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

ALGORITMOS LOCALES PARA RUTEO EN REDES AD HOC

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
ANA KAREN FLORES GARCÍA

TUTOR
DR. JORGE URRUTIA GALICIA
INSTITUTO DE MATEMÁTICAS, UNAM

CIUDAD DE MÉXICO, MARZO DE 2022

Índice general

Glosario

d distancia euclidiana

u vértice del conjunto V

e arista del conjunto E

p camino de una gráfica G

ϵ elipse con focos s y t y eje mayor de longitud l

V conjunto de vértices de una gráfica

E conjunto de aristas de una gráfica

S subgráfica de una gráfica G

$G[S]$ subgráfica inducida por los vértices de S en G

$K_{1,6}$ gráfica estrella de orden 6

Δ grado máximo de una gráfica G

(u, v) arista con extremos u y v

N_v vecindad de v

N_v^+ vecindad de v incluyéndose a sí mismo

$w(u, v)$ peso asignado a la arista (u, v)

$w_{enlace}(e)$ métrica de enlace

$w_{dist}(e)$ métrica euclidiana

$w_{energia(a)}(e)$ métrica de energía con exponente de atenuación a

$c(e)$ costo asignado a la arista e

$c(p)$ costo de un camino p

$c(\mathcal{A})$ costo de un algoritmo $\mathcal{A}$

$c_{enlace}(e)$ costo basado en la métrica de enlace

$c_{dist}(e)$ costo baso en la métrica euclidiana

$c_{energia(a)}(e)$ costo basado en la métrica de energía

$gr(v)$ grado de un vértice v

(u, v, w) triángulo con vértices u, v y w

$\Omega(f(n))$ cota inferior asintótica de la función f

$O(f(n))$ cota superior asintótica de la función f

$\Theta(f(n))$ cota inferior y superior asintótica de la función f

$|\cdot|$ cardinalidad de un conjunto

$\overline{st}$ línea definida por los vértices s y t

GDU Gráfica de Disco Unitario

CDM Conjunto Dominante Mínimo

CDMC Conjunto Dominante Mínimo Conexo

CI Conjunto Independiente

CIM Conjunto Independiente Maximal

GG subgráfica de Gabriel

GVR subgráfica de vecindad relativa

GY subgráfica de Yao

GCC subgráfica de cobertura por conos

TD triangulación de Delaunay

AGPM árbol generador de peso mínimo

RG Ruteo Glotón

RC Ruteo por Caras

RCA Ruteo por Caras Acotado

RCAD Ruteo por Caras Adaptado

ORC Otro Ruteo por Caras

ORCA Otro Ruteo por Caras Acotado

ORCAD Otro Ruteo por Caras Adaptado

ORCADG Otro Ruteo por Caras Adaptado Glotón

Índice de figuras

Lista de Algoritmos

Introducción

El ruteo (del inglés *routing*) en una red de comunicación es el proceso de enviar un mensaje desde una fuente, llamada *emisor*, hacia un destino llamado *receptor*, por medio de puntos intermediarios llamados *nodos*. En redes alámbricas, dicho ruteo suele ser llevado a cabo por nodos específicos llamados *enrutadores*, los cuales son dispositivos de red que pueden transferir datos entre nodos estableciendo la ruta a seguir y su objetivo es dirigir el tráfico entrante y saliente en la red de manera eficiente. En contraste, en redes inalámbricas *ad hoc*, cada nodo de la red puede tener la función de un enrutador que participa en el envío de un mensaje de un nodo a otro. Este proceso es particularmente importante en redes *ad hoc*, puesto que podemos pensar en los nodos como pequeñas estaciones móviles de radio que tienen restricciones en cuanto a sus recursos de energía y, por lo tanto, intentan enviar mensajes a baja potencia de transmisión, ver figura 1. Esto implica que el destino de un mensaje será alcanzado desde la fuente sólo si se cumplen ciertos requerimientos.

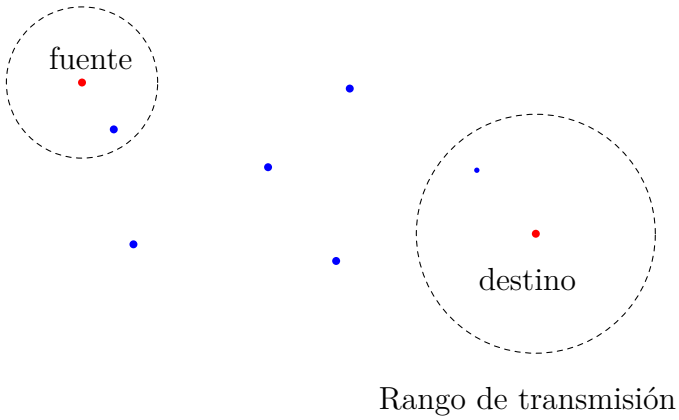
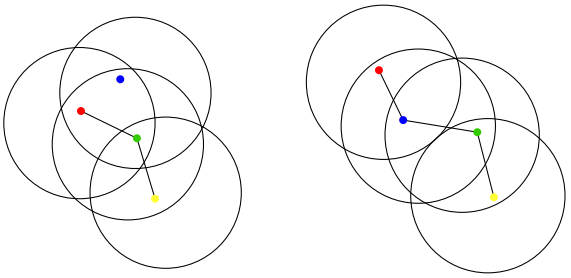


FIGURA 1: Red *ad hoc* con nodo fuente, nodo destino y sus respectivos rangos de transmisión.

Las redes inalámbricas convencionales dependen de una infraestructura compuesta por estaciones de base que interactúan con dispositivos móviles en una relación cliente - servidor (e.g. en redes como INTERNET). Actualmente, la investigación se centra en desarrollar técnicas para garantizar una comunicación eficiente en redes con topología muy dinámica[2] y que, sin embargo, deben ser capaces de comunicarse entre ellas.

Ejemplos de estos sistemas son las llamadas redes *ad hoc*. Se define una red *ad hoc* como un sistema autónomo de servidores móviles conectados inalámbricamente y cuya unión forma una red de comunicación modelada como una *gráfica de comunicación* (ver [40], sección 2.2.1).

El ruteo en redes alámbricas suele llevarse a cabo en condiciones estables, al menos en cuanto a la topología de la red, la cual puede permanecer inmutable por largos periodos de tiempo. Las redes inalámbricas *ad hoc* son de un carácter fundamentalmente diferente: por ejemplo, las conexiones inalámbricas son por naturaleza menos estables que las redes alámbricas. Esto implica que los sistemas de ruteo en este tipo de redes deben ser capaces de lidiar con varios problemas que pueden influir en la propagación de señales de radio, entre los cuales se encuentran el blindaje, la reflexión, la dispersión y la interferencia. Por ejemplo, al moverse los nodos de una red, la topología de la red de comunicación puede cambiar, ver figura 2. Debido a esto, en general, los requerimientos de una red *ad hoc* no son satisfechos por los protocolos tradicionales de ruteo.



(a) (b)

FIGURA 2: En (a) el nodo rojo usa al nodo verde para llegar al nodo amarillo. Debido al movimiento, en (b) el nodo rojo usa ahora a los nodos azul y verde para llegar al nodo amarillo.

[2] Es decir, los nodos de la red se mueven rápido y libremente.

Se han diseñado un número considerable de algoritmos de ruteo cuyo objetivo es lidiar con problemas de las redes *ad hoc*, como los previamente mencionados. Usualmente, estos algoritmos se dividen en dos grupos: los "proactivos" y los "reactivos".

Los algoritmos de ruteo *proactivos* acumulan información del ruteo antes de tiempo a través de tablas de ruteo, lo cual implica que en redes como internet, cuya topología cambia constantemente, dichas tablas tengan que ser actualizadas frecuentemente. En cambio, los algoritmos *reactivos* tienen por objetivo principal reaccionar según la demanda y el estado de la red.

La mayoría de los algoritmos de ruteo, aquellos desarrollados hasta finales del siglo XX, han sido estudiados y analizados desde un punto de vista centralizado y global, lo cual no es adecuado para las redes *ad hoc*. En contraste, un tipo específico de algoritmos de ruteo ha sido diseñado: el *ruteo geográfico*, también llamado *direccional* o *geométrico*, *basado en la ubicación* o *basado en la posición*. Este tipo de algoritmos consiste en dos suposiciones principales: que cada nodo conoce la posición de sus nodos adyacentes y que el nodo que quiere enviar un mensaje a un destinatario conoce su propia posición y la posición del nodo destino. Estos supuestos se justifican por la llegada de los sistemas miniatura de bajo costo y de los servicios de ubicación. El ruteo geográfico es particularmente útil en este contexto, puesto que una vez conocida la posición del nodo destino, el resto de las operaciones son estrictamente locales, esto es, cada nodo sólo tiene que mantener la información de sus nodos adyacentes.

Los algoritmos para el ruteo geográfico cuya implementación sólo necesitan de operaciones locales, son llamados *algoritmos locales*. Estos son una particularidad entre los llamados *algoritmos distribuidos*: sistemas de unidades autónomas de cómputo que son capaces de comunicarse entre sí.

Los algoritmos locales para el ruteo geográfico suelen implementarse en *gráficas de disco unitario*, las cuales son gráficas cuyos vértices representan estaciones (nodos) de transmisión con un rango uniforme y cuyas aristas están definidas si la distancia euclidiana, d, entre sus vértices es a lo más 1 (ver figura 3). Comúnmente, son usadas para modelar varios tipos de redes, en particular, redes *ad hoc*.

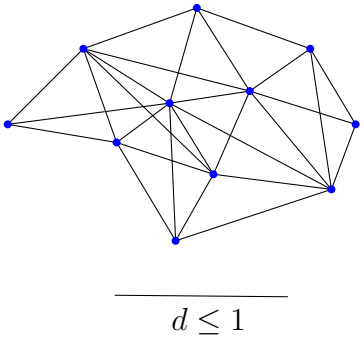


FIGURA 3: Gráfica de disco unitario del conjunto de puntos azules, en la cual sus aristas tienen una distancia menor o igual a uno.

Algoritmos Locales. En este trabajo analizaremos una nueva clase de algoritmos que han sido desarrollados para resolver varios problemas en redes, en particular, en redes que pueden ser modeladas por gráficas de disco unitario. Estos algoritmos emplean sólo operaciones locales y la posición de los vértices de las gráficas. Definamos un *algoritmo local* como un algoritmo en el cual el procesador (vértice) sólo trabaja con la información obtenida de los vértices que están a una distancia de, a lo más, k saltos, donde k es una constante en algunos casos menor o igual que 4. Además, ningún tipo de información extra como el número de vértices o la topología de la red es dada a los procesadores [39].

El análisis de este trabajo se enfocará en algoritmos de ruteo locales, en particular en el ya mencionado ruteo geográfico. Observemos que el ruteo geográfico no puede ser considerado como una versión del "enrutamiento de origen" ajustado a redes dinámicas: mientras que en el enrutamiento de origen la ruta completa que tiene que seguir el mensaje es dada por el nodo fuente, en el ruteo geográfico cada nodo simplemente direcciona el mensaje según la posición del nodo destino. Comúnmente se considera que el nodo destino se puede mover ligeramente en comparación con la frecuencia en los cambios topológicos entre el nodo fuente y el nodo destino. Por esta razón, en los algoritmos de tipo local, sólo es necesario mantener información local de los nodos, así como la posición del nodo de partida y la del destino; si el nodo destino no se mueve muy rápido, el mensaje es entregado sin importar los posibles cambios de los nodos intermediarios. Finalmente, desde un punto de vista menos técnico, podemos esperar que a través del estudio del ruteo geográfico sea posible obtener conocimiento sobre las redes *ad hoc* en general, sin necesitar la información de la posición de todos los nodos de la red [25, 40].

