. Como no ofrecía mejora en el

orden asintótico y tampoco mejora segura en la eficiencia práctica, decidimos no incluir el preproceso de ordenamiento de las aristas en el algoritmo.

Algoritmo 4.4.1 $O(VE)$

```
1: procedure KernelVérticesAristas1(G)
2:     K ← ∅

3:     for all b ∈ V(G) do
4:         A ← VérticesAristasAtraídos1(b, G)
5:         if |A| = |V(G)| + |E(G)| then
6:             K ← K ∪ {b}
7:         end if
8:     end for

9:     return K
10: end procedure
```

Teorema 4.4.2. *Sea $G = (V, E)$ una gráfica geométrica. El Algoritmo 4.4.1 computa correctamente $K(G)$, el kernel de beacons de G en la variante vértices y aristas.*

Demostración. La línea 2 inicializa el conjunto K con el vacío.

El ciclo de la línea 3 itera por todos los vértices b de G, y para cada uno en la línea 4 halla su conjunto de vértices y aristas atraídos $A(b)$ utilizando el Algoritmo 3.5.1. La línea 5 comprueba si la cardinalidad de $A(b)$ es igual a la cardinalidad de V más la cardinalidad de E. Solo en caso positivo se añade el vértice b a K en la línea 6.

Al terminar el ciclo entonces K contendrá los vértices $v \in V$ tal que $|A(v)| = |V| + |E|$, es decir, tal que $A(v) = V \cup E$. Luego por Definición 4.2.2 $K = K(G)$, por lo que se retorna correctamente $K(G)$. □

Teorema 4.4.3. *Sea $G = (V, E)$ una gráfica geométrica. El Algoritmo 4.4.1 computa $K(G)$, el kernel de beacons de G en la variante vértices y aristas, en orden $O(VE)$.*

Demostración. El ciclo de la línea 3 itera por todos los vértices de G, y para cada uno halla su conjunto de vértices y aristas atraídos utilizando el Algoritmo 3.5.1, el cual es $O(E)$. Entonces el orden del ciclo es $O(VE)$.

El resto de las líneas del algoritmo son $O(1)$ por lo que no afectan la complejidad temporal. Luego, el orden del algoritmo es $O(VE)$. □

El rendimiento de este algoritmo es malo, basta asumir que la gráfica es densa para ver que el algoritmo puede terminar en $O(V^3)$. Aun así se puede caer en la errónea conclusión de que este algoritmo no puede ser mejorado asintóticamente, dado que para poder incluir un vértice candidato en el kernel debemos tener seguridad de que atrae toda la gráfica, por lo que suena razonable que tengamos que ejecutar el algoritmo de vértices y aristas atraídos para comprobar si efectivamente el candidato atrajo o no toda la gráfica. El punto aquí es que no se necesita realmente saber quienes son los vértices y aristas atraídos por el candidato, basta conocer la cantidad que atrae para determinar si pertenece o no al kernel.

4.5. Kernel en Vértices y Aristas - 2

El algoritmo expuesto en esta sección halla el kernel de beacons $K(G)$ en la variante de vértices y aristas en orden $O(V^2 \log E/V)$. Como se ve, esto es una mejora considerable al $O(VE)$ del algoritmo anterior para el mismo problema. El algoritmo comienza hallando el kernel de vértices $K_V(G)$, dado que estos son los únicos candidatos de G a ser parte de $K(G)$. Luego, para cada uno de estos verifica que el total de aristas atraídas sean todas las de la gráfica. La mejora de este algoritmo consiste en que no es necesario conocer cuáles son las aristas atraídas por un candidato, basta solo saber el total de estas por lo que se evita tener que iterar por las aristas atraídas.

Para hallar el total de aristas atraídas por un candidato b se utiliza una técnica parecida a la mostrada en el Algoritmo 3.6.1, utilizando las búsquedas binarias de l_{ub}^{u+} y l_{ub}^{u-} sobre cada vértice u. Con estas búsquedas obtendríamos los límites de las aristas que alcanzan u en $Adj[u]$, por lo que calculando la diferencia circular tendríamos la cantidad de estas que alcanzan u. Luego, como b atrae todos los vértices, podemos afirmar que esta cantidad de aristas de u son atraídas por b. Finalmente comparamos el total de aristas atraídas por b con $|E|$, y en caso de ser iguales entonces incluimos b en $K(G)$.

Asumamos entonces que el vértice u tiene su lista de adyacencia ordenada circularmente, y que dado un beacon b queremos obtener la cantidad de aristas de u que alcanzan u bajo la acción de b:

Sea u^+ un punto en l_{ub}^{u+} distinto de u, y sea u^- un punto en l_{ub}^{u-} distinto de u. Sean entonces $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u u^-$ los ángulos formados por u^+u y u^-u respectivamente, partiéndose de $v_1^u u$ girando en sentido contrario a las manecillas del reloj. Entonces podemos hacer búsquedas binarias de $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u u^-$ respecto a los ángulos de la lista de adyacencia de u.

Las búsquedas binarias nos arrojarían los vértices v_i^u y v_j^u respectivamente tales que $\measuredangle v_1^u u v_i^u$ y $\measuredangle v_1^u u v_j^u$ son los mayores de los menores de $\measuredangle v_1^u u u^+$ y

$\measuredangle v_1^u u u^-$ respectivamente. Luego, por Definición 3.2.13 de l_{ub}^{u+} y l_{ub}^{u-}, y por Teorema 3.2.14, las aristas de u que lo alcanzan bajo la acción de b son las que están circularmente partiendo desde v_{i+1}^u (próxima circular) hasta v_j^u. La fórmula entonces para calcular la cantidad de estas sería:

$$alcanzan_u = \begin{cases} j-(i+1)+1 = j-i & si \quad i \leq j \\ \\ d(u)-(i+1)+1+j = d(u)-i+j & si \quad i > j \end{cases}$$

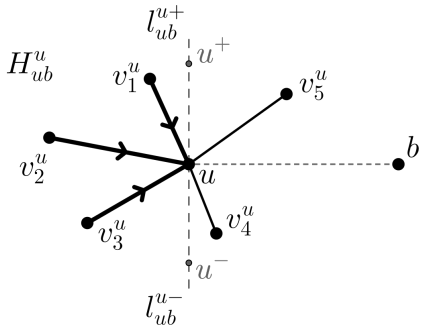


Figura 4.1: Las búsquedas binarias de $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u u^-$ nos arrojarían los vértices v_5^u y v_3^u, por ser $\measuredangle v_1^u u v_5^u$ el mayor de los menores de $\measuredangle v_1^u u u^+$ y $\measuredangle v_1^u u v_3^u$ el mayor de los menores de $\measuredangle v_1^u u u^-$ respectivamente. El próximo vértice circularmente a v_5^u es v_1^u. Entonces todas las aristas desde v_1^u hasta llegar a v_3^u, en la lista de adyacencia de u, son las que alcanzan u bajo la acción de b. En la figura son las aristas gruesas. Para calcular la cantidad de estas basta hacer el cálculo indicado por la fórmula: $alcanzan_u = d(u) - 5 + 3 = 5 - 5 + 3 = 3$.

Estableceremos que el método "CantidadAristasIncidentesAlcanzan(u, b, G)" hace las dos búsquedas binarias descritas sobre las aristas de u, y después utiliza la fórmula anterior para calcular la cantidad de aristas de u que lo alcanzan bajo la acción de b. El orden del método es $O(\log d(u))$ por las dos búsquedas binarias sobre las aristas de u. Como caso especial, cuando el primer parámetro sea $u = b$, el método simplemente devolverá $d(b)$ pues trivialmente todas lo alcanzan.

A continuación enunciamos entonces el segundo algoritmo para hallar el kernel de beacons para vértices y aristas. El algoritmo no asume que las aristas de cada vértice están ordenadas circularmente, dado que cuando se invoca a "KernelVértices" este internamente las ordena circularmente.

Algoritmo 4.5.1 $O(V^2 \log E/V)$

```
1: procedure KernelVérticesAristas2(G)
2:     K_V ← KernelVértices(G)
3:     K ← ∅

4:     for all b ∈ K_V do
5:         total_b ← 0
6:         for all u ∈ V(G) do
7:             alcanzan_u ← CantidadAristasIncidentesAlcanzan(u, b, G)
8:             total_b ← total_b + alcanzan_u
9:         end for
10:        if total_b = |E(G)| then
11:            K ← K ∪ {b}
12:        end if
13:    end for

14:    return K
15: end procedure
```

Teorema 4.5.2. *Sea $G = (V, E)$ una gráfica geométrica. El Algoritmo 4.5.1 computa correctamente $K(G)$, el kernel de beacons de G en la variante vértices y aristas.*

Demostración. La línea 2 obtiene $K_V = K_V(G)$ el kernel de beacons para vértices de G. Internamente este ordena circularmente las aristas de los vértices de G por lo que eventualmente podemos utilizar "CantidadAristasIncidentesAlcanzan".

La línea 3 inicializa el conjunto K con el vacío.

El ciclo de la línea 4 itera por todos los vértices candidatos b de $K_V = K_V(G)$. Para cada uno comienza inicializando la variable numérica $total_b$ en cero, indicando que el candidato b todavía no ha provocado que ninguna arista alcance un extremo.

El ciclo de la línea 6 entonces itera por todos los vértices u de la gráfica. Entonces con el método "CantidadAristasIncidentesAlcanzan(u, b, G)" se calcula la cantidad de aristas de u que lo alcanzan bajo la acción de b y se almacena en la variable $alcanzan_u$. Estas aristas vu, que alcanzan u bajo la acción de b, no serán contadas como que alcanzan sus $v's$ cuando se ejecute "CantidadAristasIncidentesAlcanzan(v, b, G)" sobre sus $v's$, pues por Corolario 2.6.8 una sola de ambas cosas puede ocurrir. Luego, estas aristas bajo la acción de b alcanzan su extremo u y solo su extremo u, por lo que

podemos agregarlas al total de aristas que alcanzan extremo $total_b$ sin hacer omisiones ni repeticiones.

Al concluir el ciclo de la línea 6 entonces la variable $total_b$ contiene el total de aristas de G que alcanzan algún extremo bajo la acción de b. Si este total es igual a $|E|$ entonces $\forall uv \in E$ uv alcanza alguno de sus extremos bajo la acción de b, y como además tenemos que $b \in K_V(G)$, entonces por Teorema 4.2.5 nos queda que $b \in K(G)$, por lo que en la línea 11 agregamos el candidato b al conjunto K.

Al terminar el ciclo de la línea 4 entonces K contendrá los vértices $b \in K(G)$ tal que $b \in K_V(G)$, pero como el Teorema 4.2.5 exige también que necesariamente todo $v \in K(G)$ cumpla que $v \in K_V(G)$, entonces $K = K(G)$.

Luego, el algoritmo devuelve correctamente $K = K(G)$. □

Teorema 4.5.3. *Sea $G = (V, E)$ una gráfica geométrica. El Algoritmo 4.4.1 computa $K(G)$, el kernel de beacons de G en la variante vértices y aristas, en orden $O(V^2 \log E/V)$.*

Demostración. La línea 2 obtiene $K_V = K_V(G)$ el kernel de beacons para vértices de G utilizando el Algoritmo 4.3.1, que es $O(V^2 \log E/V)$.

El ciclo de la línea 4 itera por todos los vértices candidatos b de $K_V = K_V(G)$. El ciclo de la línea 6 itera entonces por cada vértice u de G y hace dos búsquedas binarias sobre sus aristas, las cuales son $O(\log d(u))$. Repitiendo entonces el análisis hecho para hallar la complejidad del Algoritmo 3.4.1, nos queda que el proceso global del ciclo de la línea 6 toma de forma amortizada $O(V \log E/V)$, alcanzándose este bajo la distribución equitativa de grados sobre los vértices. Luego, como esto ocurre para cada vértice candidato b en $K_V = K_V(G)$, que potencialmente pueden ser todos los de G, entonces todo el ciclo de la línea 4 queda en $O(VV \log E/V) = O(V^2 \log E/V)$.

El resto de las líneas del algoritmo son $O(1)$ por lo que no afectan la complejidad temporal. Luego, el orden del algoritmo es $O(V^2 \log E/V)$. □

Es un poco sorprendente que las complejidades del kernel para vértices y del kernel para vértices y aristas, queden igual en $O(V^2 \log E/V)$, y sin tan siquiera tomar en cuenta la densidad de la gráfica. Aun así no tenemos sospecha de que exista un mejor algoritmo asintótico para el kernel de vértices, puesto que, lo que provoca que ambos algoritmos tengan el mismo orden es precisamente la restricción de que en el kernel de vértices y aristas todas las aristas tengan que ser atraídas, lo que nos evita enumerarlas bastando solo conocer el total de estas.

Por otra parte, dado el orden $O(V^2 \log E/V)$ de este algoritmo, se infiere directamente que a medida que la gráfica sea más densa entonces el algoritmo necesitará más tiempo. De hecho, podemos pensar que el problema en sí,

hallar el kernel para vértices y aristas, es en general más difícil de solucionar a medida que la gráfica se hace más densa. La siguiente sección muestra una técnica que contradice esta idea, llegándose incluso a concluir que el problema en una gráfica completa es más fácil asintóticamente. La técnica se basa en reducir candidatos al kernel de vértices y aristas, y se logra con un proceso que solo toma $O(V + E) = O(E)$, lo cual la hace muy llamativa para incorporarla a los algoritmos mostrados.

4.6. Kernel en Vértices y Aristas - Técnica

En esta sección no discutiremos un algoritmo para hallar kernel de vértices y aristas, sino una técnica de reducción de candidatos que puede ser aplicada a los algoritmos de este problema. La técnica se logra recorriendo los vértices y aristas de la gráfica un número constante de veces, por lo que su orden es $O(V + E) = O(E)$. Aunque puede ser muy efectiva, esta no reduce la complejidad temporal de los algoritmos en los que puede ser usada. La efectividad de la técnica está en dependencia de la densidad de la gráfica, su geometría y la distribución de las aristas. Como se dijo inicialmente, la técnica intenta reducir candidatos al kernel.

Supongamos que tomamos un vértice u y queremos evaluar la posibilidad de que sus vértices adyacentes sean parte del kernel de vértices y aristas. Sea un vértice v adyacente a u, claramente para que $v \in K(G)$, v no puede estar en la banda B_{wz} de ninguna arista wz de la gráfica, puesto que si no entonces existiría un punto muerto en wz respecto a v y todos sus puntos lo alcanzarían. En particular, v no puede estar en la banda de ninguna arista de u por la misma razón. Enunciaremos rápidamente estos resultados.

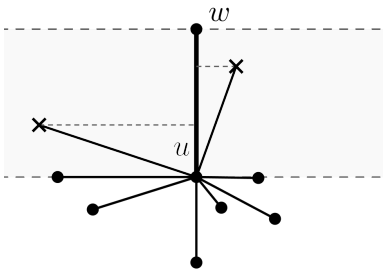
Teorema 4.6.1. *Sea v un vértice en una gráfica geométrica $G = (V, E)$. Si $v \in K(G) \implies \forall\ wz \in E$ se cumple que $v \notin B_{wz}$.*

Demostración. Supongamos que $v \in K(G)$ y que $\exists\ wz \in E$ tal que $v \in B_{wz}$. Entonces como $v \in B_{wz}$ por Corolario 2.6.8 existe un punto muerto $d \in wz$ y todos los puntos de wz lo alcanzarán, lo que implica que todos permanecerán estacionarios en d por Corolario 2.6.4. Luego, wz no sería atraída por v por lo que v no pudiera estar en $K(G)$. Esto contradice la hipótesis por lo que no puede existir tal arista wz y el teorema se cumple. □

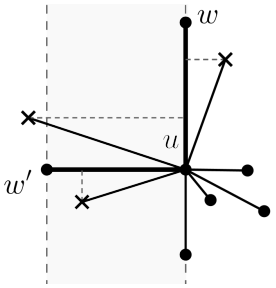
Corolario 4.6.2. *Sea u un vértice en una gráfica geométrica $G = (V, E)$. Si un vértice $v \in Adj[u]$ es tal que $v \in K(G)$, entonces $\forall\ w \in Adj[u]$ se cumple que $v \notin B_{uw}$.*

Demostración. Si $v \in K(G)$ y existiera un $w \in Adj[u]$ tal que $v \in B_{uw}$, entonces existiría una arista $uw \in E$ tal que $v \in B_{uw}$, y por el teorema anterior esto no puede ocurrir. □

Teniendo estos resultados entonces supongamos que tomamos el vértice $w \in Adj[u]$, tal que la distancia entre u y w es la mayor entre u y sus adyacentes. Es decir, tomamos $w \in Adj[u]$ tal que $\forall\ v \in Adj[u]$ se cumple que $d(w, u) \geq d(v, u)$. Sin perdida de generalidad y por sencillez asumamos que w está arriba de u en la misma vertical. Entonces todo vértice v adyacente a u que esté encima de la horizontal donde está u, no podrá pertenecer al kernel, puesto que necesariamente está en la banda de uw dado que no puede estar más arriba que w. De esta forma podemos eliminar como candidatos todos los vecinos de u que estén por encima de su horizontal, quedando en este semiplano solo un candidato, w.

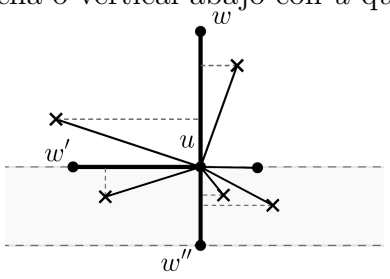


Supongamos ahora que pasamos los vértices de $Adj[u]$ que no fueron eliminados, sin incluir w, a una nueva lista $Adj[u]'$, y repetimos el proceso sobre esta. Tomamos $w' \in Adj[u]'$ tal que $\forall\ v \in Adj[u]'$ se cumple que $d(w', u) \geq d(v, u)$, y después eliminemos todos los que están en $B_{uw'}$. El peor caso es que w' se encuentre sobre la horizontal de u, digamos a la izquierda sin perder generalidad, puesto que se estarían eliminando candidatos de un solo nuevo cuadrante de u, el inferior izquierdo.

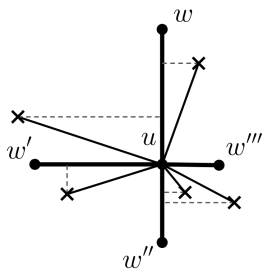


Igualmente, si pasamos ahora los vértices sobrevivientes de $Adj[u]'$ sin w' a $Adj[u]''$, podemos repetir el proceso, el cual nos dará un nuevo w''. En este caso, lo peor sería que w'' estuviese sobre la vertical de u abajo o sobre la horizontal con u a la derecha, lo cual eliminaría todos los candidatos del

cuadrante inferior derecho pero dejaría espacio a un sobreviviente más (w''') sobre la horizontal derecha o vertical abajo con u que no ha sido utilizada.



Al terminar este proceso entonces podemos percatarnos que los únicos sobrevivientes de $Adj[u]$ son, w, w', w'' y w''', es decir, solo fueron cuatro, puesto que los demás candidatos estarían en alguno de los cuadrantes que estos eliminaron.



Más aun, si tomamos en cuenta que se pide posición general entonces los sobrevivientes serían solo tres. El siguiente teorema enuncia que los sobrevivientes $w's$ a este proceso solo pueden ser cuatro sin tomar en cuenta la posición general.

Teorema 4.6.3. *Sea u un vértice en una gráfica geométrica G. La eliminación de candidatos al $K(G)$ anteriormente descrita, aplicada sobre los vértices adyacentes al vértice u, dejará a lo más cuatro de estos como candidatos.*

Demostración. Supongamos que $W = [w_1, w_2, ..., w_k]$, es la lista ordenada circularmente de candidatos sobrevivientes al proceso de eliminación descrito sobre u. Nótese que al estar ordenada circularmente entonces no se está exigiendo que $d(w_i, u) \geq d(w_{i+1}, u)\ \forall 1 \leq i \leq |W|$.

Para cada w_i, tenemos que circularmente $\measuredangle w_i u w_{i+1} \geq 90$, porque si ocurriese que $\measuredangle w_i u w_{i+1} < 90$ entonces w_i y w_{i+1} estarían cada uno en el semiplano de eliminación del otro, y esto no es posible porque uno de los dos se escogió primero en el proceso, por lo que su distancia a u es mayor o igual que la del otro, quedando el otro eliminado por lo que no pertenecería a W.

Entonces para cada arista $w_i u$ existe una región abierta de 90 grados a su izquierda donde no puede haber ningún w_j, puesto que $\measuredangle w_i u w_{i+1} \geq 90$ y w_{i+1} es el próximo circularmente a la izquierda. Luego $|W| \leq 4$, dado que si no entonces la región angular alrededor de u sería mayor que 360 grados. $\square$

A continuación presentamos el algoritmo "CandidatosEliminados" que recibirá la gráfica como entrada. Como salida devolverá una lista de candidatos eliminados por la aplicación del proceso de eliminación realizado sobre todos los vértices de la gráfica. Ya que la descripción del proceso fue informal, el algoritmo será dado más en detalle.

Algoritmo 4.6.4 $O(E)$

```
1: procedure CandidatosEliminados(G)
2:     Eliminados ← ∅

3:     for all u ∈ V(G) do
4:         Enlazada ← Adj[u].ListaEnlazada()

5:         while Enlazada ≠ ∅ do              ▷ cuatro iteraciones a lo más.
6:             w ← Enlazada.Primero
7:             for all v ∈ Enlazada do          ▷ halla w, el más lejano a u.
8:                 if  d(v,u) > d(w,u) then
9:                     w ← v
10:                end if
11:            end for

12:            for all v ∈ Enlazada do      ▷ remueve y reporta los v ∈ B_uw.
13:                if v ∈ B_uw then
14:                    Enlazada.Remueve(v)
15:                    Eliminados ← Eliminados ∪ {v}
16:                end if
17:            end for
18:            Enlazada.Remueve(w)   ▷ remueve w para próxima iteración.
19:        end while

20:    end for

21:    return Eliminados
22: end procedure
```

Nótese que en el método expuesto se usa una lista enlazada para manejar los vértices candidatos en cada ronda de eliminación. Esto se hace para poder eliminar en $O(1)$ de la lista, aunque en el método hacemos abstracción de esto utilizando directamente el valor de los nodos de la lista enlazada, y no los nodos como en realidad se necesitaría para hacer la eliminación en $O(1)$. El siguiente teorema establece la complejidad temporal del método.

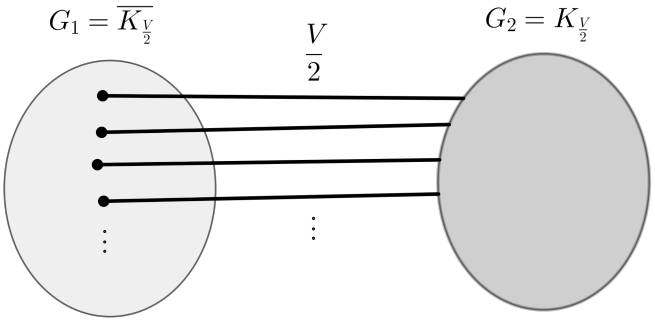
Teorema 4.6.5. *Sea $G = (V, E)$ una gráfica geométrica. El Algoritmo 4.6.4 computa la lista de candidatos eliminados, aplicando el proceso de eliminación realizado sobre todos los vértices de la gráfica, en $O(E)$.*

Demostración. El ciclo de la línea 3 itera por todos los vértices u de la gráfica Para cada u el ciclo de la línea 5 hace el proceso de eliminación descrito sobre sus adyacentes.