El enfoque que abrió camino en el estudio de los algoritmos para ruteos geográficos fue uno sencillo por medio de un *algoritmo glotón* (del inglés *greedy*). En este algoritmo, para generar la ruta del mensaje, el vértice que representa al nodo fuente lo envía directamente a su vecino más cercano al vértice que representa al nodo destino [11, 15]. A este tipo de ruteo se le conoce como *ruteo glotón*. Comparado con otros algoritmos, el ruteo glotón tiene muy poca sobrecarga de tráfico y tiene buen funcionamiento cuando la cantidad de nodos aumenta. Sin embargo, existe la posibilidad de que este tipo de ruteo no logre entregar el mensaje al nodo destino, puesto que podría pasar que lleguemos a un callejón sin salida, es decir, a un nodo tal que el vértice que lo respresenta no tiene vecinos más cercanos al vértice que representa al nodo destino, que no sea él mismo (al vértice que representa a este nodo se le llama *mínimo local*) [11, 14].

Una técnica que puede asegurar la entrega del mensaje en gráficas planas, fijas [3] y conexas, es el llamado *ruteo por caras* [11, 14, 15]. El ruteo por caras consiste en enviar el mensaje a lo largo de los límites de las caras que son intersecadas por el segmento de línea que une al vértice que representa al nodo fuente con el vértice que representa al nodo destino [28, 31].

En la literatura [14, 15, 36, 20], algoritmos como el anterior tienen las siguientes dos restricciones: necesitan una subgráfica generadora plana de la gráfica original y suponen que dicha subgráfica se mantiene fija durante el proceso de ruteo. Este proceso es sencillo si se realiza en una subgráfica plana de una gráfica de distancia unitaria. Esto se logra utilizando la *subgráfica de Gabriel* [1], ver figura 4.

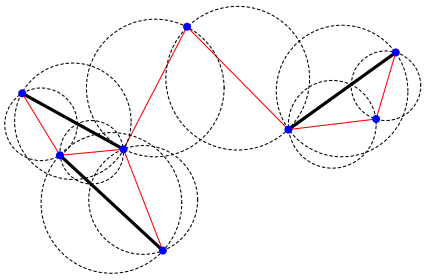


FIGURA 4: Subgráfica de Gabriel de una GDU. Las aristas negras representan aquellas aristas de la GDU que a diferencia de las aristas rojas, no son parte de la subgráfica de Gabriel.

[3] Entendemos por gráficas fijas aquellas que durante la implementación de un algoritmo sus vértices se suponen fijos respecto a su posición.

Otro tipo de algoritmos de ruteo que garantizan la entrega de los mensajes en redes fijas han sido obtenidos al combinar el ruteo por caras y el ruteo glotón [14, 15, 20, 22]. Estos algoritmos funcionan en modo glotón hasta que el mensaje llega a un nodo en donde el ruteo glotón falla. Entonces, el algoritmo cambia a modo ruteo por caras como un mecanismo de recuperación de ruta y vuelve a cambiar a ruteo glotón en cuanto sea posible.

Capítulo 1

Preliminares

Al iniciar el análisis teórico sobre cuál es la mejor manera de mandar mensajes entre los nodos de una red, surge la pregunta de cómo modelar el sistema a considerar. Una manera común de hacerlo es modelar una red por medio de una gráfica tal que sus vértices representan dispositivos y sus aristas conexiones entre éstos. En este estudio suponemos que las redes a utilizar son modeladas por gráficas[1], en particular, por gráficas geométricas.

Definición 1.1 (Gráfica geométrica)**.** Una *gráfica geométrica* $G(V, E)$ es una gráfica dibujada en el plano de tal manera que sus vértices (elementos de V) son representados por puntos y las aristas (elementos de E) son representadas por segmentos de líneas rectas que conectan los puntos correspondientes.

En adelante suponemos que los puntos de las gráficas geométricas con las que trabajamos están en posición general, esto es, no hay tres de ellos alineados.

Una técnica que nos ayudará a hacer más eficiente el modelo para nuestro sistema es el control de la topología. Los parámetros de eficiencia son, por ejemplo, minimizar la energía utilizada o incrementar el rendimiento y la vida útil de la red. El control de la topología consiste en elegir un subconjunto E' de aristas de entre todos los enlaces factibles, tal que la subgráfica inducida por E' sea conexa y que el rendimiento de nuestros algoritmos sea adecuado respecto a dichos parámetros.

El problema del control de la topología es descrito dentro del marco de las redes inalámbricas como: Sea $G(V, E)$ una gráfica geométrica cuyos vértices representan

[1]Suponemos que el lector o la lectora conocen las definiciones básicas de la teoría de gráficas.

estaciones de transmisión. Una arista (u, v) está en E, si el vértice u está dentro del rango de transmisión del vértice v y viceversa. El problema del control de la topología puede ser formulada ahora como: Dada la gráfica $G(V, E)$ y un algoritmo de ruteo, calcular una subgráfica $G'(V, E')$, $E' \subseteq E$, tal que permita la ejecución de dicho algoritmo de forma eficiente.

Basándonos en el modelo seleccionado para la red, podemos asignar *pesos* a las aristas de $G(V, E)$. Esto es, a cada arista $e \in E$ asignamos un valor $w(e)$. Estos pesos son asignados de acuerdo a alguna métrica.

Las métricas más usadas suelen ser la *métrica de enlace*, en la cual todas las aristas tienen peso 1, i.e., es el número de saltos el que es contabilizado; la *métrica euclidiana*, que es la distancia euclidiana d entre dos puntos; y la *métrica de energía*, que considera la distancia euclidiana y la eleva a alguna potencia a mayor que 1, esta potencia es llamada *exponente de atenuación*, el cual es comúnmente un valor entre 2 y 5 [40]. Por lo tanto, dependiendo de la métrica que se use, a la arista $e = (u, v)$ le puede ser asignado uno de los siguientes pesos:

$$w_{enlace}(e) := 1, \qquad w_{dist}(e) := d(u, v), \qquad \text{o} \qquad w_{energia(a)}(e) := d(u, v)^a$$

donde $a > 1$ y w_{enlace}, w_{dist} y $w_{energia(a)}$ denotan los pesos de las aristas para las métricas de enlace, euclidiana y de energía, respectivamente.

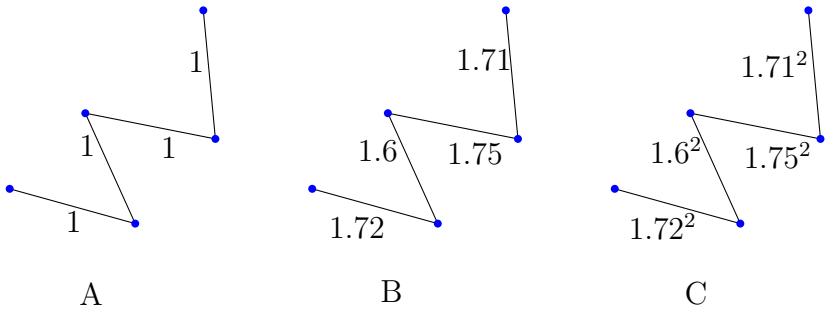


FIGURA 1.1: Las figuras A, B y C muestran las métricas de enlace, euclidiana y de energía con exponente de atenuación 2, respectivamente.

El concepto de métrica que acabamos de ver puede extenderse a cualquier asignación númerica que se le da a una arista y que llamamos *costo*.

Definición 1.2 (Función de costo)**.** Dada una gráfica $G(V, E)$, una *función de costo* $c : E \mapsto \mathbb{R}^+$ es una función no decreciente (en cuanto al peso w de las

aristas) que mapea una arista $e \in E$ a un valor real positivo $c(e)$ tal que para cualesquiera $e, e' \in E$ $w(e') > w(e) \implies c(e') \geq c(e)$.

Así, podemos definir ahora los costos basados en las métricas definidas anteriormente: $c_{enlace}(e) := 1$ como el costo basado en la métrica de enlace, $c_{dist}(e) := d$ el costo basado en la métrica euclidiana y $c_{energia(a)}(e) := d^a$ usualmente con $2 \leq a \leq 5$, el costo basado en la métrica de energía.

Definición 1.3 (Camino)**.** Dada una gráfica $G(V, E)$, una sucesión $p = (u_1, u_2, ..., u_n)$ es un *camino* de G si u_i es un vértice de G para todo $i \in \{1, 2, ..., n\}$ y u_i es adyacente a u_{i+1} para todo $i \in \{1, 2, ..., n-1\}$.

Por conveniencia, definimos el costo de un camino y el costo de un algoritmo. El *costo de un camino* p es definido como la suma de los valores del costo de sus aristas y lo denotamos como $c(p)$. Análogamente, el *costo de un algoritmo* $\mathcal{A}$ es definido como la suma de los costos de los caminos que son recorridos durante la ejecución del algoritmo en una gráfica y lo denotamos como $c(\mathcal{A})$.

1.1. Gráficas de disco unitario

Existen varios problemas que son más sencillos de resolver utilizando gráficas de disco unitario en lugar de gráficas generales, puesto que en las primeras se supone que la comunicación es posible entre dos vértices si y sólo si la distancia euclidiana entre éstos es menor o igual a un rango de transmisión constante. Esto implica que es necesario determinar qué parejas de vértices *pueden* comunicarse entre sí, cuáles vértices *deben* comunicarse y a través de qué caminos para poder ahorrar energía, reducir la interferencia y, en particular para algoritmos de ruteo locales, poder obtener una subgráfica plana.

Un problema específico en las redes de comunicación es que su topología puede cambiar durante la ejecución del algoritmo, debido entre otras cosas, al movimiento de los nodos, fallas, o por enlaces de comunicación inestables. Basándonos en esta observación, concluimos que para nuestro objetivo es más conveniente modelar nuestras redes de comunicación con gráficas de disco unitario.

Definición 1.4 (Gráfica de disco unitario GDU)**.** La gráfica cuyo conjunto de vertices es V y tal que cualesquiera dos de dichos vértices son adyacentes si su distancia es a lo más 1, es llamada la *gráfica de disco unitario de* V.

1.1.1. Propiedades de las GDU

Una de las características principales de las GDU es que cualquier *subgráfica inducida por vértices* es también una GDU, donde definimos dicha subgráfica como:

Definición 1.5. (Subgráfica inducida por vértices). Dada una gráfica G y S un subconjunto de los vértices de G, la *subgráfica inducida por S en G*, denotada como $G[S]$, tiene por conjunto de vértices a S y dos vértices en $G[S]$ son adyacentes si y sólo si son adyacentes en G.

Al respecto, el siguiente lema es un resultado que muestra una propiedad de las GDU respecto a sus gráficas inducidas por vértices de las GDU.

Lema 1.6 ([8, Lema 3.2]). *Sea $G(V, E)$ una gráfica de disco unitario. Entonces G no contiene a $K_{1,6}$ como gráfica inducida por vértices.*

Demostración. Procedamos por contradicción suponiendo que la gráfica G contiene a $K_{1,6}$ como gráfica inducida. Entonces, observemos que $K_{1,6}$ siempre tiene dos aristas cuyo ángulo es menor o igual a 60 grados, lo cual implica que también debe de existir una arista de G entre dos de las aristas de la estrella y, por lo tanto, esta arista también sería inducida por los mismos 7 vértices, ver la figura 1.2. Esto es una contradicción a la hipótesis de que $K_{1,6}$ puede ser inducida por G. □

En busca de simplicidad, suponemos que la *distancia* (costo) de un camino entre cualesquiera dos vértices no puede ser arbitrariamente pequeña:

Definición 1.7 (Modelo $\Omega(1)$)**.** Dada una gráfica $G(V, E)$, decimos que G está restringida al modelo $\Omega(1)$ si la distancia entre cualesquiera dos vértices $u, v \in V$ está acotada por abajo por un término de orden $\Omega(1)$, i.e., si existe una constante positiva d_0 (cota inferior) tal que $d_0 < d(u, v)$ para cualesquiera $u, v \in V$.

Lema 1.8 ([20, Lema 5.1])**.** *Sean $c_1(\cdot)$ y $c_2(\cdot)$ funciones de costo de acuerdo a la definición 1.2 y sea G una gráfica de disco unitario en el modelo $\Omega(1)$. Más aún, sea p un camino en G. Entonces tenemos*

$$c_1(p) \leq \alpha \cdot c_2(p)$$

para una constante α.