En cada paso de este ciclo, se escoge el vértice w que está a mayor distancia de u y se eliminan los que están en B_{uw}, ambos procesos son lineales en cuanto a $Adj[u]$ por utilizarse una lista enlazada para eliminar en $O(1)$. En la línea 18 del ciclo de la línea 5, se elimina también w de la lista, por lo que en la próxima iteración este no existirá y se tendrá que escoger otro vértice que este a mayor distancia de u. Entonces por Teorema 4.6.3 solo se pueden escoger cuatro $w's$, por lo que este ciclo solo puede hacer a lo más cuatro iteraciones.

Luego, el ciclo de la línea 3 itera por todos los vértices u de la gráfica, para cada u el ciclo de la línea 3 hace a lo más 4 iteraciones, y dentro de este se hacen $O(d(u))$ operaciones. Entonces amortizadamente el orden queda $O(V + E) = O(E)$. □

Este método entonces se pudiera insertar al inicio de los algoritmos de kernel de vértices y aristas, para disminuir los candidatos a pertenecer al kernel y hacer menos computo después. Como la complejidad de los algoritmos de kernel de vértices y aristas es mucho mayor que $O(E)$, entonces la inserción del método no afectaría el orden y tampoco mucho la eficiencia en la práctica. Los beneficios de aplicar esta técnica para reducir candidatos pueden ser muy buenos, pero esta no mejora la complejidad temporal de los algoritmos para kernel de vértices y aristas. La siguiente familia de gráficas geométricas muestra que la técnica puede solo eliminar una fracción lineal de los vértices como candidatos y aun así haber una fracción lineal de las aristas posibles.



Las subgráficas G_1 y G_2 tienen $V/2$ vértices. Están comunicadas por $V/2$ aristas disjuntas de G_1 a G_2. La gráfica G_1 no tiene ninguna otra arista y G_2 es una completa. Entonces G es la unión de ambas gráficas con las $V/2$ aristas disjuntas. Como G_1 y G_2 están bien alejados, los vértices de G_1 sobrevivirán siempre la eliminación de candidatos. Esto nos deja con al menos $V/2$ candidatos, los de G_1, y con $\frac{V}{2}(\frac{V}{2}-1)/2+\frac{V}{2}$ aristas, las de G_2 y las $V/2$ disjuntas.

Aun así podemos notar que en algunos tipos de gráficas la técnica sí reduce el orden de los algoritmos. Si la gráfica es completa entonces el proceso de eliminación desde un solo vértice u dejaría solo cinco candidatos contando u, por lo que bastaría ejecutar el algoritmo de atracción desde estos cinco para determinar si alguno de ellos está en el kernel. Luego, para gráficas completas solo se necesitaría hacer cinco veces $O(V \log E/V)$, por lo que el orden quedaría en $O(V \log E/V)$.

Capítulo 5

Cobertura

5.1. Introducción

En este capítulo trabajaremos sobre el problema de cobertura para las dos variantes. El problema de cobertura está basado en encontrar un conjunto de vértices (beacons), de cardinalidad mínima, tal que todo vértice o todo vértice y arista, sea atraído por al menos uno del conjunto. Respecto a Geometría Computacional, la cobertura por beacons puede ser vista como una extensión de la cobertura por visibilidad, donde los beacons dominan o cubren puntos más allá de la frontera de lo visible. Respecto a Teoría de Gráficas, la cobertura por beacons puede ser vista como una extensión de los problemas de dominación y cobertura de gráficas, donde se dominan o cubren vértices o aristas más allá de la vecindad de un vértice.

En la próxima sección se definirá el problema de manera formal para ambas variantes. Además, se definirá también el problema *Set Cover*, el cual será utilizado para mostrar la dificultad del problema en ambas variantes. En las secciones posteriores se tratará la cobertura en ambas variantes, exponiéndose también algunas cotas de beacons necesarios y suficientes para cubrir las gráficas.

5.2. Preliminares

Definición 5.2.1. Sea G una gráfica geométrica. El problema de optimización *Cobertura por Beacons* de G en la variante de vértices, se refiere a: Encontrar un conjunto de vértices (beacons) de cardinalidad mínima, tal que todo vértice de la gráfica es atraído por al menos uno de ellos.

Definición 5.2.2. Sea G una gráfica geométrica. El problema de optimización *Cobertura por Beacons* de G en la variante de vértices y aristas, se refiere

a: Encontrar un conjunto de vértices (beacons) de cardinalidad mínima, tal que todo vértice y arista de la gráfica es atraído por al menos uno de ellos.

Como referencia para el lector enunciamos a continuación el problema *Set Cover* en su versión de optimización.

Definición 5.2.3. Sea U un conjunto de n elementos llamado universo, y S una colección de m subconjuntos de U cuya unión es U. El problema de optimización *Set Cover*, se refiere a: Encontrar un subconjunto de S de cardinalidad mínima, cuya unión sea U.

Los problemas mencionados son todos de optimización, requiriéndose siempre que la cardinalidad del conjunto solución sea mínima. A continuación enunciamos los mismos problemas en su versión de problemas de decisión.

Definición 5.2.4. Sea G una gráfica geométrica y k un entero positivo. El problema de decisión *Cobertura por Beacons* de G en la variante de vértices, se refiere a: Decidir si existe un conjunto de vértices (beacons) de cardinalidad menor o igual a k, tal que todo vértice de la gráfica es atraído por al menos uno de ellos.

Definición 5.2.5. Sea G una gráfica geométrica y k un entero positivo. El problema de decisión *Cobertura por Beacons* de G en la variante de vértices y aristas, se refiere a: Decidir si existe un conjunto de vértices (beacons) de cardinalidad menor o igual a k, tal que todo vértice y arista de la gráfica es atraído por al menos uno de ellos.

Para demostrar la NP-completitud de estos dos problemas utilizaremos el problema *Set Cover* en su versión de decisión, la cual enunciamos a continuación.

Definición 5.2.6. Sea U un conjunto de n elementos llamado universo, S una colección de m subconjuntos de U cuya unión es U, y k un entero positivo. El problema de decisión *Set Cover*, se refiere a: Decidir si existe un subconjunto de S de cardinalidad menor o igual a k, cuya unión sea U.

El problema *Set Cover* es bien conocido en la literatura y fue uno de los problemas incluidos en la lista de los 21 problemas tratados por Karp en [8] en 1972. En su versión de optimización el problema es NP-difícil (*NP-hard*) y en su versión de decisión es NP-completo (*NP-complete*).

A continuación mostraremos que los problemas de decisión de cobertura por beacons pertenecen a la clase NP.

Teorema 5.2.7. *Sea G una gráfica geométrica. El problema de decisión Cobertura por Beacons de G en la variante de vértices, pertenece a la clase de problemas NP.*

Demostración. Supongamos que para una gráfica geométrica G y un entero positivo k, tenemos como certificado un conjunto B de beacons. Entonces podemos comprobar en tiempo polinomial si $|B| \leq k$ y si cada vértice de G es atraído por algún beacon de B. Para el segundo paso ejecutaríamos alguno de los algoritmos polinomiales de vértices atraídos desde cada beacon de B, y marcaríamos todos los vértices de G atraídos. Al final haríamos una pasada por todos los vértices de G comprobando si todos quedaron marcados.

Luego, en tiempo polinomial podemos verificar el certificado por lo que el problema pertenece a NP. □

Teorema 5.2.8. *Sea G una gráfica geométrica. El problema de decisión Cobertura por Beacons de G en la variante de vértices y aristas, pertenece a la clase de problemas NP.*

Demostración. Supongamos que para una gráfica geométrica G y un entero positivo k, tenemos como certificado un conjunto B de beacons. Entonces podemos comprobar en tiempo polinomial si $|B| \leq k$ y si cada vértice y arista de G es atraído por algún beacon de B. Para el segundo paso ejecutaríamos alguno de los algoritmos polinomiales de vértices y aristas atraídos desde cada beacon de B, y marcaríamos todos los vértices y aristas de G atraídos. Al final haríamos una pasada por todos los vértices y aristas de G comprobando si todos quedaron marcados.

Luego, en tiempo polinomial podemos verificar el certificado por lo que el problema pertenece a NP. □

Para hacer más contextual el lenguaje respecto al término *cobertura*, si un beacon atrae un vértice diremos que lo *cubre* y si un vértice es atraído diremos que este está *cubierto*. En general, el término cobertura y el término atracción junto con sus respectivas derivaciones serán equivalentes.

5.3. Complejidad de Cobertura en Vértices

En esta sección probaremos que el problema de cobertura para vértices en su versión de decisión es NP-completo y en su versión de optimización es NP-difícil. Para la reducción inicial utilizaremos el problema *Set Cover*.

Teorema 5.3.1. *Sea G una gráfica geométrica. El problema de decisión Cobertura por Beacons de G en la variante de vértices es NP-difícil.*

Demostración. Supongamos que tenemos una instancia del problema de decisión *Set Cover*, es decir, tenemos U un conjunto de n elementos llamado universo, S una colección de m subconjuntos de U cuya unión es U, y un entero $k > 0$, y queremos decidir si existe un subconjunto de S de cardinalidad menor o igual a k, cuya unión sea U. Si U es vació la respuesta es trivialmente verdadera para todo S y todo k. Igualmente la respuesta es trivialmente verdadera si $k > m$, pues la unión de S es U. Asumiremos entonces que U no es vació, por lo que S tampoco puede serlo, y que $k \leq m$.

Crearemos entonces una gráfica geométrica G, colocando vértices a partir de U y S, junto con otros vértices extras, y agregando algunas aristas.

Colocación de Vértices

A cada elemento $e_i \in U$, $0 \leq i \leq n-1$, le asignaremos un vértice en G que llamaremos también e_i y que pertenece a un conjunto de vértices igualmente llamado U. Los vértices e_i serán colocados en una línea vertical a partir del origen de coordenadas. Luego, el vértice e_i quedará en $(0; i)$, estando el primero en $(0; 0)$ y el último en $(0; n-1)$.

A cada subconjunto $s_j \in S$, $0 \leq j \leq m-1$, le asignaremos un vértice en G que llamaremos también s_j y que pertenece a un conjunto de vértices igualmente llamado S. Los vértices s_j serán colocados en una línea vertical partiendo de $(c; 0)$ para un $c > 0$, a intervalos $\Delta = (n-1)/(m-1)$. Luego, el vértice s_j quedará en $(c; j\Delta)$, estando el primero en $(c; 0)$ y el último en $(c; n-1)$. Sean p_u y p_s los puntos medios de los segmentos que parten respectivamente desde $(0; 0)$ y $(c; 0)$ hasta respectivamente $(0; n-1)$ y $(c; n-1)$, la elección de c estará condicionada entonces a que $\sphericalangle s_0 p_u s_{m-1} < 90$, lo cual equivale a decir que $\sphericalangle e_0 p_s e_{n-1} < 90$.

Esta ubicación sobre la recta $x = c$ implica que: $\sphericalangle s_j e_i s_l < 90$ $\forall$ terna de vértices $s_j, s_l \in S$ y $e_i \in U$, y que análogamente $\sphericalangle e_i s_j e_l < 90$ $\forall$ terna de vértices $e_i, e_l \in U$ y $s_j \in S$.

Ambos conjuntos están entonces de frente verticalmente, dentro de la misma franja horizontal comprendida entre las rectas $y = 0$ y $y = n-1$, y a una distancia c que asegura que los ángulos entre dos cualesquiera de un conjunto y uno cualquiera del otro, sean menores de 90.

También colocaremos un vértice b en G cuya coordenada y será $(n-1)/2$. Su coordenada x será negativa, y lo suficientemente lejana de los e_i, tal que $\sphericalangle s_0 b s_{m-1} < 90$, $\sphericalangle s_0 e_{n-1} b \geq 90$ y $\sphericalangle s_0 e_0 b \geq 90$. Las últimas dos condiciones equivalen respectivamente a que $\sphericalangle s_{m-1} e_0 b \geq 90$ y $\sphericalangle s_{m-1} e_{n-1} b \geq 90$. Esto se puede calcular en $O(1)$ dado que conocemos la posición de s_0, s_{m-1}, e_0, e_{n-1} junto con la y de b.

Esta ubicación de b implica que $\sphericalangle s_j e_i b \geq 90$ $\forall$ par de vértices $s_j \in S$ y $e_i \in U$. Entonces para todo par igual, $\sphericalangle e_i s_j b < 90$, puesto que si no entonces

la suma de los ángulos interiores de $\triangle s_j e_i b$ sería mayor que 180.

Por último colocaremos un conjunto $F = f_0, f_1, ..., f_{n+m-1}$ de $n + m$ vértices, muy cerca de b. Estos vértices tendrán sus coordenadas $x's$ negativas pero mayores que la de b, por lo que estarán entre la x de b y la recta $x = 0$. Su coordenada y puede ser cualquiera tal que estos queden en el área definida por $\sphericalangle s_0 b s_{m-1}$, es decir, entre los segmentos bs_0 y bs_{m-1}. De esta forma siempre se cumple que: $\sphericalangle f_f b f_g < 90\ \forall$ par de vértices $f_f, f_g \in F$, dado que estos siempre están dentro del área definida por $\sphericalangle s_0 b s_{m-1} < 90$; y $\sphericalangle s_j b f_f < 90\ \forall$ par de vértices $s_j \in S$ y $f_f \in F$, por la misma razón anterior.

Para los tríos de vértices que quedan en línea recta, podemos siempre hacer una pequeña perturbación para lograr la posición general.

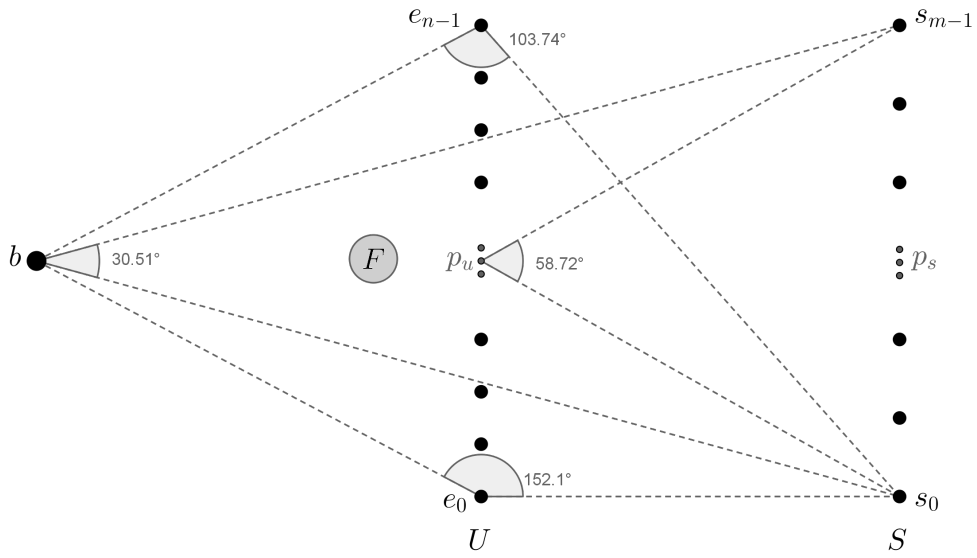


Figura 5.1: En la figura se muestra la colocación de los vértices con los ángulos exigidos. Nótese que: $\sphericalangle s_0 p_u s_{m-1} < 90$, $\sphericalangle s_0 b s_{m-1} < 90$, $\sphericalangle s_0 e_{n-1} b \geq 90$ y $\sphericalangle s_0 e_0 b \geq 90$.

Colocación de Aristas

Para las aristas de G, crearemos una arista entre un vértice e_i y un vértice s_j, si y solo si el subconjunto s_j contiene al elemento e_i. Desde cada s_j insertaremos una arista a b, y también desde cada f_f insertaremos una arista a b. Estas son las únicas aristas que tendrá G.

Como se puede notar, G es una gráfica geométrica conexa, puesto que desde todo vértice existe un camino a b, por lo que entre todo par de vértices existe un camino: Cada e_i tiene que tener arista hacia algún s_j ya que la unión de los subconjuntos s_j es igual a U, desde cada s_j hay arista a b, y

desde cada f_f hay arista a b. La figura siguiente muestra como quedaría la gráfica de forma genérica.

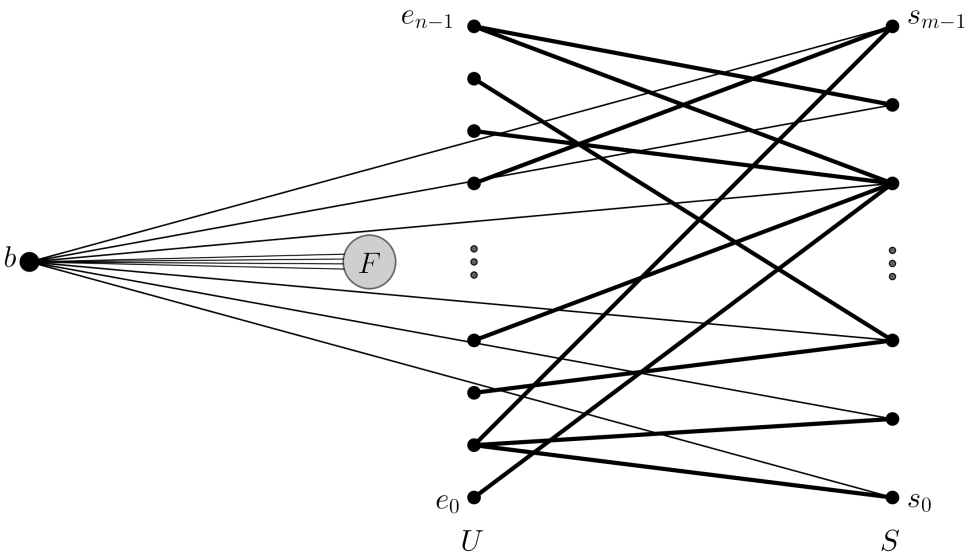


Figura 5.2: En la figura se muestra un ejemplo de la reducción. Las aristas gruesas unen los e_i con los s_j, y se colocan si y solo si $e_i \in s_j$ en *Set Cover*.

Conjuntos Atraídos

Un vértice $f_f \in F$ puede atraer a b aparte de sí mismo. Por construcción $\sphericalangle f_f b f_g < 90\ \forall f_f, f_g \in F$, por lo que f_f no puede atraer ningún otro vértice de F. Además, como $\sphericalangle s_j b f_f < 90\ \forall$ par de vértices $s_j \in S$ y $f_f \in F$, f_f no puede atraer ningún vértice en S, y como no puede atraer ninguno de S entonces tampoco puede atraer ninguno de U.

El vértice b puede atraer a todos los de F y a todos los de S por tener aristas con todos ellos. El vértice b no puede atraer ningún vértice $e_i \in U$, pues los e_i solo están conectados con los $s_j \in S$, y por construcción vimos que $\sphericalangle e_i s_j b < 90$.

Un vértice $s_j \in S$ puede atraer a b y a todos los $e_i \in U$ con los que tenga arista. Un vértice $s_j \in S$ no puede atraer ningún $f_f \in F$ por $\sphericalangle s_j b f_f < 90\ \forall$ par de vértices $s_j \in S$ y $f_f \in F$. Tampoco un s_j puede atraer ningún otro vértice $s_l \in S$, puesto que solo sería a través de algún vértice $e_i \in U$, y por construcción vimos que $\sphericalangle s_j e_i s_l < 90$. Luego, un s_j no podría atraer tampoco a un e_i con el que no tuviera arista, pues e_i si acaso alcanzaría algún $s_l \in S$ con el que esté conectado, y vimos que s_j no puede atraer ningún otro $s_l \in S$.

Un vértice $e_i \in U$ puede atraer a todos los $s_j \in S$ con los que tenga arista. Un vértice $e_i \in U$ no puede atraer ningún otro vértice $e_l \in U$, puesto que solo sería a través de algún vértice $s_j \in S$ y por construcción vimos que

$\sphericalangle e_i s_j e_l < 90$. Luego, un e_i no podría atraer tampoco a un s_j con el que no tuviera arista, pues el s_j si acaso alcanzaría algún $e_l \in U$ con el que esté conectado, y vimos que e_i no puede atraer ningún otro $e_l \in U$. Además, e_i tampoco pudiera atraer a b, pues los e_i solo están conectados con los $s_j \in S$, y por construcción vimos que $\sphericalangle e_i s_j b < 90$. Luego, tampoco e_i pudiera atraer a los $f_f \in F$, pues si acaso estos alcanzarían b, y vimos que b no es atraído por e_i.

Resumiendo, los conjuntos de vértices atraídos por cada vértice de G son los siguientes:

- $A_V(f_f) = \{f_f, b\}$, $\forall$ vértice $f_f \in F$.
- $A_V(b) = \{b\} \cup F \cup S$.
- $A_V(s_j) = \{s_j, b\} \cup \{e_i : e_i s_j \in G\}$, $\forall$ vértice $s_j \in S$.
- $A_V(e_i) = \{e_i\} \cup \{s_j : s_j e_i \in G\}$, $\forall$ vértice $e_i \in U$.

Algoritmo Polinomial para Cobertura por Beacons

La gráfica geométrica G obtenida a partir del problema *Set Cover* tiene $2(n+m)+1$ vértices: los n de U, los m de S, los $n+m$ de F y b. Recordemos que asumimos que U no era vació, por lo que S tampoco lo es, lo que provoca que los respectivos conjuntos de vértices U y S de la gráfica tampoco sean vacíos. Recordemos también que asumimos que $k \leq m$, por lo que se cumple que $k \leq m < n + m = |F|$, es decir $k < |F|$.

Supongamos entonces que tenemos un algoritmo polinomial para el problema de decisión *Cobertura por Beacons* en la variante de vértices, y le pasamos como parámetros la gráfica geométrica G y $k + 1$. El algoritmo entonces devolvería "verdadero" si y solo si existe un conjunto de vértices (beacons) B de cardinalidad menor igual a $k + 1$, tal que todo vértice en G es atraído por el menos uno de ellos. Mostraremos entonces como serían elegidos los beacons en G.

El vértice b siempre será escogido para estar en B: Cada vértice en F solo puede ser atraído por él mismo y por b, y estos no pueden atraer a ningún otro en G. Como el conjunto U es no vació, entonces se necesita al menos un beacon que esté o en U o en S para atraerlo, luego, nos quedarían a lo más k posibles beacons para atraer b y F. Entonces como $k < |F|$, escogiendo b es la única forma de atraer a todos los de F, y no queda ningún otro vértice afectado por no escoger uno de F.

Al ser escogido necesariamente b como beacon, entonces no es necesario escoger ningún vértice de F como beacon. Estos solo pudieran ser elegidos por exceso si con menos de k se cubrió $U \cup S$.

Al ser escogido necesariamente b como beacon, entonces también todos los vértices de S quedan atraídos, por lo que desaparece el incentivo de atraerlos. Entonces quedan solo por atraer los vértices de U, y se pueden utilizar a lo más k beacons, por lo que el algoritmo respondería "verdadero" si y solo si es posible atraerlos a todos con una cantidad menor o igual a k.

Interpretando la Salida

Supongamos que el algoritmo responde "falso". Entonces no existiría ningún subconjunto $B \in U \cup S$ con $|B| \leq k$, tal que cada vértice de U quede cubierto por al menos uno de B. Entonces evidentemente tampoco puede pasar que exista algún $B \in S$ con $|B| \leq k$, tal que cada vértice de U quede cubierto por al menos uno de B.

Supongamos que el algoritmo responde "verdadero". Entonces existiría un subconjunto $B \in U \cup S$ con $|B| \leq k$, tal que cada vértice de U queda cubierto por al menos uno de B. Como cada $e_i \in U$ solo puede atraer a algunos de S y a sí mismo, y ya los de S están atraídos por b, entonces las únicas razones por las que un e_i pertenecería a B serían porque e_i se está cubriendo a sí mismo y nadie más lo está cubriendo, o por exceso. Pero como cada e_i tiene que estar conectado con al menos un $s_j \in S$, y s_j lo puede atraer, entonces podemos transformar B intercambiando e_i por s_j, quedando B con la misma cardinalidad o menor y sin contener e_i. Entonces este razonamiento lo podemos repetir hasta que no quede ningún $e_i \in U$ en B. Luego, si el algoritmo responde "verdadero", entonces existiría un subconjunto $B \in S$ con $|B| \leq k$, tal que cada vértice de U queda cubierto por al menos uno de B.