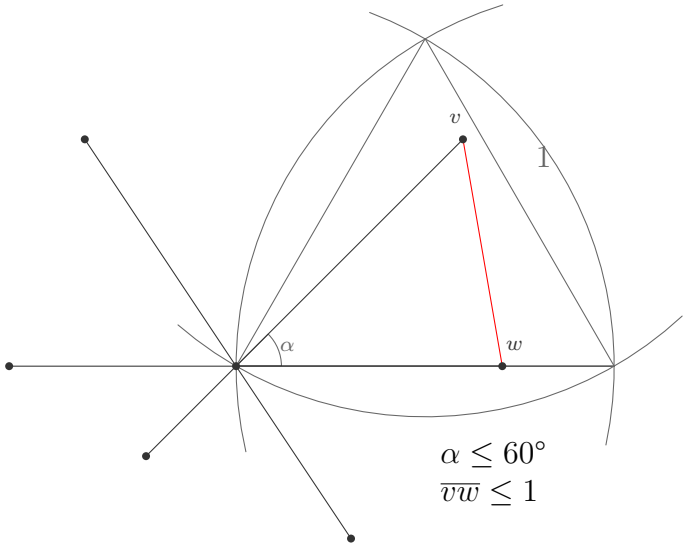


FIGURA 1.2: Una gráfica de disco unitario no puede contener a $K_{1,6}$ como subgráfica inducida por vértices.

Demostración. Supongamos sin pérdida de generalidad que p consiste de k aristas, esto es, $c_{enlace}(p) = k$. Dado que $d_0 \leq c_{dist}(e) \leq 1$ para todas las aristas $e \in E$ y las funciones de costo son no decrecientes, tenemos que $c_1(p) \leq c_1(1) \cdot k$ y $c_2(d_0) \cdot k \leq c_2(p)$. Dado que $c_1(1)$ y $c_2(d_0)$ son constantes mayores que 0, el lema se cumple para $\alpha = \frac{c_1(1)}{c_2(d_0)}$. □

De manera análoga, podemos observar que, en una gráfica, la *distancia de un par de vértices u y v* -definida como el costo del camino más corto entre u y v- difiere sólo por un factor constante entre las distintas métricas inducidas por las funciones de costo:

Lema 1.9 ([20, Lema 5.2])**.** *Sea $G(V, E)$ una gráfica de disco unitario en el modelo $\Omega(1)$, y $s, t \in V$ dos vértices cualesquiera de G. Sean p_1^* y p_2^* caminos óptimos de s a t en G con respecto a las métricas inducidas por las funciones de costo $c_1(\cdot)$ y $c_2(\cdot)$, respectivamente. Entonces se cumple que*

$$c_1(p_2^*) \leq \alpha \cdot c_1(p_1^*) \text{ y } c_1(p_2^*) \geq \beta \cdot c_1(p_1^*)$$

para dos constantes α y β. Esto es, los valores de los costos de los caminos óptimos para las diferentes métricas, difieren solamente por un factor constante.

Demostración. Dado que p_2^* es óptimo, tenemos que

$$c_2(p_2^*) \leq c_2(p_1^*) \quad \text{(a)}$$

Aplicando el lema anterior obtenemos:

$$c_1(p_2^*) \leq \gamma \cdot c_2(p_2^*) \text{ y } c_2(p_1^*) \leq \delta \cdot c_1(p_1^*) \text{ (b)}$$

para dos constantes γ y δ. Combinando las ecuaciones (a) y (b) obtenemos que $c_1(p_2^*) \leq \alpha \cdot c_1(p_1^*)$ para $\alpha = \gamma \cdot \delta$. Más aún, dado que p_1^* es óptimo tenemos que $c_1(p_2^*) \geq c_1(p_1^*)$ y por lo tanto la segunda ecuación del lema se cumple para $\beta = 1$. □

1.1.2. Organización eficiente de la red

Además de la comunicación en las redes inalámbricas, las interferencias durante las transmisiones simultáneas también son de gran importancia, puesto que son un problema común en redes *ad hoc*. En lo que sigue nos interesan los problemas relativos a la creación de subconjuntos de nodos que permitan un ruteo eficiente, en particular, uno con poca interferencia. Para esto, supondremos que los vértices de los subconjuntos ya mencionados están al menos a distancia 2. Las técnicas utilizadas para obtener dichos subconjuntos de nodos están basadas en las siguientes definiciones:

Definición 1.10 (Conjunto dominante mínimo CDM)**.** En una gráfica $G(V, E)$, un conjunto dominante es un subconjunto $S \subseteq V$, tal que cada vértice en V está en S o tiene al menos un vecino en S.

El problema del conjunto dominante mínimo es encontrar un conjunto dominante S de cardinalidad mínima [38].

Notemos que un conjunto dominante mínimo no garantiza la existencia de un camino entre dos vértices (no necesariamente en el conjunto dominante) tal que dicho camino consista estrictamente de vértices dominantes.[2] Por lo tanto, si queremos garantizar el ruteo, necesitamos trabajar con conjuntos dominantes que sean conexos.

[2]En este caso nos referimos a "vértice dominante" como un vértice en el conjunto dominante.

Definición 1.11 (Conjunto dominante mínimo conexo CDMC)**.** En una gráfica $G(V, E)$, un conjunto dominante mínimo conexo S, es un conjunto dominante $S \subseteq V$, que además induce una subgráfica conexa, de cardinalidad mínima posible.

Tanto la versión general, como la versión conexa del problema de encontrar un conjunto dominante de cardinalidad mínima, son NP-duros, incluso para las gráficas de disco unitario. En consecuencia, se ha intentado dar aproximaciones casi óptimas para dichos conjuntos, por ejemplo, el algoritmo glotón simple para gráficas de grado máximo Δ, calcula una aproximación de factor $ln(\Delta)$, es decir, un conjunto dominante de tamaño a lo más $ln(\Delta)$ veces el tamaño del conjunto obtenido con la solución óptima [2, 3].

Si bien el ruteo basado en CDMC garantiza la comunicación entre todos los vértices, cabe la posibilidad de que según las condiciones del control de la topología, necesitemos un conjunto dominante en el cual algún par de vértices de ese conjunto no sean vecinos. Esto nos lleva a la siguiente definición:

Definición 1.12 (Conjunto independiente CI)**.** En una gráfica $G(V, E)$, un conjunto independiente $S \subseteq V$ de G es un subconjunto de vértices tal que $\forall u, v \in S$ u, v no son vecinos en G.

Observemos que los vértices de un conjunto independiente no se interfieren entre sí durante las transmisiones simultáneas, lo cual es ideal pues lo que necesitamos es que dichas transmisiones no causen interferencia. Por lo tanto, la estrategia que se siguió y que explicaremos a detalle más adelante, fue partir de un conjunto dominante y particionarlo en subconjuntos independientes, en particular, en *subconjuntos independientes maximales.*

Definición 1.13 (Conjunto independiente maximal CIM)**.** En una gráfica $G(V, E)$, un conjunto independiente $S \subseteq V$ de G es un subconjunto independiente maximal si todo vértice que no está en S tiene un vecino en S, es decir, si S forma un conjunto dominante.

Notemos que un conjunto independiente maximal, no es lo mismo que un conjunto independiente máximo. Específicamente, en el problema de encontrar un conjunto independiente máximo buscamos encontrar un conjunto independiente de cardinalidad máxima. Sin embargo, en el problema de encontrar un CIM, sólo buscamos un conjunto de vértices mutuamente independientes (es decir que no sean vecinos),

tal que cada vértice que no esté en S, tenga al menos un vecino en S. Ambos problemas difieren en el sentido de que calcular un conjunto independiente máximo es NP-completo, mientras que calcular un CIM puede hacerse en tiempo lineal: recursivamente, elegimos uno a uno los vértices independientes y eliminamos todos sus vértices vecinos.

Si suponemos que la distancia entre dos vértices está acotada inferiormente por una constante, se puede demostrar que el grado maximo Δ de la gráfica de disco unitario está acotado superiormente. Debido a las caracterísitcas de las redes *ad hoc*, este requerimiento en las gráficas de disco unitario es de gran utilidad. Por ejemplo, el ruteo que mostraremos en el capítulo 2 (que es una combinación entre el ruteo por caras y el ruteo glotón), es más eficiente en gráficas de disco unitario con grado máximo acotado que en gráficas de disco unitario generales. [18].

1.2. Gráficas definidas localmente y gráficas de proximidad

En esta sección presentamos algunas gráficas de disco unitario restringidas que son de nuestro interés, conocidas como *gráficas de proximidad*. Éstas son gráficas geométricas tal que sus aristas conectan parejas de vértices que, de alguna manera, están cerca el uno del otro.

Definición 1.14 (Gráfica de proximidad)**.** Dada una propiedad P sobre parejas de puntos en el plano, una gráfica de proximidad G es una gráfica en la que dos vértices u y v son adyacentes si P es satisfecha para u y v.

Es de importancia mencionar que para que los algoritomos presentados en este trabajo sean locales, es necesario suponer en varios casos que estamos trabajando con subgráficas de gráficas de disco unitario. De esta manera aseguramos que dos vértices vecinos no puedan estar abitrariamente lejos.

A continuación enunciaremos algunos ejemplos de subgráficas geométricas de proximidad:

Subgráfica de Gabriel (GG)

Dada una gráfica $G(V, E)$ diremos que una arista $(u, v) \in E$ es una arista de la *subgráfica de Gabriel* de $G(V, E)$, si el disco definido por el diámetro que forman los extremos de dicha arista no contiene más vértices de V.

Formalmente, una arista (w, z) *existe* si no hay vértice p tal que $d(w, p)^2 + d(z, p)^2 \leq d(w, z)^2$, i.e., no existe vértice p tal que (w, p, z) formen un triángulo con un ángulo mayor a $\frac{\pi}{2}$ en p (ver figura 1.3).

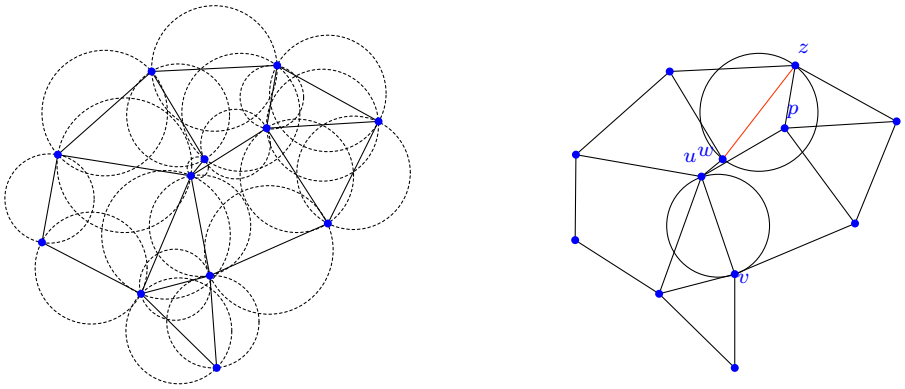


FIGURA 1.3: La figura de la izquierda muestra la subgráfica de Gabriel de la gráfica completa generada por el conjunto de puntos azules. La figura de la derecha muestra que la arista (u, v) está en la subgráfica de Gabriel mientras que (w, z) no, puesto que el punto p está en el disco definido por (w, z).

Subgráfica de vecindad relativa (GVR)

Dada una gráfica $G(V, E)$ diremos que una arista $(u, v) \in E$ es una arista de la *subgráfica de vecindad relativa* de $G(V, E)$ si la intersección de las regiones delimitadas por las circunferencias de radio $d(u, v)$ y con centro en u y v, respectivamente, no contiene ningún otro vértice de la gráfica G. A dicha intersección se le llama la vecindad relativa de la arista (u, v) (ver figura 1.4).