En cualquiera de ambos casos, podemos interpretar la salida del algoritmo respecto a si existe o no un subconjunto $B \in S$ con $|B| \leq k$, tal que cada vértice de U quede cubierto por al menos uno de B. Además, tenemos que cada $s_j \in S$ solo podía atraer un $e_i \in U$ si y solo si existía arista entre ellos en G, y cada arista $s_j e_i$ fue colocada en G si y solo si el elemento e_i pertenecía al subconjunto s_j en *Set Cover*.

Entonces podemos directamente interpretar la salida del algoritmo respecto al problema de decisión *Set Cover*. Si la respuesta del algoritmo con G y $k+1$ fue "falso" para el problema de decisión *Cobertura por Beacons*, entonces para el problema de decisión *Set Cover* no existe un subconjunto de S de cardinalidad menor o igual a k, cuya unión sea U. Si la respuesta del algoritmo con G y $k+1$ fue "verdadero" para el problema de decisión *Cobertura por Beacons*, entonces para el problema de decisión *Set Cover* existe un subconjunto de S de cardinalidad menor o igual a k, cuya unión es U.

Complejidad

Luego, si existiera tal algoritmo polinomial para el problema de decisión *Cobertura por Beacons* en la variante de vértices, tendríamos un algoritmo polinomial para el problema de decisión *Set Cover*, puesto que:

La transformación inicial descrita de *Set Cover* a la gráfica geométrica G es polinomial, el algoritmo para el problema de decisión *Cobertura por Beacons* de G en la variante de vértices sería polinomial, y la interpretación de la salida de este para *Set Cover* es trivialmente polinomial.

Luego, como el problema de decisión *Set Cover* es NP-difícil, entonces el problema de decisión *Cobertura por Beacons* en la variante de vértices es NP-difícil. □

Teniendo que el problema de decisión es NP y también NP-difícil, enunciamos entonces inmediatamente la NP-completitud de este.

Teorema 5.3.2. *Sea G una gráfica geométrica. El problema de decisión Cobertura por Beacons de G en la variante de vértices es NP-completo.*

Demostración. Por Teorema 5.2.7 tenemos que el problema es NP. Por Teorema 5.3.1 tenemos que el problema es NP-difícil. Luego, el problema es NP-completo. □

Finalmente, partiendo de que el problema de decisión es NP-difícil, enunciaremos entonces que el problema de optimización es también NP-difícil.

Teorema 5.3.3. *Sea G una gráfica geométrica. El problema de optimización Cobertura por Beacons de G en la variante de vértices es NP-difícil.*

Demostración. Supongamos que tenemos una instancia del problema de decisión con G y un entero $k > 0$ como parámetros. Supongamos entonces que tenemos un algoritmo polinomial para el problema de optimización. Entonces ejecutaríamos dicho algoritmo sobre G, obteniendo como salida un conjunto B de cardinalidad mínima, tal que cada vértice de G es atraído por al menos un $b \in B$.

Si $|B| > k$, entonces es imposible que exista un conjunto de cardinalidad menor o igual a k tal que cada vértice sea atraído, pues B tenía cardinalidad mínima. Entonces, devolveríamos "falso" para el problema de decisión.

Si $|B| \leq k$, entonces existe un conjunto de cardinalidad menor o igual a k tal que cada vértice es atraído, el propio B. Entonces, devolveríamos "verdadero" para el problema de decisión.

Luego, si existiera dicho algoritmo polinomial para el problema de optimización, tendríamos un algoritmo polinomial para el problema de decisión, puesto que: la transformación del problema de decisión al de optimización es

trivialmente polinomial, el algoritmo de optimización sería polinomial, y la transformación de la salida de este al de decisión es trivialmente polinomial.

Luego, como el problema de decisión *Cobertura por Beacons* en la variante de vértices es NP-difícil, entonces el problema de optimización *Cobertura por Beacons* en la variante de vértices es NP-difícil. □

5.4. Combinatoria de Cobertura en Vértices

En esta sección expondremos cotas de beacons necesarios y suficientes para cubrir gráficas geométricas con n vértices en la variante de vértices. Para exponer la cota de necesidad mostraremos una familia infinita de gráficas que necesitan siempre una cierta cantidad para ser cubiertas. Para la cota de suficiencia utilizaremos una bicoloración de un árbol abarcador de la gráfica.

Teorema 5.4.1. *Existe una familia infinita de gráficas geométricas G, tal que si G tiene n vértices entonces se necesitan $\lfloor \frac{n}{2} \rfloor$ beacons para cubrir todos sus vértices.*

Demostración. Una "rueda" (*wheel*) W_k, es una gráfica resultado de la operación *join* entre un ciclo $C_k = v_1v_2...v_kv_1$ y un vértice c, de manera tal que entre todo vértice de C_k y c existe una arista. El nombre proviene precisamente de que esta puede ser dibujada como una rueda, donde los vértices de C_k están en el exterior de manera circular y c está en el centro, con las aristas del ciclo bordeando el círculo y las aristas a c dirigidas al centro como radios (*spokes*). Imaginemos entonces que tenemos la rueda W_k dibujada de esta manera pero manteniendo que las aristas son segmentos rectos.

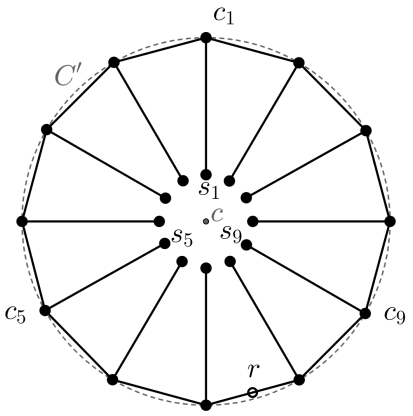
Ahora supongamos que a la rueda se le extrae una muy pequeña área circular desde el centro. El vértice central y los pedazos de radio que tiene los desechamos. La rueda que nos queda sigue siendo circular pero los radios han sido cortados casi al llegar al centro. Por cada vértice $v_i \in C_k$, colocaremos un nuevo vértice s_i en el punto de corte de cada radio (*spoke*), de esta manera tenemos aristas v_is_i para todo $1 \leq i \leq k$.

Familia de n Vértices

Sea entonces un $n > 0$ y asumamos por el momento que n es par. Construiremos la gráfica geométrica descrita teniendo $n/2$ vértices en el exterior y $n/2$ en el interior. Al conjunto de los vértices del exterior lo llamaremos $C_{n/2} = C$ y a los del interior $S_{n/2} = S$. A los elementos de los conjuntos C y S les daremos un orden circular en contra de la manecillas del reloj procurando que: $\forall\ 1 \leq i \leq n/2$, $c_i \in C$ sea el i-ésimo en el orden, $s_i \in S$ sea el i-ésimo en el orden, y la arista c_is_i exista, es decir, el orden es paralelo entre ellos.

Si n es impar entonces haremos el mismo procedimiento anterior con $n-1$, y al vértice restante r lo colocaremos en el ciclo exterior entre cualquier par de vértices consecutivos haciendo una subdivisión a la mitad de la arista que los une. Diremos que este no está en C para no perder la ordenación circular paralela de C y S.

Llamaremos C' a la circunferencia cuyo centro es c que pasa sobre los vértices de C.



Conjuntos Atraídos

Tomemos cualquier par de vértices $c_i \in C$ y $s_i \in S$ de G. Evidentemente c_i puede atraer a s_i. Tracemos entonces la recta $l^{c_i}_{s_i c_i}$ que define $H^{c_i}_{s_i c_i}$. La recta $l^{c_i}_{s_i c_i}$ es perpendicular a $s_i c_i$ por definición, por lo que es perpendicular al radio $c_i c$ de C'. Luego, $l^{c_i}_{s_i c_i}$ es tangente a C' por lo que deja estrictamente a todos los vértices de G en el semiplano contrario a $H^{c_i}_{s_i c_i}$. Entonces, bajo la acción de ningún $c_j \neq c_i$ de C, s_i alcanzará c_i, por lo que ningún $c_j \neq c_i$ de C podrá atraerlo.

Tomemos igualmente cualquier par de vértices $c_i \in C$ y $s_i \in S$ de G. Evidentemente s_i también puede atraer a c_i. Los vértices circularmente adyacentes a c_i en C son c_{i-1} y c_{i+1}. Supongamos que s_i ejerce atracción sobre estos. Entonces como s_i está muy cerca de s_{i-1} y s_{i+1} por construcción, las aristas greedy respecto a s_i de c_{i-1} y c_{i+1} serán $c_{i-1}s_{i-1}$ y $c_{i+1}s_{i+1}$ respectivamente. Entonces, al ejercer s_i acción sobre c_{i-1} y c_{i+1} estos tomarán estas aristas, por lo que no podrán alcanzar s_i. Luego, como ni c_{i-1} y c_{i+1} son atraídos por s_i, entonces ningún otro $c_j \in C$, $c_j \neq c_i$, será atraído por s_i, puesto que a lo más estos alcanzarán c_{i-1} y c_{i+1} que no son atraídos por este. Evidentemente entonces tampoco s_i puede atraer a ningún $s_j \in S$, $s_j \neq s_i$, puesto que estos a lo más alcanzarían algún $c_j \in C$, $c_j \neq c_i$, y vimos que estos no son atraídos por s_i.

Dado un vértice $c_i \in C$, el conjunto de vértices de C que puede atraer c_i no es importante para el análisis que se hará. Respecto al vértice restante r en caso de que n sea impar, no importa por quién pueda ser atraído, lo

importante es que por el mismo análisis hecho respecto a los otros de C, r no puede atraer ningún $s_i \in S$.

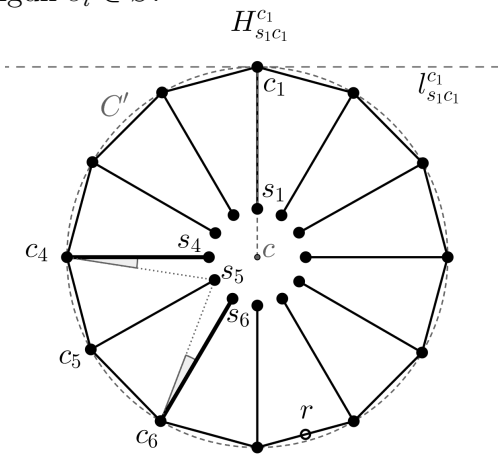


Figura 5.3: En la figura el vértice s_1 solo puede ser atraído por c_1, por ser c_1 el único punto en $H^{c_1}_{s_1c_1}$. El vértice s_5 solo puede atraer c_5, puesto que la atracción sobre c_4 y c_6 los lleva hacia s_4 y s_6 respectivamente, por ser c_4s_4 y c_6s_6 respectivamente las aristas greedy respecto a s_5.

Beacons Necesarios

Entonces cada vértice $s_i \in S$ solo puede ser atraído por su respectivo $c_i \in C$ y por sí mismo, y si se escoge al mismo s_i para atraerse entonces ya sabemos que este solo puede atraer además a su respectivo c_i. Entonces por cada $s_i \in S$ necesitamos un beacon, y $|S| = \lfloor \frac{n}{2} \rfloor$, por lo que se necesitan al menos $\lfloor \frac{n}{2} \rfloor$ beacons para cubrir todos los vértices de G.

Luego, existe una familia infinita de gráficas geométricas G, tal que si G tiene n vértices, entonces se necesitan $\lfloor \frac{n}{2} \rfloor$ beacons para cubrir todos sus vértices. □

Pasaremos ahora a enunciar la cota de suficiencia.

Teorema 5.4.2. *Sea G una gráfica geométrica. Entonces existe un conjunto de vértices en G con cardinalidad a lo más $\lfloor \frac{n}{2} \rfloor$, tal que cubre todos los vértices de G.*

Demostración. Todo par de vértices adyacentes en G se atraen mutuamente a través de la arista que los une. Como G es conexa, entonces existe un árbol T, tal que T es árbol abarcador de G. El árbol abarcador T se puede obtener computacionalmente utilizando por ejemplo un recorrido a lo ancho (BFS). Digamos entonces que se obtuvo T mediante este recorrido.

Entonces como T es un árbol, podemos hacer una bicoloración de sus vértices de manera tal que no dos vértices adyacentes en T tengan el mismo

color. La bicoloración se puede hacer por niveles de profundidad pares o impares en el árbol. Sea B el conjunto de vértices de menor cardinalidad, de los dos formados por la partición en colores de los vértices de T. Entonces $|B| \leq \lfloor \frac{n}{2} \rfloor$, puesto que si no entonces entre los dos conjuntos hubiese más de n vértices.

Entonces colocando un beacon en cada vértice de B cubrimos todos los vértices de G, puesto que ya los vértices de B quedan trivialmente cubiertos, y cada vértice de G que no está en B tiene que tener algún adyacente en B por la bicoloración de T, por lo que también estaría cubierto.

Luego, existe un conjunto de vértices en G con cardinalidad a lo más $\lfloor \frac{n}{2} \rfloor$, tal que cubre todos los vértices de G. □

5.5. Complejidad de Cobertura en Vértices y Aristas

En esta sección probaremos que el problema de cobertura para vértices y aristas en su versión de decisión es NP-completo y en su versión de optimización es NP-difícil. Para la reducción inicial utilizaremos nuevamente el problema *Set Cover*.

Teorema 5.5.1. *Sea G una gráfica geométrica. El problema de decisión Cobertura por Beacons de G en la variante de vértices y aristas es NP-difícil.*

Demostración. Supongamos que tenemos una instancia del problema de decisión *Set Cover*, es decir, tenemos U un conjunto de n elementos llamado universo, S una colección de m subconjuntos de U cuya unión es U, y un entero $k > 0$, y queremos decidir si existe un subconjunto de S de cardinalidad menor o igual a k, cuya unión sea U. Si U es vació la respuesta es trivialmente verdadera para todo S y todo k. Igualmente la respuesta es trivialmente verdadera si $k > m$, pues la unión de S es U. Asumiremos entonces que U no es vació, por lo que S tampoco puede serlo, y que $k \leq m$.

Crearemos entonces una gráfica geométrica G, colocando vértices a partir de U y S, junto con otros vértices extras, y agregando algunas aristas. Partiremos de la misma gráfica geométrica G creada en la demostración del Teorema 5.3.1 pero le haremos algunas transformaciones a esta.

Colocación de Vértices

Además de los vértices y los conjuntos declarados de vértices, colocaremos otro conjunto de vértices U' paralelos a los vértices de U. Estos vértices serán colocados en un recta vertical $x = \varepsilon$ con $\varepsilon > 0$, muy cercana a la recta $x = 0$ donde se encuentran los de U. Por cada vértice $e_i \in U$ crearemos entonces

un vértice $e'_i \in U'$ colocado en $(\varepsilon; i)$, estando el primero en $(\varepsilon; 0)$ y el último en $(\varepsilon; n-1)$.

Cambiaremos un poco las exigencias hechas respecto a los ángulos, por lo que la colocación de los conjuntos cambiará un poco.

Establecimos que los vértices $s_j \in S$ serían colocados en una línea vertical partiendo de $(c; 0)$ para un $c > 0$. Sean $p_{u'}$ y p_s los puntos medios de los segmentos que parten respectivamente desde $(\varepsilon; 0)$ y $(c; 0)$ hasta respectivamente $(\varepsilon; n-1)$ y $(c; n-1)$. La elección de c estará condicionada entonces a que $\sphericalangle s_0 p_{u'} s_{m-1} < 90$, lo cual equivale a decir que $\sphericalangle e'_0 p_s e'_{n-1} < 90$. Esta ubicación de S sobre la recta $x = c$ implica que:

$\sphericalangle e'_i s_j e'_l < 90$ $\forall$ terna de vértices $e'_i, e'_l \in U'$ y $s_j \in S$.

$\sphericalangle e_i s_j e_l < 90$ $\forall$ terna de vértices $e_i, e_l \in U$ y $s_j \in S$.

$\sphericalangle e'_i s_j e_l < 90$ $\forall$ terna de vértices $e'_i \in U'$, $e_l \in U$ y $s_j \in S$.

$\sphericalangle s_j e'_i s_l < 90$ $\forall$ terna de vértices $s_j, s_l \in S$ y $e'_i \in U'$.

$\sphericalangle s_j e_i s_l < 90$ $\forall$ terna de vértices $s_j, s_l \in S$ y $e_i \in U$.

Como se puede notar, los conjuntos U, U' y S están de frente verticalmente, dentro de la misma franja horizontal comprendida entre las rectas $y = 0$ y $y = n-1$. Además, los conjuntos U' y S están a una distancia c que asegura que los ángulos entre: dos cualesquiera de $U \cup U'$ y uno cualquiera de S, y dos cualesquiera de S y uno cualquiera de $U \cup U'$, sean menores de 90.

La coordenada x del vértice b en G seguirá siendo negativa, pero lo suficientemente lejana de los e_i, tal que $\sphericalangle e'_0 b e'_{n-1} < 90$, $\sphericalangle s_0 e'_{n-1} b \geq 90$, $\sphericalangle s_0 e_{n-1} b \geq 90$, $\sphericalangle s_0 e'_0 b \geq 90$ y $\sphericalangle s_0 e_0 b \geq 90$. Las últimas cuatro condiciones equivalen respectivamente a que $\sphericalangle s_{m-1} e'_0 b \geq 90$, $\sphericalangle s_{m-1} e_0 b \geq 90$, $\sphericalangle s_{m-1} e'_{n-1} b \geq 90$ y $\sphericalangle s_{m-1} e_{n-1} b \geq 90$. Esto se puede calcular en $O(1)$ dado que conocemos la posición de s_0, s_{m-1}, e'_0, e'_{n-1}, e_0, e_{n-1} junto con la $y = (n-1)/2$ de b. Esta ubicación de b implica que:

$\sphericalangle s_j e'_i b \geq 90$ $\forall$ par de vértices $s_j \in S$ y $e'_i \in U'$.

$\sphericalangle e'_i s_j b < 90$ $\forall$ par de vértices $s_j \in S$ y $e'_i \in U'$.

$\sphericalangle e_i e'_i b < 90$ $\forall$ par de vértices $e_i \in U$ y $e'_i \in U'$.

Los vértices del conjunto F tendrán sus coordenadas $x's$ negativas pero mayores que la de b, por lo que estarán entre la x de b y la recta $x = 0$. Su coordenada y puede ser cualquiera tal que estos queden en el área definida por $\sphericalangle e'_0 b e'_{n-1}$, es decir, entre los segmentos be'_0 y be'_{n-1}. De esta forma siempre se cumple que:

$\sphericalangle f_f b f_g < 90$ $\forall$ par de vértices $f_f, f_g \in F$.

$\sphericalangle e'_i b f_f < 90$ $\forall$ par de vértices $e'_i \in U'$ y $f_f \in F$.

Para los tríos de vértices que quedan en línea recta, podemos siempre hacer una pequeña perturbación para lograr la posición general.

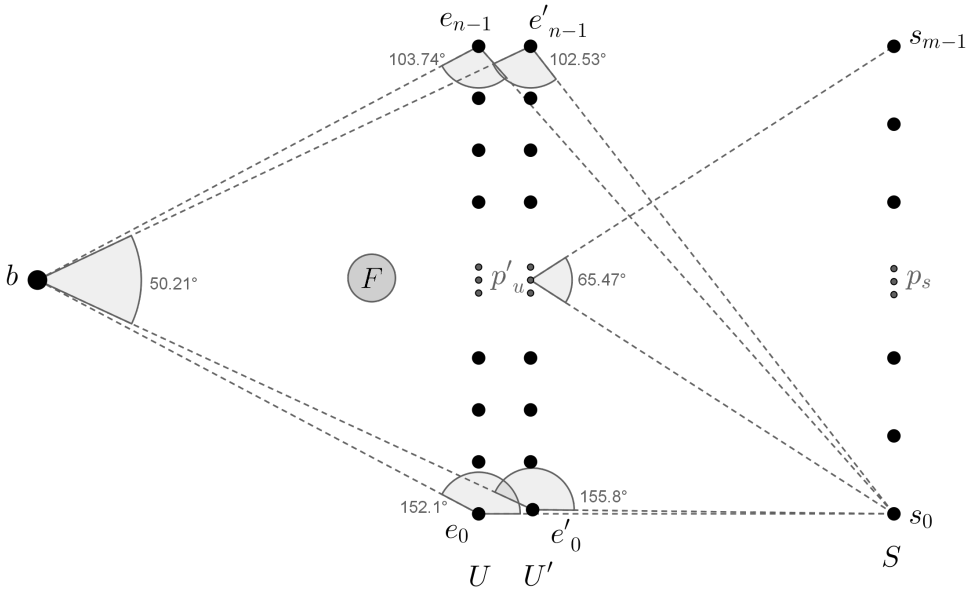


Figura 5.4: En la figura se muestra la colocación de los vértices con los ángulos exigidos. Nótese que: $\sphericalangle s_0 p_{u'} s_{m-1} < 90$, $\sphericalangle e'_0 b e'_{n-1} < 90$, $\sphericalangle s_0 e'_{n-1} b \geq 90$, $\sphericalangle s_0 e_{n-1} b \geq 90$, $\sphericalangle s_0 e'_0 b \geq 90$ y $\sphericalangle s_0 e_0 b \geq 90$.

Colocación de Aristas

Eliminemos todas las aristas de G para explicar solo las que estarán. Crearemos una arista $e'_i s_j$ entre un vértice $e'_i \in U'$ y un vértice $s_j \in S$ si y solo si, el subconjunto s_j contiene al elemento e_i, a estas aristas las agruparemos en el conjunto $E_{SU'}$. Desde cada $e'_i \in U'$ insertaremos una arista a b, a estas aristas las agruparemos en el conjunto $E_{U'b}$. Desde cada $e'_i \in U'$ insertaremos una arista a su respectivo $e_i \in U$, a estas aristas las agruparemos en el conjunto $E_{UU'}$. Por último, desde cada $f_f \in F$ insertaremos una arista a b, a estas aristas las agruparemos en el conjunto E_{Fb}. Estas son las únicas aristas que tendrá G.

Como se puede notar, G es una gráfica geométrica conexa, puesto que desde todo vértice existe un camino a b, por lo que entre todo par de vértices existe un camino.

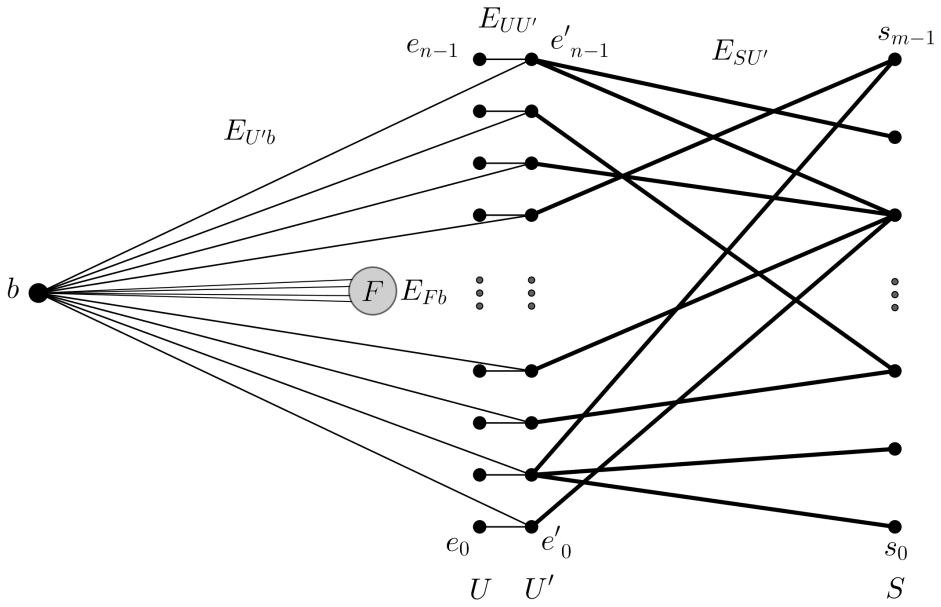


Figura 5.5: En la figura se muestra un ejemplo de la reducción. Las aristas gruesas unen los e'_i con los s_j, y se colocan si y solo si $e_i \in s_j$ en *Set Cover*.