Formalmente, una arista (u, v) existe si:

$$d(u, v) \leq max_{w \in V - \{u,v\}}(d(u, w), d(w, v)).$$

Subgráfica de Yao (GY)

Dada una gráfica $G(V, E)$ y dado un vértice $u \in V$, las aristas de la *subgráfica de Yao* serán aquellas que resultan de unir el vértice u con el vértice más cercano de

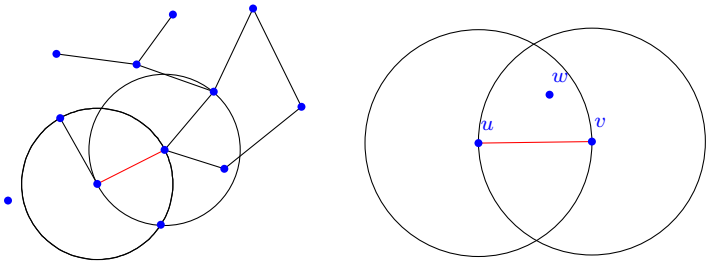


FIGURA 1.4: La figura de la izquierda muestra una arista de la subgráfica de vecindad relativa mientras la figura de la derecha muestra una arista que no está en dicha subgráfica ya que el vértice w está en la vecindad relativa de (u, v).

cada uno de los k sectores en que queda dividido el plano al trazar k semirectas con origen en el punto u y distancia angular $\frac{2\pi}{k}$. Observemos que la subgráfica de Yao obtenida no es necesariamente plana (ver figuras 1.5 y 1.6). Cuando más de un vértice en el sector es el más cercano al vértice u decimos que hay un *empate*[3]. En este caso hacemos un *desempate*[4], i.e., seleccionamos uno de los dos vértices más cercanos a conectar con u.

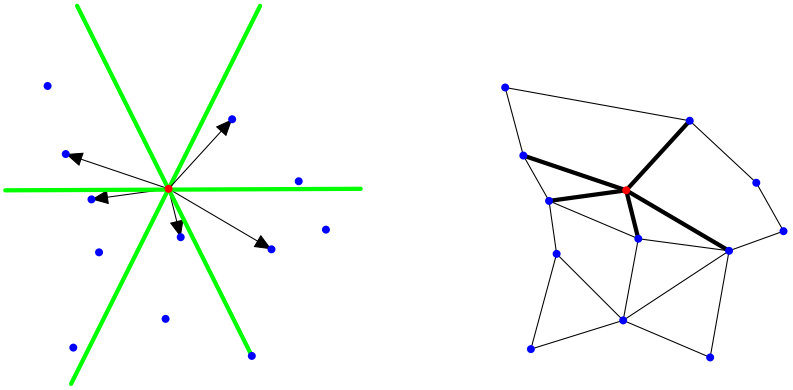


FIGURA 1.5: La figura muestra las aristas de la subgráfica de Yao para el vértice rojo.

[3]Decimos que hay un *empate* cuando dos vértices o más cumplen las condiciones de elección del algoritmo.

[4]Le llamamos *desempate* a la elección que se hace entre los vértices (comúnmente con identificadores únicos para cada vértice) que están empatados.

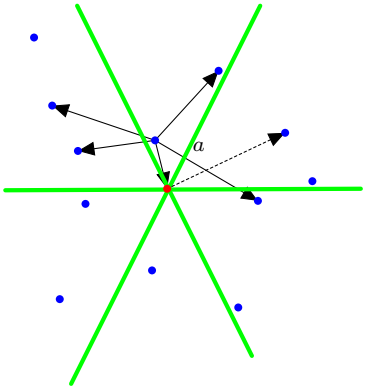


FIGURA 1.6: La figura muestra cómo al obtener las aristas de la subgráfica de Yao para el vértice rojo, la arista discontinua interseca a la arista a (figura anterior), lo cual implica que la subgráfica de Yao no es plana necesariamente.

Subgráfica de cobertura por conos (GCC)

Dada una gráfica $G(V, E)$ y dado un vértice $u \in V$, consideremos a todos los vecinos de u, ordenados de manera ascendente según su distancia con éste. La elección de las aristas de la *subgráfica de cobertura por conos* se realiza seleccionando a (u, v) como la primer arista de la subgráfica (donde v es el vecino más cercano a u) y considerando un cono (sector) de ángulo α, llamado α-cono, de manera que la arista (u, v) sea su bisector. Se continúa eligiendo aristas adyacentes a los vecinos de u que no han sido cubiertos por los conos, hasta que el sector angular de u quede cubierto por los α-conos elegidos o no hay más puntos sin cubrir por los α-conos (ver figura 1.7).

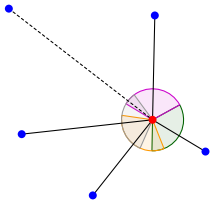


FIGURA 1.7: La figura muestra las aristas con un extremo en el vértice rojo y los α conos, con $\alpha = 100°$ para los 4 vecinos más cercanos a dicho vértice. Observemos que la línea punteada ya no es una arista de la gráfica puesto que el rango angular del vértice rojo ha sido cubierto por sus primeros 4 vecinos más cercanos.

1.2.1. Otras gráficas de interés

A continuación, definiremos otras gráficas geométricas que son de nuestro interés y que no son de proximidad.

Triangulación de Delaunay (TD) y sus variantes.

Dado un conjunto de vértices $V = \{u_1, u_2, ..., u_n\}$, una arista $(u_i, u_j) \in V \times V$ pertenece a la *triangulación de Delaunay*, si existe una circunferencia que pase por ambos vértices y que no contenga en su interior algún otro vértice de V (ver figura 1.8).

Es bien sabido que a partir de esta propiedad, se deduce que dichas aristas forman una triangulación, la cual tiene como característica que el disco que circunscribe a cada uno de sus triángulos no contiene ningún otro vértice de V [23]. A estos triángulos los llamamos *triángulos de Delaunay* y los denotamos por (u, v, w) donde u, v y w son sus vértices. La triangulación de Delaunay es una gráfica conexa y plana [42].

Notemos que los vértices de alguna de las aristas de algún triángulo podrían estar arbitrariamente lejos, y por lo tanto, el área determinada por el circuncírculo del triángulo puede ser arbitrariamente grande. Esto implica que la triangulación de Delaunay no se puede construir localmente, puesto que podríamos estar en la situación en la que tengamos que saber información de vértices que estén arbitrariamente lejos y que sean necesarios para determinar si un triángulo es de Delaunay. Por esta razón, versiones localizadas de dicha triangulación han tenido que ser definidas: la triangulación de Delaunay *k-localizada*, la triangulación de Delaunay restringida y la triangulación de Delaunay parcial. Por motivos de interés de este trabajo, sólo describiremos la triangulación de Delaunay k-localizada.

La triangulación de Delaunay k-localizada está definida por los triángulos que cumplen que cada uno está circunscrito por un disco no contiene en su interior ningún vértice de las *k-vecindades* (vecinos a distancia a lo más k saltos) de los vértices de (u, v, w). La triangulación de Delaunay k-localizada contiene a la triangulación de Delaunay y, al igual que ésta, su grado máximo puede llegar a ser incluso $n-1$.

Árbol generador de peso mínimo (AGPM).

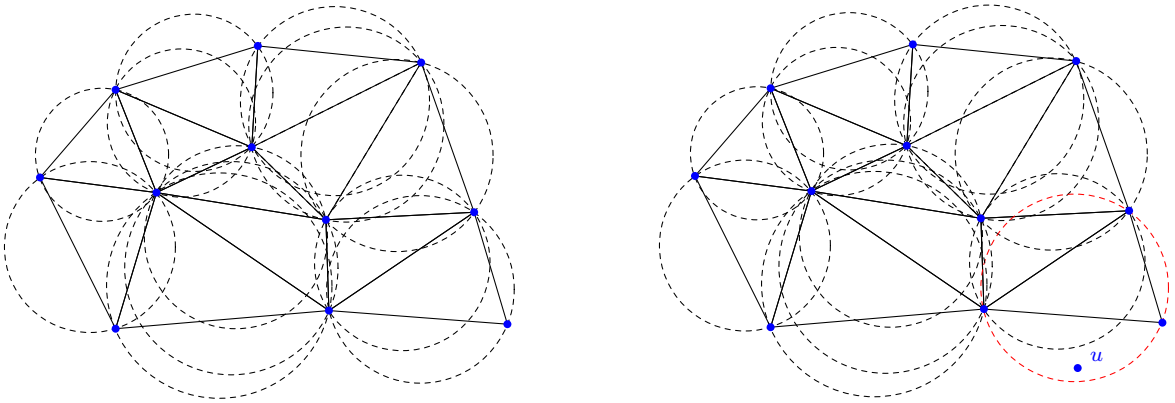


FIGURA 1.8: La gráfica en el lado izquierdo muestra la triangulación de Delaunay para el conjunto de vértices V. De lado derecho observamos que si el punto u fuera parte del conjunto de vértices inicial, el triángulo con circuncírculo rojo no podría ser parte de la triangulación.

Dada una gráfica $G(V, E)$ con peso asignado a sus aristas, un *árbol generador de peso mínimo* de G es una subgráfica conexa $S(V', E')$ de $G(V, E)$ con $V' = V$. Esta subgráfica no tiene ciclos, tiene $|V| - 1$ aristas y la suma de los pesos de éstas es la mínima posible. Una gráfica G puede tener más de un AGPM (ver figura 1.9).

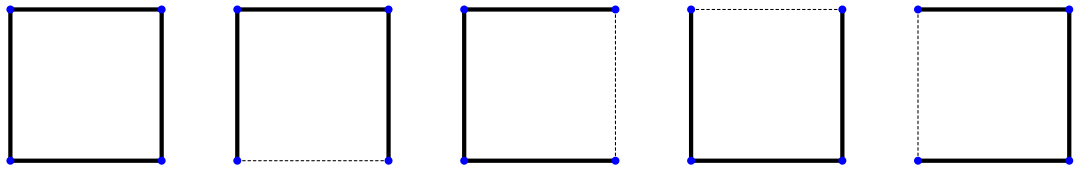


FIGURA 1.9: La figura muestra que para la gráfica de lado extremo izquierdo con la métrica de enlace, existen 4 diferentes AGPM con mismo peso.

Se sabe que con la métrica euclidiana el árbol generador de peso mínimo de una gráfica es una subgráfica de la triangulación de Delaunay [12] (ver figura 1.10).

Gráfica del vecino más cercano (GVC).

Dada una familia de puntos $V = \{u_1, u_2, ..., u_n\}$, las aristas de la *gráfica* del vecino más cercano son aquellas que resultan de unir para cada vértice u_i con

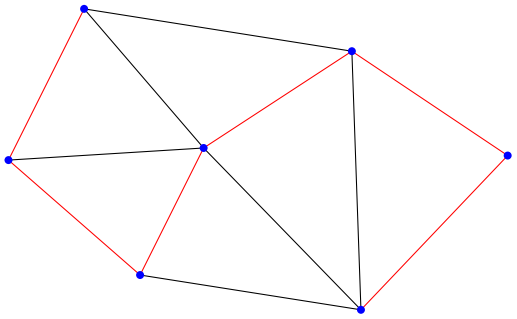


FIGURA 1.10: Árbol generador de peso mínimo como subgráfica de la triangulación de Delaunay.

$i \in \{1, 2, ..., n\}$, a éste con el punto más cercano según la distancia euclidiana (ver figura 1.12). La gráfica del vecino más cercano es una gráfica dirigida y está contenida en un árbol generador de peso mínimo. Podemos definir una variante de esta gráfica como la gráfica del k-vecino más cercano en la cual el vértice en cuestión es conectado a sus k-vecinos más cercanos. En general, estas gráficas no son conexas, por lo tanto, no suelen ser usadas para modelar problemas de ruteo (ver figura 1.11).

FIGURA 1.11: Los dos lados cortos del rectángulo son las aristas de la subgráfica del vecino más cercano, la cual no es conexa.

1.2.2. Variaciones de gráficas de disco unitario

Además, como veremos a detalle más adelante, algunos algoritmos de ruteo como el ruteo por caras requieren subgráficas planas. Para esto, hacemos usa de la gráfica de Gabriel de un conjunto de vértices, definida previamente, la cual es una gráfica plana que puede ser calculada por un algoritmo sencillo, generealmente en

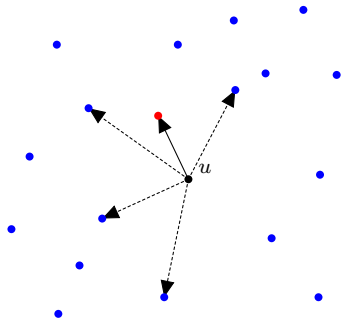


FIGURA 1.12: El vértice rojo es el vértice más cercano al punto u. Por lo tanto, la arista que sale de u al punto rojo es una arista de la subgráfica del vecino más cercano.

tiempo $O(nlogn)$. La relación entre estas gráficas la podemos ver representada en la figura 1.13 y son mostradas en orden de inclusión. Dichas gráficas son de gran importancia para el control de la topología y pueden ser vistas con más detalle en [40, Subsección 5.3].

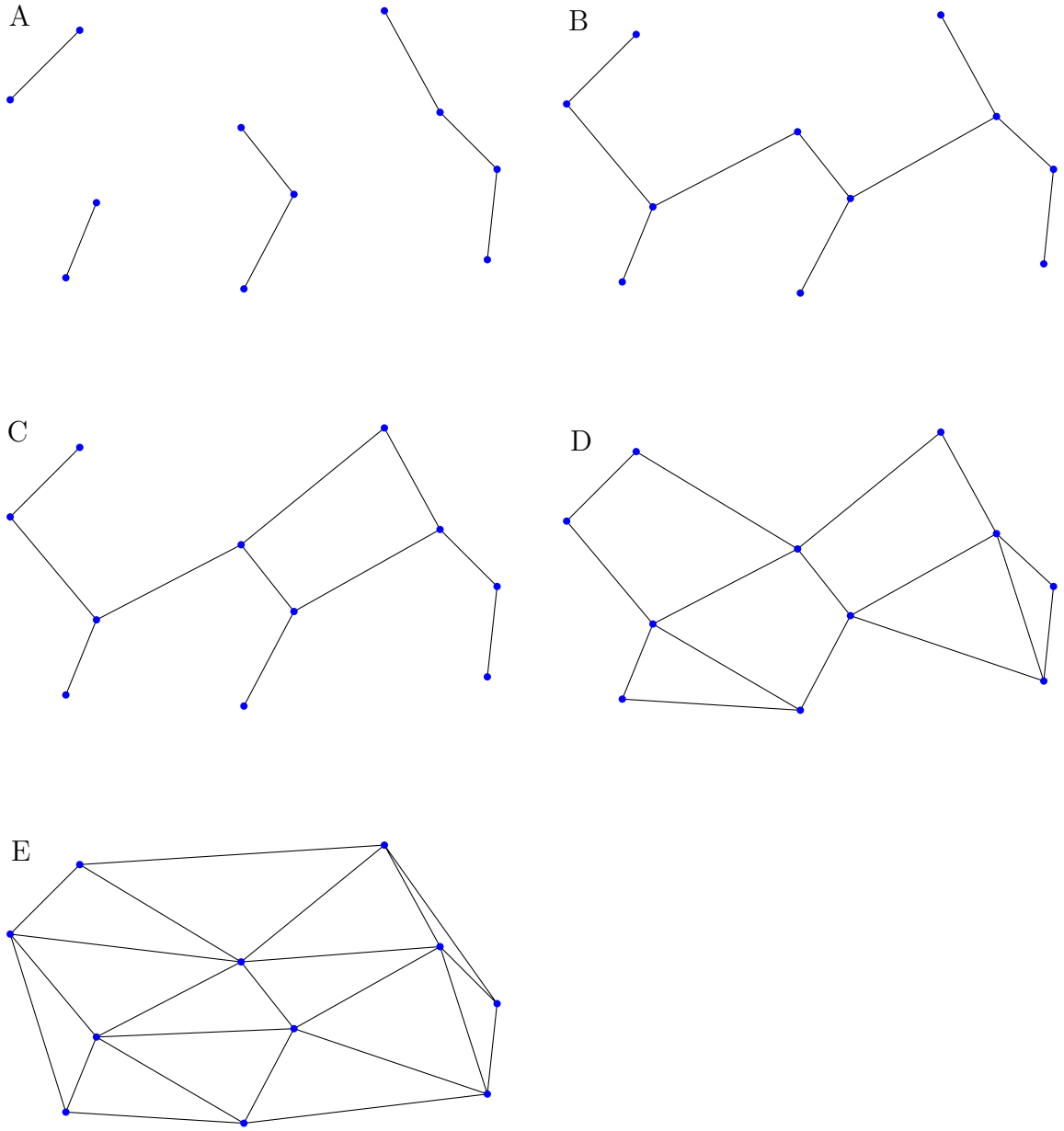


FIGURA 1.13: Ejemplos de gráficas geométricas en orden de inclusión.

Capítulo 2

Problemas fundamentales

Como hemos mencionado anteriormente, en ausencia de una infraestructura fija, los nodos intermediarios de la red tienen que funcionar como enrutadores. Debido a que la topología de la red cambia constantemente, los algoritmos de ruteo para redes *ad hoc* se diferencian significativamente de los esquemas estándar de ruteo para redes alámbricas. Una manera efectiva de mejorar el desempeño de los algoritmos de ruteo es agrupar el conjunto de nodos en *cúmulos* (del inglés *clusters*). El ruteo es entonces llevado a cabo por representantes de cúmulo que actúan como enrutadores. Todos los demás nodos se comunicarán vía un nodo adyacente que es representante de su cúmulo.

¿Cuál sería un buen agrupamiento? La respuesta a esta pregunta variará dependiendo del problema en el que se quiera trabajar. Sin embargo, tomando en cuenta la naturaleza inalámbrica de las redes *ad hoc* y su comunicación multi-saltos[1] entre nodos, todo buen agrupamiento debería satisfacer al menos la siguiente propiedad: con el objetivo de lograr una comunicación eficiente entre cada par de nodos, cada nodo debe ser adyacente a al menos un representante del cúmulo al que pertenece. Esto implica que el conjunto de representantes de cúmulo debe de formar un conjunto dominante (de preferencia un conjunto dominante mínimo).

La definición de conjuntos dominantes deja abierta la posibilidad de que dos vértices que sean representantes de cúmulo puedan ser vecinos. De hecho, una buena solución al problema de encontrar un conjunto dominante de cardinalidad mínima puede requerir que dos vértices vecinos se conviertan en representantes de cúmulo,

[1] Del inglés *multi-hop* que nos indica que la comunicación entre dos nodos es a través de pequeños *saltos*, es decir, entre nodos intermediarios adyacentes a corta distancia.

como lo muestra la figura 2.1. Sin embargo, considerando la naturaleza inalámbrica de la comunicación en redes *ad hoc*, sería inconveniente que las gráficas que las modelan tengan vértices vecinos que sean representantes de cúmulo, puesto que habría interferencia entre ellos. La tarea de definir una estructura eficiente de cúmulos es facilitada si no existen dos vértices que sean representantes de cúmulo y que se encuentren en el rango de transmisión uno del otro. Este requisito adicional nos lleva naturalmente al uso de conjuntos independientes maximales.

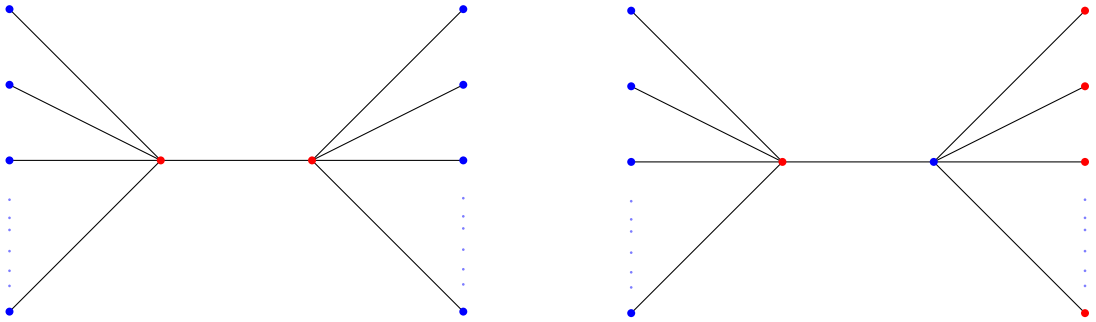


FIGURA 2.1: Gráfica con un conjunto dominante mínimo conexo de tamaño 2 (izquierda) y con un conjunto independiente maximal de tamaño $\frac{n}{2}$ (derecha).

Hay que notar que no todos los algoritmos distribuidos son igualmente adecuados para aplicarlos en redes *ad hoc*. En particular, aquellos algoritmos que están basados en sostener una visión global de la red tienden a consumir mucha energía y tiempo, y por lo tanto, son difícilmente acoplables a las redes en cuestión. En otras palabras, los algoritmos que tienen un tiempo de ejecución que puede crecer linealmente conforme aumenta el número de nodos no son aptos para usarse en redes *ad hoc* a gran escala [36].

En su lugar, estamos interesados en los algoritmos que ya hemos definido como algoritmos locales. Es de gran interés, tanto teórico como práctico, investigar la relación que hay entre la cantidad de comunicación[2] (complejidad de tiempo) y la calidad de la solución global que se obtenga (factor de aproximación). En otras palabras, queremos encontrar cuánta comunicación y tiempo son necesarios en redes *ad hoc* para conseguir un ruteo adecuado.

Llamaremos *modelo GDU* a la gráfica de disco unitario que representa a una red de comunciación dada. En este trabajo, usaremos el modelo GDU para los

[2]Decimos *cantidad de comunicación* para referirnos a la cantidad de rondas de comunicación que ejecuta un algoritmo.

problemas a tratar puesto que frecuentemente se supone que la distribución física de los nodos representados por los vértices de una GDU, sigue un patrón específico. Particularmente, se supone que los vértices están distribuidos de manera uniforme e independiente en el plano.

2.1. Algoritmos locales para obtener un CIM en gráficas GDU

Esta sección está basada en el capítulo 3 de [40].

El problema de encontrar un conjunto dominante mínimo en gráficas generales es NP-duro. Ha sido demostrado en [9] que para este problema el mejor factor de aproximación posible es $ln(\Delta)$, donde Δ es el grado máximo de la gráfica. En gráficas de disco unitario, el problema (así como su contraparte conexa) permanece NP-duro, pero se vuelven posibles ciertas aproximaciones de factor constante. En dichas gráficas y en algunas de sus generalizaciones, el problema puede permitir incluso esquemas de aproximación de tiempo polinomial aún sin representación geométrica [33].

El algoritmo básico de aproximación a un conjunto dominante mínimo puede ser descrito de la siguiente manera. Asignamos aleatoriamente para cada vértice u un identificador i del conjunto $[1, 2, ..., n^2]$[3]. Cada vértice u_i elige como vértice dominante a su vecino con el identificador más grande (notemos que un vértice se puede elegir a sí mismo). Todos los puntos que hayan sido elegidos al menos una vez, forman un CDM.

Dada una gráfica $G(V, E)$ con n vértices, definimos el siguiente algoritmo para aproximar un CDM.

Para el caso de algoritmos distribuidos, la complejidad de calcular un conjunto dominante eficiente depende de lo que se suponga respecto al modelo en cuestión. Para dar un ejemplo de esto vamos a modificar un poco nuestro modelo, de manera que en lugar de discos unitarios tendremos cuadrados unitarios: para cualesquiera

[3]Es usual que los identificadores se elijan de un conjunto de cardinalidad cuadrática, pues es suficiente esta cota para reducir la probabilidad de que dos vértices tengan el mismo identificador. Sin embargo, si esto llega a suceder, se vuelve a hacer la elección aleatoria hasta asegurar que cada vértice tiene un identificador distinto.