Conjuntos Atraídos

Los conjuntos atraídos de vértices y aristas por cada vértice de G son los siguientes:

- $A(b) = \{b\} \cup E_{Fb} \cup F \cup E_{U'b} \cup U' \cup E_{SU'} \cup S$.
- $A(f_f) = \{f_f, bf_f, b\}$, $\forall$ vértice $f_f \in F$.
- $A(e_i) = \{e_i, e_ie'_i, e'_i\} \cup \{s_je'_i, s_j : s_je'_i \in G\}$, $\forall$ vértice $e_i \in U$.
- $A(e'_i) = \{e'_i, be'_i, b, e_ie'_i, e_i\} \cup \{s_je'_i, s_j : s_je'_i \in G\}$, $\forall$ vértice $e'_i \in U'$.
- $A(s_j) = \{s_j\} \cup \{e'_is_j, e'_i, e_ie'_i, e_i, be'_i : e'_is_j \in G\} \cup \{b\}$, $\forall$ vértice $s_j \in S$.

Algoritmo Polinomial para Cobertura por Beacons

Supongamos entonces que tenemos un algoritmo polinomial para el problema de decisión *Cobertura por Beacons* en la variante de vértices y aristas, y le pasamos como parámetros la gráfica geométrica G y $k+1$. El algoritmo entonces devolvería "verdadero" si y solo si existe un conjunto de vértices (beacons) B de cardinalidad menor igual a $k+1$, tal que todo vértice y arista en G es atraído por el menos uno de ellos. Mostraremos entonces como serían elegidos los beacons en G.

El vértice b siempre será escogido para estar en B por razón homóloga a la explicada en la demostración del Teorema 5.3.1.

Al ser escogido necesariamente b como beacon, entonces no es necesario escoger ningún vértice de F como beacon para cubrir $F \cup E_{Fb}$. Estos solo pudieran ser elegidos por exceso si con menos de k se cubrió $E_{U'b} \cup U' \cup E_{UU'} \cup U \cup E_{SU'} \cup S$.

Al ser escogido necesariamente b como beacon, entonces también $E_{U'b} \cup U' \cup E_{SU'} \cup S$ quedan atraídos, por lo que desaparece el incentivo de atraerlos. Entonces queda solo por atraer $E_{UU'} \cup U$. Obsérvese en los conjuntos atraídos, que todo vértice $u \in G$ atrae $e_i \in U$ si y solo si u atrae a $e_i e'_i$, es decir, $e_i \in A(u) \iff e_i e'_i \in A(u)$. Esto implica que es equivalente atraer U a atraer $E_{UU'}$, por lo que podemos decir que solo queda por atraer a U.

Entonces quedan solo por atraer los vértices de U, y se pueden utilizar a lo más k beacons, por lo que el algoritmo respondería "verdadero" si y solo si es posible atraerlos a todos con una cantidad menor o igual a k.

Interpretando la Salida

Supongamos que el algoritmo responde "falso". Entonces no existiría ningún subconjunto $B \in U \cup U' \cup S$ con $|B| \leq k$, tal que cada vértice de U quede cubierto por al menos uno de B. Entonces evidentemente tampoco puede pasar que exista algún $B \in S$ con $|B| \leq k$, tal que cada vértice de U quede cubierto por al menos uno de B.

Supongamos que el algoritmo responde "verdadero". Entonces existiría un subconjunto $B \in U \cup U' \cup S$ con $|B| \leq k$, tal que cada vértice de U queda cubierto por al menos uno de B, lo que equivale a decir que B cubre $U \cup E_{UU'}$. Como el resto de la gráfica ya está cubierta por b, entonces las únicas razones por las que un $e_i \in U$ o un $e'_i \in U'$ pertenecerían a B, serían porque están cubriendo a $\{e_i, e_i e'_i\}$ y nadie más lo está cubriendo, o por exceso. Pero como cada e'_i tiene que estar conectado con al menos un $s_j \in S$, y s_j puede atraer $\{e_i, e_i e'_i\}$, entonces podemos transformar B intercambiando e_i o e'_i por s_j, quedando B con la misma cardinalidad o menor y sin contener e_i o e'_i. Entonces este razonamiento lo podemos repetir hasta que no quede ningún $e \in U \cup U'$ en B. Luego, si el algoritmo responde "verdadero", entonces existiría un subconjunto $B \in S$ con $|B| \leq k$, tal que $U \cup E_{UU'}$ queda cubierto. Entonces, si el algoritmo responde "verdadero", existiría un subconjunto $B \in S$ con $|B| \leq k$, tal que cada vértice de U queda cubierto por al menos uno de B.

En cualquiera de ambos casos, podemos interpretar la salida del algoritmo respecto a si existe o no un subconjunto $B \in S$ con $|B| \leq k$, tal que cada vértice de U quede cubierto por al menos uno de B. Además, tenemos que cada $s_j \in S$ solo podía atraer un $e_i \in U$ si y solo si existía arista $e'_i s_j$ en G, y cada arista $e'_i s_j$ fue colocada en G si y solo si el elemento e_i pertenecía al subconjunto s_j en *Set Cover*.

Entonces podemos directamente interpretar la salida del algoritmo respecto al problema de decisión *Set Cover*. Si la respuesta del algoritmo con G y $k+1$ fue "falso" para el problema de decisión *Cobertura por Beacons*, entonces para el problema de decisión *Set Cover* no existe un subconjunto de S de cardinalidad menor o igual a k, cuya unión sea U. Si la respuesta del algoritmo con G y $k+1$ fue "verdadero" para el problema de decisión *Cobertura por Beacons*, entonces para el problema de decisión *Set Cover* existe un subconjunto de S de cardinalidad menor o igual a k, cuya unión es U.

Complejidad

Luego, si existiera tal algoritmo polinomial para el problema de decisión *Cobertura por Beacons* en la variante de vértices y aristas, tendríamos un algoritmo polinomial para el problema de decisión *Set Cover*, puesto que:

La transformación inicial descrita de *Set Cover* a la gráfica geométrica G es polinomial, el algoritmo para el problema de decisión *Cobertura por Beacons* de G en la variante de vértices y aristas sería polinomial, y la interpretación de la salida de este para *Set Cover* es trivialmente polinomial.

Luego, como el problema de decisión *Set Cover* es NP-difícil, entonces el problema de decisión *Cobertura por Beacons* en la variante de vértices y aristas es NP-difícil. □

Teniendo que el problema de decisión es NP y también NP-difícil, enunciamos entonces inmediatamente la NP-completitud de este.

Teorema 5.5.2. *Sea G una gráfica geométrica. El problema de decisión Cobertura por Beacons de G en la variante de vértices y aristas es NP-completo.*

Demostración. Por Teorema 5.2.8 tenemos que el problema es NP. Por Teorema 5.5.1 tenemos que el problema es NP-difícil. Luego, el problema es NP-completo. □

Finalmente, partiendo de que el problema de decisión es NP-difícil, enunciaremos entonces que el problema de optimización es también NP-difícil.

Teorema 5.5.3. *Sea G una gráfica geométrica. El problema de optimización Cobertura por Beacons de G en la variante de vértices y aristas es NP-difícil.*

Demostración. Supongamos que tenemos una instancia del problema de decisión con G y un entero $k > 0$ como parámetros. Supongamos entonces que tenemos un algoritmo polinomial para el problema de optimización. Entonces ejecutaríamos dicho algoritmo sobre G, obteniendo como salida un conjunto B de cardinalidad mínima, tal que cada vértice y arista de G es atraído por al menos un $b \in B$.

Si $|B| > k$, entonces es imposible que exista un conjunto de cardinalidad menor o igual a k tal que cada vértice y arista sea atraído, pues B tenía cardinalidad mínima. Entonces, devolveríamos "falso" para el problema de decisión.

Si $|B| \leq k$, entonces existe un conjunto de cardinalidad menor o igual a k tal que cada vértice y arista es atraído, el propio B. Entonces, devolveríamos "verdadero" para el problema de decisión.

Luego, si existiera dicho algoritmo polinomial para el problema de optimización, tendríamos un algoritmo polinomial para el problema de decisión, puesto que: la transformación del problema de decisión al de optimización es trivialmente polinomial, el algoritmo de optimización sería polinomial, y la transformación de la salida de este al de decisión es trivialmente polinomial.

Luego, como el problema de decisión *Cobertura por Beacons* en la variante de vértices y aristas es NP-difícil, entonces el problema de optimización *Cobertura por Beacons* en la variante de vértices y aristas es NP-difícil. □

5.6. Combinatoria de Cobertura en Vértices y Aristas

En esta sección expondremos una cota de beacons necesarios para cubrir gráficas geométricas con n vértices en la variante de vértices y aristas. Para exponer la cota mostraremos una familia infinita de gráficas que necesitan siempre una cierta cantidad para ser cubiertas. Para la cota de suficiencia no tenemos ningún resultado que no sea trivial, por lo que solo mencionamos aquí que con $n - 1$ beacons siempre podemos cubrir G dado que es conexa.

Teorema 5.6.1. *Existe una familia infinita de gráficas geométricas G, tal que si G tiene n vértices entonces se necesitan $\lfloor \frac{2n}{3} \rfloor$ beacons para cubrir todos sus vértices y aristas.*

Demostración. Sea $\triangle abc$ un triángulo equilátero. Entonces para cubrir $\triangle abc$ necesitamos dos beacons: Tomemos un vértice cualquiera del triángulo, digamos a. Trivialmente a atrae al conjunto $\{a, ba, b, ca, c\}$, por estar directamente comunicados con a. Pero para el caso de la arista bc, esta no puede ser atraída por a pues $\sphericalangle abc = 60 < 90$ y $\sphericalangle acb = 60 < 90$. De hecho, por ser $\triangle abc$ equilátero entonces la recta desde a al punto medio de bc forma una perpendicular con bc, por lo que el punto medio de bc sería un punto muerto respecto a a. Entonces si tomamos a a como beacon necesitaríamos tomar alguno de los otros dos vértices, b o c, para poder cubrir bc, por lo que se necesitarían dos beacons para cubrir el triángulo. Luego, como la elección de

a fue arbitraria entonces esto ocurre homólogamente también para b y c, por lo que para cubrir un $\triangle abc$ equilátero cualquiera se necesitan dos beacons.

En general para cubrir cualquier triángulo acutángulo se necesitarían dos beacons por ser todos los ángulos menores de 90 grados.

Familia de n Vértices

Asumiremos por el momento que n es múltiplo de tres. Utilizaremos para la demostración un conjunto de $n/3$ triángulos equiláteros encajados recursivamente sobre un mismo centro y con la misma orientación.

Tomemos primero tres vértices y construyamos un primer triángulo equilátero $\triangle a_1b_1c_1$ con centro c. Dentro de este colocaremos un segundo triángulo equilátero $\triangle a_2b_2c_2$, el cual estará cercano a $\triangle a_1b_1c_1$, tendrá el mismo centro c y la misma orientación. Básicamente $\triangle a_2b_2c_2$ es una copia de $\triangle a_1b_1c_1$ un poco más pequeña y reducida respecto al centro c. A partir de $\triangle a_2b_2c_2$ construiremos otro triángulo $\triangle a_3b_3c_3$ con las mismas características respectivas a $\triangle a_2b_2c_2$, y así recursivamente construiremos los $n/3$ triángulos equiláteros. Según la profundidad i en la recursión, llamaremos entonces al triángulo respectivo $\triangle a_ib_ic_i$ con $1 \geq i \geq n/3$.