Algoritmo 1 Algoritmo de aproximación CDM de una ronda ([16], Sección 3.1)

1: Cada vértice u selecciona aleatoriamente un identificador ID del intervalo $[1, 2, ..., n^2]$.
2: Cada vértice u intercambia su ID con cada uno de sus vecinos v.
3: Cada vértice u elige de su vecindad al vértice v con el ID más grande.
4: Todos los vértices que han sido elegidos al menos una vez se convierten en vértices dominantes.

puntos en una gráfica geométrica p y q, decimos que p está en la vecindad de q si está en el cuadrado con lados de longitud 1 y centro en q.

Así, procedemos a dar un ejemplo que nos permita analizar el algoritmo. Consideremos una gráfica geométrica $G(V, E)$ y para cada $v \in V$ un subcuadrado S de lados de longitud $1/2$ con centro en v. Para cada subcuadrado S, definimos la región de visibilidad de S, L, como los puntos del plano que están en la unión de los cuadrados con lados de longitud 1 y centro en s, para cada $s \in S$. Observemos que L es un cuadrado con lados de longitud $\frac{3}{2}$.

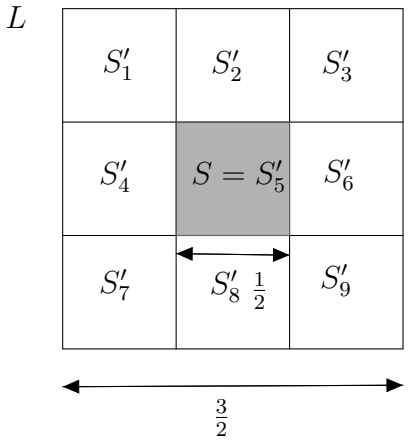


FIGURA 2.2: Rango de visibilidad L de un subcuadrado S.

Llamemos L_v a la región de visibilidad de cada $v \in V$. Observemos que para la gráfica $G(V, E)$, $V \subset \bigcup_{v \in V} L_v$. Además, sea $G' = G \cap \bigcup_{v \in V} L_v$ y para cada L_v, definamos $L'_v = G \cap L_v$.

Demostremos lo siguiente:

Lema 2.1 ([16, Lema 3.4]). *Sea L'_v el rango de visibilidad de un subcuadrado S para un vértice $v \in V$ y sea $m \in \mathbb{N}$ tal que $|L'_v| \leq m$. Entonces el número esperado de vértices dominantes seleccionados por el algoritmo en S es $O(\sqrt{m})$.*

Demostración. Consideremos sólo los puntos en L'_v. Observemos que es suficiente con acotar el número esperado de vértices dominantes en S elegidos por puntos en cada uno de los subcuadrados S'_i con $1 \leq i \leq 9$ de L_v. Sea S'_i para algún i en $1 \leq i \leq 9$. Si $S'_i = S$ entonces sólo un vértice es elegido en este subcuadrado puesto que todos los puntos son mutuamente visibles. De otra manera, supongamos que $|S \cap G| = a$ y $|S'_i \cap G| = b$. Un vértice $u \in S$ es elegido por un vértice $v \in S'_i$ si u es el vecino de v con identificador mayor. Así, como todos los vértices en S'_i son vecinos de v, u tiene identificador mayor que dichos vértices. Por lo tanto, la probabilidad de que u sea elegido es a lo más $\frac{1}{1+b}$ y así, existen a lo más $\frac{a}{1+b}$ vértices elegidos en S por S'_i. Por otro lado, tenemos que como hay b vértices en S', entonces hay a lo más b vértices elegidos por vértices en S'. Así, existe un conjunto que domina a los vértices de S'_i de tamaño a lo más $min(b, \frac{a}{1+b}) \leq \sqrt{a+b+1} - 1 < \sqrt{m}$ [4]. Sumando esto para cada uno de los 9 subcuadrados tenemos que el número de vértices dominantes está acotado por $O(\sqrt{m})$. □

Este lema nos permite demostrar lo siguiente:

Teorema 2.2 ([16, Teorema 3.5]). *Para un conjunto de n puntos, el algoritmo 1 calcula un conjunto dominante de cardinalidad esperada a lo más $O(\sqrt{n}) \cdot k$ veces más grande que el que nos da el caso óptimo.*

Demostración. Consideremos un CDM con vértices $u_1, u_2, ..., u_k$. Para cada u_i con $i \in \{1, 2, ..., k\}$, consideremos su región de visibilidad (cuadrado de longitud 1 con u_k como centro) L_i. Consideremos los 4 subcuadrados de cada L_i de longitud $\frac{1}{2}$ y apliquemos a cada uno de ellos el lema 2.1. Como todos los n vértices están contenidos en la unión de las regiones de visibilidad de los u_i, entonces el valor esperado del número de vértices selciionados en total es $O(\sqrt{n}) \cdot k$, lo cual demuestra el enunciado del teorema. □

[4] Se puede demostrar que la primer igualdad se cumple para números a, b enteros positivos.

2.1.1. CDM a través de un CIM

El algoritmo presentado en la sección anterior tiene el inconveniente de tener mal desempeño en el peor caso. Con el objetivo de encontrar mejores soluciones, en esta sección abordaremos el problema ya mencionado desde una perspectiva algorítmica distinta, específicamente, usaremos un CIM como una aproximación a un conjunto dominante mínimo en una gráfica de disco unitario. Como hemos visto en la figura 2.1, la cardinalidad de un CIM en una gráfica general podría ser arbitrariamente grande en comparación con la cardinalidad de un conjunto dominante mínimo. De esto se sigue que, en general, no existe un algoritmo de aproximación para el problema de encontrar un conjunto dominante mínimo que contenga siempre un CIM. Además, observemos que en gráficas GDU es imposible tener conjuntos densos[5] de vértices que no se conviertan en vecinos. Esta intuición es formalizada en los siguientes lemas, puesto que muestran que un CIM es una aproximación constante al problema de encontrar un conjunto dominante mínimo en gráficas de disco unitario [8, 43].

Lema 2.3. *Sea $G(V, E)$ una gráfica de disco unitario, OPT un conjunto dominante mínimo e I un conjunto independiente maximal. Entonces, I es un conjunto dominante con $|I| \leq 5|OPT|$.*

Demostración. Demostraremos que un vértice v en OPT domina, a lo más, a 5 vértices en I. Notemos que si $v \in I$, entonces v no domina a ningún otro vértice en I. Supongamos entonces que v no está en I y que v domina a $v_1, v_2, ..., v_6 \in I$. Entonces v y los v_i con $i \in \{1, 2, 3, 4, 5, 6\}$ forman la subgráfica inducida $K_{1,6}$ (pues ningún par de vértices en I está conectado), lo cual contradice al lema 1.6. Puesto que OPT domina a V y por lo tanto a I, sólo pueden haber 5 veces más vértices en I que en OPT. □

Lema 2.4 ([8, Teorema 4.8])**.** *Sea $G(V, E)$ una gráfica de disco unitario y sea OPT el conjunto dominante mínimo óptimo conexo en G. El tamaño de cualquier conjunto independiente S en G es a lo más $4 \cdot |OPT| + 1$.*

Demostración. Sea U un conjunto independiente de V y sea T un árbol generador del conjunto dominante mínimo conexo OPT. Sea T el recorrido pre-orden dado por $v_1, ..., v_{|OPT|}$. Sea U_1 el conjunto de vértices en U que son adyacentes a v_1 en

[5] Sea $B \subseteq A$, decimos que B es denso en A si para todo $a \in A$ y para todo $\epsilon > 0$, existe $b \in B$ tal que $d(a, b) < \epsilon$.

T. Para cualquier i tal que $2 \leq i \leq |OPT|$, sea U_i el conjunto de vértices en U que son adyacentes a v_i en T, pero no a $v_1, ..., v_{i-1}$. Entonces $U_1, U_2, ..., U_{|OPT|}$ forman una partición de U. Como v_1 es adyacente a lo más a 5 vértices independientes, $|U_1| \leq 5$. Como $G(V, E)$ es conexa, entonces para $2 \leq i \leq |OPT|$, al menos un vértice en $v_1, ..., v_{i-1}$ es adyacente a v_i. Por lo tanto, U_i está en un sector de a lo más 240 grados dentro del sector angular de v_i (ver figura 2.3). Esto implica que $|U_i| \leq 4$ y por lo tanto

$$|U| = \sum_{i=1}^{|OPT|} |U_i| \leq 5 + 4(|OPT| - 1) = 4|OPT| + 1.$$

□

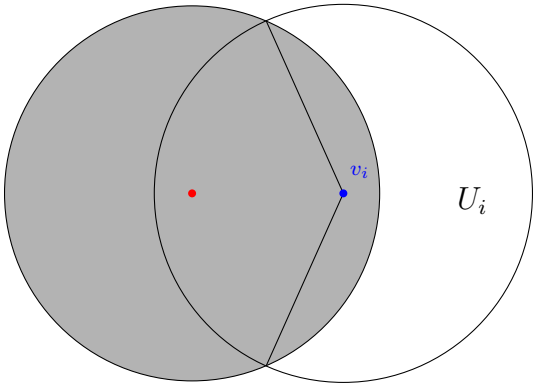


FIGURA 2.3: U_i está en un sector de a lo más 240 grados dentro del sector angular de v_i.

Dados los lemas 2.3 y 2.4 podemos presentar un algoritmo de aproximación constante para el problema de encontrar un conjunto dominante mínimo en la gráfica de disco unitario, usando un conjunto independiente maximal. Como hemos mencionado anteriormente, suponemos que cada vértice tiene un único identificador, así, la forma más sencilla de obtener un CIM es el siguiente algoritmo.

Observemos que el algoritmo anterior es síncrono, es decir, puede ser ejecutado en rondas ordenadas y de duración acotada. Este procedimiento genera un CIM correcto. Sin embargo, observemos que, en el peor de los casos, la complejidad del tiempo y de los mensajes es $\Omega(n)$ y $\Omega(|E|)$, respectivamente, puesto que podría pasar que sólo haya actividad en un vértice a la vez. Por ejemplo, consideremos

Algoritmo 2 Algoritmo para encontrar un CIM ([8] p.11)

1: Se elije un vértice aleatorio u de manera que éste se une al CIM y por lo tanto sus vecinos se eliminan.
2: Cada vértice $v \in V$ con $v \neq u$ que tenga un ID espera hasta que una de las siguientes dos condiciones se cumpla.
3: v tiene un vecino que se une al CIM $\implies$ v se elimina.
4: v tiene el ID más grande de entre todos los vecinos que no han sido cubiertos $\implies$ v se une al CIM.
5: El algoritmo se detiene cuando ya no hay vértices qué procesar.

n vértices colocados en una línea[6] y con indentificadores monótonamente decrecientes. En este caso, el vértice que se encuentra en la posición extrema derecha, tendrá que esperar $n-1$ intervalos de tiempo antes de poder decidir si se une o no al CIM.

2.1.2. Algoritmo eficiente para un CIM

El siguiente algoritmo presenta una solución elegante y mucho más rápida al problema de encontrar un conjunto independiente maximal. Este algoritmo clásico, dado por Luby, [6] genera un CIM en tiempo $O(log(n))$ con alta probabilidad (en lugar de tiempo $\Omega(n)$ como el algoritmo anterior). El algoritmo opera en rondas sincrónicas agrupadas en fases de tres pasos cada una. Al inicio de cada fase, todo vértice informa a sus vecinos si su estado es activo o pasivo (es decir, si está o no en el CIM). De manera similar, el segundo y tercer paso pueden ser implementados usando una ronda sencilla de comunicación.

Denotamos por N_v a la vecindad de v y a N_v^+ a la vecindad de v incluyéndose a sí mismo.

Claramente, el algoritmo es correcto puesto que dos vértices vecinos nunca estarán al mismo tiempo en el CIM. Además, si existe un vértice que no tenga vecinos en el CIM, tarde o temprano se marcará a él mismo y por lo tanto se unirá al CIM. A lo largo del algoritmo, el grado[8] de un vértice $gr(v)^+$ es el número de sus vecinos que están activos.

[6] Este ejemplo aplica sólo para dimensión 1 puesto que en el plano pedimos que todos los puntos estén en posición general.
[7] Entendemos que un vértice se *marca* cuando agrega información para compartir, además de su identificador.
[8] Definimos el grado de un vértice u como $gr(u)$.

Algoritmo 3 Algoritmo rápido para encontrar un CIM ([6], p.3)

1: El código es ejecutado por cada vértice v, el cual está activo inicialmente.
2: **while** v está activo **do**
3: v se *marca* [7] a sí mismo con probabilidad $\frac{1}{2gr(v)^+}$, donde $gr(v)$ es el grado actual de v.
4: Si no existe vecino de v con grado mayor o igual que éste y que también esté marcado, v se une al CIM.
5: Si existe un vértice $u \in N_v^+$ que se ha unido al CIM, entonces v se vuelve pasivo.
6: **end while**

Demostraremos ahora que el tiempo de ejecución del Algoritmo 3 es $O(log(n))$. El primer lema muestra una cota inferior para la probabilidad (P) con la que un vértice se une al CIM en una fase dada.

Lema 2.5 ([6, Lema 1])**.** *Un vértice v se une al CIM en el paso 4 de alguna ronda arbitraria con probabilidad al menos $\frac{1}{4gr(v)^+}$.*

Demostración. Sea M un conjunto de vértices marcados después del paso 3 de la ronda dada. $H(v)$ es el conjunto de vecinos de v que tienen un grado actual mayor que v o mismo grado pero identificador mayor. La probabilidad de que v no se una al CIM a pesar de haberse marcado a sí mismo en el paso 3 es

$$\begin{aligned} P[v \notin CIM | v \in M] &= P[\exists w \in H(v), w \in M | v \in M] \\ &\leq \sum_{w \in H(v)} P[w \in M] = \sum_{w \in H(v)} \frac{1}{2gr(w)^+} \\ &\leq \sum_{w \in H(v)} \frac{1}{2gr(v)^+} \leq \frac{gr(v)^+}{2gr(v)^+} = \frac{1}{2}, \end{aligned} \tag{2.1}$$

para todo vértice w tal que $gr(v)^+ \leq gr(w)^+$

Por lo tanto, la probabilidad con la que v se une al CIM en el paso 4 está acotada por abajo por

$$P[v \in CIM] = P[v \in CIM | v \in M] \cdot P[v \in M] \geq \frac{1}{2} \cdot \frac{1}{2gr(v)^+} = \frac{1}{4gr(v)^+}. \tag{2.2}$$

□

Vamos a llamar *bueno* a un vértice, si la suma de las probabilidades de unirse al CIM de los vértices marcados en su vecindad es más grande que una cierta constante. Formalmente, un vértice v es llamado bueno, si $\sum_{w \in N_v} \frac{1}{2gr(w)^+} \geq \frac{1}{6}$. Intuitivamente, los vértices buenos tienen muchos vecinos de grado bajo y, por lo tanto, la probabilidad de que alguno de ellos se una a un CIM es alta. Esta idea se formaliza en el siguiente lema:

Lema 2.6 ([40, Lema 3.3.5])**.** *La probabilidad de que un vértice bueno sea eliminado en el paso 5 del algoritmo 3 es al menos* $\frac{1}{36}$.

Demostración. Sea v un vértice bueno. Notemos que hay dos casos y empecemos con el más simple.

Caso 1: existe un vecino $w \in N_v$ con grado $gr(w)^+ \leq 2$. Por el lema 2.5 w se une al CIM en el paso 4 (y v es eliminado en el paso 5) con probabilidad de al menos $\frac{1}{8}$.

Caso 2: Todos los vecinos de v tienen grado mayor o igual que 3. Para cualquier vecino $w \in N_v$, tenemos $\frac{1}{2gr(w)^+} \leq \frac{1}{6}$. Como por definición de vértice bueno $\sum_{w \in N_v} \frac{1}{2gr(w)^+} \geq \frac{1}{6}$, entonces existe un subconjunto no vacío de vecinos $X \subseteq N_v$ con $|X| \geq 3$, tal que

$$\frac{1}{6} \leq \sum_{w \in X} \frac{1}{2gr(w)^+} \leq \frac{1}{3}. \tag{2.3}$$

Observemos que la última aseveración se cumple puesto que si no existiera tal conjunto entonces todo subconjunto no vacío de N_v cumpliría que $\sum_{w \in X} \frac{1}{2gr(w)^+} > \frac{1}{6}$, y por lo tanto, $\frac{1}{2gr(w)^+} > \frac{1}{6}$, lo cual contradice que todos los vecinos de v tienen grado mayor o igual que 3.

Podemos ahora acotar por abajo la probabilidad de que el vértice v sea eliminado. De nuevo, si alguno de los vecinos de v se une al CIM en el paso 2, v es eliminado en el paso 3. Por lo tanto tenemos:

$$\begin{aligned} P[\text{v es eliminado}] &\geq P[\exists u \in X, u \in CIM] \\ &\geq \sum_{u \in X} P[u \in CIM] - \sum_{u,w \in X, u \neq w} P[u \in CIM \wedge w \in CIM]. \end{aligned} \tag{2.4}$$

Observemos que la última desigualdad se cumple por el principio de inclusión-exclusión [9]

Recordemos que M es el conjunto de vértices marcados después del paso 1, entonces:

$$\begin{aligned} P[\text{v es eliminado}] &\geq \sum_{u \in X} P[u \in CIM] - \sum_{u,w \in X, u \neq w} P[u \in CIM \wedge w \in CIM] \\ &\geq \sum_{u \in X} P[u \in CIM] - \sum_{u \in X} \sum_{w \in X u \neq w} P[u \in M] \cdot P[w \in M] \\ &\geq \sum_{u \in X} \frac{1}{4gr(u)^+} - \sum_{u \in X} \sum_{w \in X u \neq w} \frac{1}{2gr(u)^+} \cdot \frac{1}{2gr(w)^+} \\ &\geq \sum_{u \in X} \frac{1}{2gr(u)^+} \left(\frac{1}{2} - \sum_{w \in X u \neq w} \frac{1}{2gr(w)^+} \right) \\ &\geq \sum_{u \in X} \frac{1}{2gr(u)^+} \left(\frac{1}{2} - \left(\sum_{u \in X} \frac{1}{2 \cdot gr(u)^+} - \frac{1}{6} \right) \right) \\ &\geq \frac{1}{6} \left(\frac{1}{2} - \frac{1}{3} \right) = \frac{1}{36}. \end{aligned} \tag{2.5}$$

Observemos que esta última desigualdad se debe al hecho de que $\sum_{u \in X} \frac{1}{2gr(u)^+} > \frac{1}{2}$, ya que $|X| \geq 3$. □

Sería ideal si pudieran haber muchos vértices buenos en cada fase. Particularmente, si una fracción constante de los vértices en cada fase son buenos, el tiempo de ejecución es logarítmico. Desafortunadamente, este no es el caso. Por ejemplo, en una gráfica estrella sólo el vértice del centro es bueno. Sin embargo, la situación mejora si consideramos las aristas. En particular, diremos que una arista es *buena* si exactamente uno de sus extremos es un vértice bueno.

[9] En probabilidad, este principio nos dice que para sucesos A_1, A_2 se cumple que $P[A_1 \cup A_2] = P[A_1] + P[A_2] - P[A_1 \cap A_2]$.

Lema 2.7 ([40, Lema 3.3.6]). *En cualquier momento de la ejecución del algoritmo al menos la mitad de las aristas son buenas.*

Demostración. Primero, dirijamos cada una de las aristas hacia el extremo con mayor grado (o hacia el vértice con identificador mayor en caso de un empate). Primero mostraremos que para un vértice *malo* v (un vértice que no es bueno), el número de aristas salientes es al menos el doble de las aristas entrantes. Si éste no fuera el caso, entonces al menos un tercio de los vecinos de v (denotados por el conjunto S) tendría grado a lo más $gr(v)^+$. Así,

$$\sum_{w \in N_v} \frac{1}{2gr(w)^+} \geq \sum_{w \in S} \frac{1}{2gr(v)^+} \geq \frac{gr(v)^+}{3} \cdot \frac{1}{2gr(v)^+} = \frac{1}{6}, \tag{2.6}$$

lo cual contradice la hipótesis de que v es un vértice malo.

Por consiguiente, el número de aristas dirigidas hacia vértices malos es, a lo más, la mitad del número de aristas dirigidas que salen de vértices malos. Esto implica que al menos la mitad de las aristas son dirigidas hacia vértices buenos, lo cual las convierte en aristas buenas. □

Finalmente, estamos preparados para demostrar el siguiente resultado:

Teorema 2.8 ([40, Lema 3.3.7]). *El algoritmo 3 termina en tiempo $O(log(n))$ con probabilidad constante.*

Demostración. Por el lema 2.6, un vértice bueno (y por lo tanto una arista buena) será eliminado con probabilidad al menos constante. Debido al lema 2.7 sabemos que al menos la mitad de las aristas son buenas. Por consiguiente, el número esperado de aristas eliminado en cada fase es al menos una fracción constante de ellas. Así, podemos decir que existe una cota superior para la probabilidad de que alguna de las aristas no sea eliminada después de $O(log|E|)$ rondas. Puesto que $|E| \leq n^2$ y por lo tanto $log|E| \leq 2(log(n))$, se sigue que el algoritmo requiere $O(log|E|) = O(log(n))$ rondas. Finalmente, cada fase consiste de un número constante de rondas de comunicación y así podemos decir que con probabilidad alta el algoritmo 3 termina después es $O(log(n))$ rondas. □

Es importante mencionar que el algoritmo 3 trabaja bien incluso en gráficas generales. Además, se puede demostrar que el tiempo de ejecución del algoritmo 3 está

cerca del óptimo [25]. Es entonces natural preguntarse si existen algoritmos más rápidos para gráficas especiales como las de disco unitario. De hecho, el algoritmo que hemos presentado[10] ha sido por mucho tiempo el algoritmo de distribución más rápido que se conoce.

2.2. Algoritmo local para obtener una gráfica plana de una GDU

Como hemos mencionado anteriormente, el primer acercamiento al ruteo en redes *ad hoc* se hizo a través de algoritmos glotones. Estos algoritmos funcionan, de manera general, buscando el camino que ofrezca el beneficio más obvio e inmediato. Sin embargo, este tipo de algoritmos no garantizan que el mensaje se entregue al nodo destino.

En contraste con esto, la investigación giró en torno a la elaboración de algoritmos que sí garantizaran la entrega de los mensajes al trabajar en una GDU conexa y estática durante el tiempo que toma la entrega del mensaje. Además, estos algoritmos cumplen con la propiedad de que los vértices en las GDU no requieren cantidades grandes de memoria y lo único que deben de recordar siempre es su posición y la de sus vecinos.

Estos algoritmos funcionan encontrando una subgráfica plana y conexa de las GDU y aplican después, como subrutinas, los algoritmos de ruteo para gráficas planas.

2.2.1. Extraer una gráfica plana y conexa de una GDU

En esta sección presentamos un algoritmo distribuido y local para extraer una subgráfica conexa y plana de una GDU. Este algoritmo funciona calculando la intersección de una gráfica de disco unitario $H(V, E)$ con una gráfica plana y conexa que ya conocemos: la gráfica de Gabriel (GG).

Cuando no exista ambigüedad, usaremos sólo H en lugar de $H(V, E)$.

Lema 2.9 ([15, Lema 1])**.** *Sea H una gráfica de disco unitario conexa y $GG(H)$ la subgráfica de Gabriel de H, entonces $GG(H) \cap H$ es conexa.*

[10] Este algoritmo se debe a Michael Luby y aparece en su artículo *A Simple Parallel Algorithm For The Maximal Independent Set Problem* de 1986.

Demostración. Sea $AGPM(H)$ un árbol generador de peso mínimo de la gráfica completa que tiene como vértices a los vértices de H y cuyas aristas tienen como peso la distancia euclidiana entre sus vértices. Es bien sabido que $AGPM(H)$ es una subgráfica de $GG(H)$ y que por lo tanto $GG(H)$ es conexa [41]. Así, sólo necesitamos demostrar que $AGPM(H) \subseteq H$ si H es conexa.