Para comunicar estos triángulos insertaremos una arista desde cada triángulo exterior a su triángulo inmediato interior, desde el vértice b_i de $\triangle a_ib_ic_i$ al vértice c_{i+1} de $\triangle a_{i+1}b_{i+1}c_{i+1}$ $\forall\ 1 \geq i < n/3$. Sin perdida de generalidad asumiremos que: b_ic_i es paralelo al eje x, a_i está arriba, b_i está a la izquierda y c_i a la derecha, $\forall\ 1 \geq i \geq n/3$.

Luego las aristas de esta gráfica geométrica G son las aristas a_ib_i, b_ic_i y c_ia_i $\forall\ 1 \geq i \geq n/3$, y las aristas b_ic_{i+1} $\forall\ 1 \geq i < n/3$.

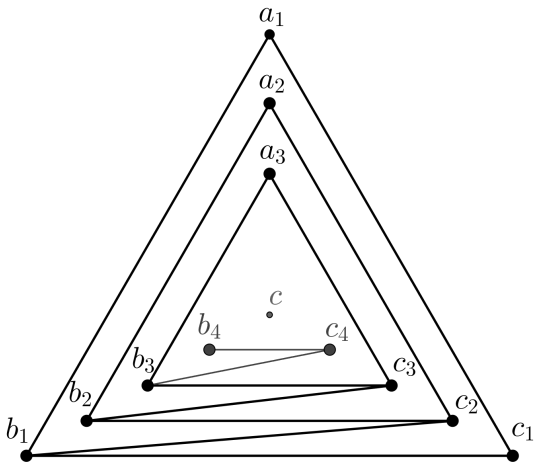


Figura 5.6: En la figura el vértice c_4 y la arista b_3c_4 serían colocados si n mód $3 = 1$. Los vértices c_4 y b_4, y las aristas b_3c_4 y c_4b_4, serían colocados si n mód $3 = 2$.

Conjuntos Atraídos

Sea $\triangle a_i b_i c_i$ con $1 < i < n/3$ un triángulo interior de la gráfica. Los conjuntos atraídos por sus vértices serían:

- $A(a_i) = \{a_i, b_i a_i, b_i, c_i a_i, c_i\}$.
- $A(b_i) = \{b_i, a_i b_i, a_i, c_i b_i, c_i, c_{i+1} b_i, c_{i+1}\}$.
- $A(c_i) = \{c_i, a_i c_i, a_i, b_i c_i, b_i, b_{i-1} c_i, b_{i-1}\}$.

El lector puede notar que los vértices del $\triangle a_i b_i c_i$ no pueden atraer ninguna de las aristas propias del $\triangle a_{i-1} b_{i-1} c_{i-1}$, puesto que: Para toda arista $uv \in \{a_{i-1} b_{i-1}, b_{i-1} c_{i-1}, c_{i-1} a_{i-1}\}$ propia del $\triangle a_{i-1} b_{i-1} c_{i-1}$ y para todo vértice b del $\triangle a_i b_i c_i$, se cumple que $b \in B_{uv}$ por estar b en el interior del $\triangle a_{i-1} b_{i-1} c_{i-1}$.

El lector puede notar además que los vértices del $\triangle a_i b_i c_i$ no pueden atraer ninguna de las aristas propias del $\triangle a_{i+1} b_{i+1} c_{i+1}$, puesto que: Al ejercer el vértice a_i acción sobre $b_{i+1} a_{i+1}$ y $c_{i+1} a_{i+1}$, ambas alcanzarían a_{i+1} siendo este evidentemente un punto muerto respecto a a_i. Al ejercer el vértice a_i acción sobre $b_{i+1} c_{i+1}$, esta alcanzaría un punto muerto respecto a a_i en el punto medio de $b_{i+1} c_{i+1}$, por estar a_i y a_{i+1} orientados respecto al mismo centro. Homólogamente ocurre si los vértices b_i y c_i intentan atraer las aristas propias del $\triangle a_{i+1} b_{i+1} c_{i+1}$.

Para el caso de los triángulos $\triangle a_1 b_1 c_1$ y $\triangle a_{n/3} b_{n/3} c_{n/3}$, los conjuntos atraídos serían los mismos que para los demás triángulos pero eliminando los elementos del triángulo inexistente más exterior o del triángulo inexistente más interior respectivamente.

Estas afirmaciones implican entonces que los vértices de un triángulo $\triangle a_i b_i c_i$, no pueden atraer ninguna de las aristas propias de ningún otro triángulo de G, ya que necesariamente en el camino de atracción estas tendrían que pasar, o por las aristas del triángulo inmediato exterior, o por las aristas del triángulo inmediato interior, según el caso, y vimos que estas no son atraídas.

Beacons Necesarios

Entonces las aristas de cada triángulo (equilátero) de G solo pueden ser atraídas por vértices del propio triángulo, y vimos que se necesitan dos vértices de un triángulo equilátero para poder atraer sus aristas. Entonces se necesitan dos vértices por cada triángulo de G para poder atraer todas sus aristas, por lo que se necesitan (al menos) dos vértices por cada triángulo de G para poder atraer todas sus aristas y vértices

Si $n \mod 3 = 0$, entonces en G quedarían $\frac{n}{3}$ triángulos y se necesitarían $\frac{2n}{3} = \lfloor \frac{2n}{3} \rfloor$ vértices (beacons) para cubrirla.

Si n mód $3 = 1$, entonces $n = 3k + 1$ para un entero $k \geq 0$. Entonces $\lfloor \frac{2n}{3} \rfloor = \frac{2(n-1)}{3}$, pues:

$\lfloor \frac{2n}{3} \rfloor = \lfloor \frac{2(3k+1)}{3} \rfloor = \lfloor \frac{6k+2}{3} \rfloor = \lfloor \frac{6k}{3} + \frac{2}{3} \rfloor = \lfloor 2k + \frac{2}{3} \rfloor = 2k + \lfloor \frac{2}{3} \rfloor = 2k$

$\frac{2(n-1)}{3} = \frac{2(3k+1-1)}{3} = \frac{2(3k)}{3} = 2k$

El vértice extra se colocaría como un $c_{\lfloor \frac{n}{3} \rfloor + 1}$ de un triángulo inexistente $\triangle a_{\lfloor \frac{n}{3} \rfloor + 1} b_{\lfloor \frac{n}{3} \rfloor + 1} c_{\lfloor \frac{n}{3} \rfloor + 1}$, y estaría conectado mediante la arista $c_{\lfloor \frac{n}{3} \rfloor + 1} b_{\lfloor \frac{n}{3} \rfloor}$. Por las mismas razones explicadas anteriormente, $c_{\lfloor \frac{n}{3} \rfloor + 1}$ no puede atraer a ninguna arista de los triángulos por lo que no tendría sentido ponerlo como beacon. Tomando el vértice $b_{\lfloor \frac{n}{3} \rfloor}$ como beacon en $\triangle a_{\lfloor \frac{n}{3} \rfloor} b_{\lfloor \frac{n}{3} \rfloor} c_{\lfloor \frac{n}{3} \rfloor}$, se cubre también $\{c_{\lfloor \frac{n}{3} \rfloor + 1} b_{\lfloor \frac{n}{3} \rfloor}, c_{\lfloor \frac{n}{3} \rfloor + 1}\}$ por lo que no hace falta otro beacon.

Luego, se necesitarían un par de beacons por cada triángulo en G y hay $\frac{(n-1)}{3}$ de estos. Luego se necesitarían $\frac{2(n-1)}{3} = \lfloor \frac{2n}{3} \rfloor$ para cubrir G.

Si n mód $3 = 2$, entonces $n = 3k + 2$ para un entero $k \geq 0$. Entonces $\lfloor \frac{2n}{3} \rfloor = \frac{2(n-2)}{3} + 1$, pues:

$\lfloor \frac{2n}{3} \rfloor = \lfloor \frac{2(3k+2)}{3} \rfloor = \lfloor \frac{6k+4}{3} \rfloor = \lfloor \frac{6k}{3} + \frac{4}{3} \rfloor = \lfloor 2k + \frac{4}{3} \rfloor = 2k + \lfloor \frac{4}{3} \rfloor = 2k + 1$

$\frac{2(n-2)}{3} + 1 = \frac{2(3k+2-2)}{3} + 1 = \frac{2(3k)}{3} = 2k + 1$

Los vértices extras se colocarían como $c_{\lfloor \frac{n}{3} \rfloor + 1}$ y $b_{\lfloor \frac{n}{3} \rfloor + 1}$, vértices de un triángulo inexistente $\triangle a_{\lfloor \frac{n}{3} \rfloor + 1} b_{\lfloor \frac{n}{3} \rfloor + 1} c_{\lfloor \frac{n}{3} \rfloor + 1}$, y estarían conectados mediante la aristas $c_{\lfloor \frac{n}{3} \rfloor + 1} b_{\lfloor \frac{n}{3} \rfloor}$ y $b_{\lfloor \frac{n}{3} \rfloor + 1} c_{\lfloor \frac{n}{3} \rfloor + 1}$. Por las mismas razones explicadas anteriormente, ni $c_{\lfloor \frac{n}{3} \rfloor + 1}$ ni $b_{\lfloor \frac{n}{3} \rfloor + 1}$ pueden atraer a ninguna arista de los triángulos, por lo que solo tendría sentido poner alguno como beacon para asegurar que estos dos y su arista queden atraídos. Tomando el vértice $b_{\lfloor \frac{n}{3} \rfloor}$ como beacon en $\triangle a_{\lfloor \frac{n}{3} \rfloor} b_{\lfloor \frac{n}{3} \rfloor} c_{\lfloor \frac{n}{3} \rfloor}$, se cubren $c_{\lfloor \frac{n}{3} \rfloor + 1} b_{\lfloor \frac{n}{3} \rfloor}$ y $c_{\lfloor \frac{n}{3} \rfloor + 1}$, pero falta por cubrir $b_{\lfloor \frac{n}{3} \rfloor + 1} c_{\lfloor \frac{n}{3} \rfloor + 1}$ y $b_{\lfloor \frac{n}{3} \rfloor + 1}$, por lo que se necesita un beacon más.

Luego, se necesitaría un beacon extra, más un par de beacons por cada triángulo en G, y hay $\frac{(n-2)}{3}$ de estos. Luego se necesitarían $\frac{2(n-2)}{3} + 1 = \lfloor \frac{2n}{3} \rfloor$ para cubrir G.

Luego, existe una familia infinita de gráficas geométricas G, tal que si G tiene n vértices, entonces se necesitan $\lfloor \frac{2n}{3} \rfloor$ beacons para cubrir todos sus vértices y aristas. □

Referencias

[1] Sang Won Bae, Chan-Su Shin, and Antoine Vigneron. Tight bounds for beacon-based coverage in simple rectilinear polygons. In *Latin American Symposium on Theoretical Informatics*, pages 110–122. Springer, 2016.

[2] Michael Biro. *Beacon-based routing and guarding*. PhD thesis, STATE UNIVERSITY OF NEW YORK AT STONY BROOK, 2013.

[3] Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph SB Mitchell. Beacon-based routing and coverage. In *21st Fall Workshop on Computational Geometry (FWCG 2011)*, 2011.

[4] Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph SB Mitchell. Combinatorics of beacon-based routing and coverage. In *Proceedings of the 25th Canadian Conference on Computational Geometry (CCCG)*, 2013.

[5] Michael Biro, Justin Iwerks, Irina Kostitsyna, and Joseph SB Mitchell. Beacon-based algorithms for geometric routing. In *Workshop on Algorithms and Data Structures*, pages 158–169. Springer, 2013.

[6] JA Bondy and USR Murty. Graph theory (graduate texts in mathematics), 2008.

[7] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[8] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[9] Der-Tsai Lee and Arthur K Lin. Computational complexity of art gallery problems. In *Autonomous robot vehicles*, pages 303–309. Springer, 1990.

[10] Thomas C Shermer. A combinatorial bound for beacon-based routing in orthogonal polygons. *arXiv preprint arXiv:1507.03509*, 2015.