En busca de una contradicción, supongamos que existe una arista (u, v) en $AGPM(H)$ cuya longitud es mayor a 1. Si eliminamos esta arista de $AGPM(H)$, entonces tenemos una gráfica con dos componentes conexas $C_u(H)$ y $C_v(H)$. Dado que H es conexa, entonces tiene una arista (w, x) de longitud no mayor a 1 tal que $w \in C_u(H)$ y $x \in C_v(H)$. Observemos que si reemplazamos a la arista (u, v) por la arista (w, x) en $AGPM(H)$ obtenemos una gráfica conexa en H con un peso menor que $AGPM(H)$, lo cual es una contradicción. En consecuencia, podemos afirmar que $AGPM(H) \subseteq H$ y así $GG(H) \cap H$ es conexa. □

Sea (u, v) una arista de H tal que $(u, v) \notin E_G$ (donde E_G es el conjunto de aristas de $GG(H)$). Entonces, por la definición de $GG(H)$ existe un punto w que está en el disco con diámetro (u, v). A este punto w le llamaremos *testigo* de que $(u, v) \notin E_G$. El siguiente lema muestra que cada arista que no pertenece a la subgráfica de Gabriel puede ser identificada y eliminada por sus extremos, usando sólo información local.

Lema 2.10 ([15, Lema 2])**.** *Sean u y v puntos de la gráfica H tal que $(u, v) \notin E_G$ y sea w un testigo de esto. Entonces $(u, w), (v, w) \in E$.*

Demostración. Sea m el punto medio de (u, v). Entonces $d(u, m) \leq \frac{1}{2}$, $d(v, m) \leq \frac{1}{2}$ y $d(w, m) \leq \frac{1}{2}$. De esta manera, por la desigualdad del triángulo tenemos que $d(u, w) \leq 1$, $d(v, w) \leq 1$ y así, (u, w) y (v, w) están en E. □

Así, al llegar a un vértice $v \in V$, se pueden eliminar las aristas incidentes en v que no estén en $E \cap E_G$ al descartar cualquier arista que no esté en $GG(N_v^+)$. Esto nos lleva al siguiente algoritmo, el cual es ejecutado por cada vértice $v \in V$.

Sea $circ(u, v)$ el disco con diámetro (u, v).

El lema 2.9 nos garantiza que si $GG(H)$ es plana [41], al aplicar el algoritmo a cada vértice de V, la gráfica resultante será conexa y plana. Podemos observar que

Algoritmo 4 Subgráfica plana y conexa usando la subgráfica de Gabriel ([15], Gabriel)

```
1: for u ∈ N_v do
2:     if circ(u, v) ∩ (N_v − {u, v}) ≠ 0 then
3:         eliminar a (u, v)
4:     end if
5: end for
```

cuando un vértice v ejecuta el algoritmo tarda a lo más $O(gr(v)^2)$ donde $gr(v)$ es el grado de v. Sin embargo, si usamos un algoritmo para encontrar el *diagrama de Voronoi* (y por lo tanto la triangulación de Delaunay) de un conjunto de n puntos de tiempo $O(n \cdot log(n))$ y si obtenemos a partir de dicha triangulación la gráfica de Gabriel (considerando las aristas de la triangulación intersecadas por aristas del digrama de Voronoi) en tiempo $O(n)$ [4], podemos obtener un algoritmo de tiempo $O(m \cdot log(m))$ donde $m = gr(v)$.

Dado lo anterior, podemos demostrar lo siguiente:

Teorema 2.11. *Sea H conexa y para cada vértice v consideremos su grado, digamos $m = gr(v)$ con $m \in \mathbb{N}$, entonces el algoritmo 4 calcula una subgráfica conexa y plana de H. El tiempo de complejidad de dicho algoritmo es $O(m \cdot log(m))$.*

Además de los resultados obtenidos probaremos que la subgráfica de Gabriel de H preserva los caminos más económicos en cuestión de energía entre cualesquiera dos vértices. Si además suponemos que estamos trabajando en el modelo $\Omega(1)$, podemos deducir que la distancia en $GG(H)$ entre cualesquiera dos vértices es igual (salvo factores constantes) a la distancia en H para toda métrica que se considere. Esto se demuestra en el siguiente lema.

Lema 2.12 ([20, Lema 5.5])**.** *En el modelo $\Omega(1)$, para cualquiera de las métricas posibles según la definición 1.2, el camino más corto en la subgráfica de Gabriel intersecada con la gráfica de disco unitario H es más largo sólo por un factor constante que el camino más corto en H para la métrica respectiva.*

Demostración. Primero demostraremos que, según el costo basado en la métrica de energía, al menos un camino óptimo en H es también un camino óptimo en $H \cap GG(H)$. Supongamos que $e = (u, v)$ es una arista de un camino óptimo p en H, según la métrica de energía. En busca de una contradicción, supongamos entonces que e no está contenida en la intersección $H \cap GG(H)$. Entonces existe un vértice

w en el disco con diámetro (u, v) o en su interior (ver figura 2.4). Observemos que las aristas $e' = (u, w)$ y $e'' = (v, w)$ son también aristas de H y como w está en el disco mencionado anteriormente, tenemos que $c_{dist}(e')^2 + c_{dist}(e'')^2 \leq c_{dist}(e)^2$. Si w está en el interior del disco con diámetro (u, v), la energía para el camino $p' : p - \{e\} \cup \{e', e''\}$ es más pequeña que la energía usada por p, y por lo tanto, p no es un camino óptimo para la métrica de energía, lo cual contradice la hipótesis. Si w está en el disco, p' es un camino óptimo y el resultado se cumple de igual manera. Así, tenemos que al menos un camino óptimo en H está contenido en $H \cap GG(H)$.

Supongamos ahora que $p^*_{H \cap GG(H)}$ es el camino óptimo para alguna función de costo $c(\cdot)$. Tenemos entonces que $c(p^*_{H \cap GG}) \leq c(p) \leq \alpha \cdot c_{energia(a)}(p)$ para alguna constante α. La última igualdad se debe al lema 1.8. Además, por el lema 1.9, tenemos que $c_{energia(a)}(p) \leq \beta \cdot c(p^*_H)$ para alguna constante β y p^*_H uno de los caminos más cortos con respecto a $c(\cdot)$ en H, que es lo que se quería demostrar. □

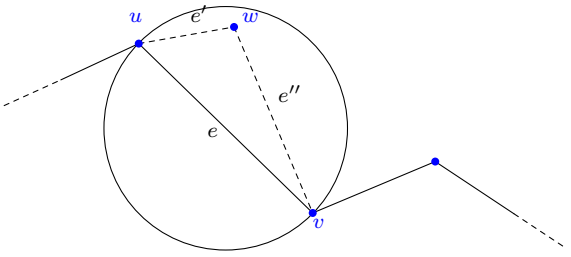


FIGURA 2.4: La arista e no está en la gráfica $H \cap GG(H)$.

2.3. Algoritmos de ruteo

Esta sección está basada en el capítulo 9 de [40].

Los algoritmos que veremos a continuación pueden ser definidos formalmente como sigue:

Definición 2.13 (Ruteo geográfico)**.** Sea $G(V, E)$ una gráfica geométrica. La tarea de un algoritmo de ruteo geográfico para una red *ad hoc* modelada por $G(V, E)$, es transmitir un mensaje desde una fuente $s \in V$ a un destino $t \in V$ a través del flujo de información hecho sobre las aristas de G, cumpliendo las siguientes condiciones:

- Todos los vértices $v \in V$ conocen sus posiciones geográficas así como las de sus vecinos en G.
- Al vértice s le es dada la posición del vértice t.
- La información que puede guardar un vértice v es de tamaño $O(log(N))$ donde N es el número (constante) de vértices que v recuerda haber visitado.
- Excepto por la información guardada antes mencionada, un vértice no puede tener más información sobre otros aspectos de la gráfica.

En la literatura podemos encontrar que el ruteo geográfico para redes *ad hoc* tiene otros nombres como algoritmos de ruteo de memoria $O(1)$ en [10, 13] y algoritmos locales de ruteo en [11].

Suponemos que el ruteo de información sucede de manera más rápida que el movimiento de los vértices en cuestión. Por lo tanto, los algoritmos de ruteo son diseñados para ser aplicados en vértices fijos respecto a su posición.

2.3.1. Ruteo glotón

El enfoque que es probablemente el más sencillo para el enrutamiento geográfico es el ruteo glotón: cada vértice envía el mensaje al *mejor* vecino según t, ver figura 2.5. Si por *mejor* entendemos *más cercano a* t. El ruteo glotón, tambien llamado *ruteo por brújula*[11] puede ser formulado de la siguiente manera:

Este procedimiento refleja claramente la simplicidad del enfoque. Sin embargo, podemos ver una desventaja como lo indica el paso 3 del algoritmo: es posible que

Algoritmo 5 Ruteo glotón (RG) ([11], *Compass Routing*)

1: Empezar en el vértice s.
2: Proceder al vecino más cercano al vértice t.
3: Repetir el paso 2 hasta llegar al vértice t o a un vértice v que sea mínimo local con respecto a la distancia de t.

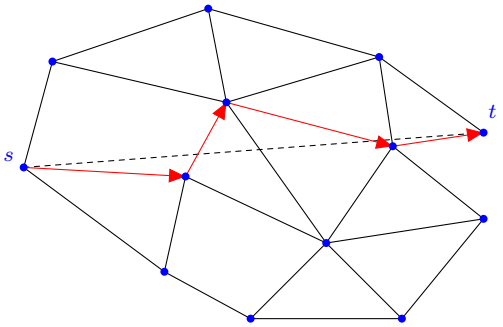


FIGURA 2.5: Ruteo glotón.

el mensaje nunca llegue al vértice t debido a un mínimo local, es decir, que se llegue a un vértice que no tenga ningún vecino que sea mejor que él, ver figura 2.6. Podríamos pensar que una manera de resolver esto es aplicando técnicas como el *backtracking*, sin embargo esta técnica no es una solución general, especialmente cuando tenemos que tomar en cuenta las restricciones de la definición 2.13. Por otro lado, interpretaciones alternativas de *mejor vecino* podrían ser consideradas, sin embargo, en la mayoría de los casos, no se logra enviar el mensaje al vértice t [11]. Informalmente podemos decir que lo anterior se debe al hecho de que el mensaje se mantiene relativamente cerca de la línea que conecta al vértice s con el vértice t.

Lo importante de este algoritmo es lo eficiente que resulta en el caso promedio y su buen comportamiento respecto a la escalabilidad:

Lema 2.14 ([29, Teorema 5.1]). *Si el algoritmo RG logra enviar el mensaje al vértice t, lo hace a lo más con una complejidad de $O(d^2)$, donde d denota la distancia euclidiana entre s y t.*

Demostración. Sea $p := u_1 = s, u_2, ..., u_k = t$ la sucesión de vértices que se visitan durante el ruteo glotón. Observemos que cualquier par de vértices u_i, u_j con índices impares i, j no son vecinos y que la distancia entre u_i y t decrece conforme i

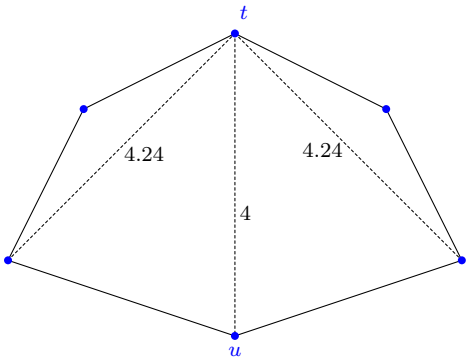


FIGURA 2.6: El vértice u es un mínimo local puesto que ninguno de sus vecinos está más cerca al destino t que él mismo.

decrece. Dado que el camino óptimo es de longitud d, todos lo vértices u_i están en una circunferencia de radio d con centro en t. Además, los vértices u_i y u_{i+k} con $k \geq 2$ no son vecinos, de lo contrario, habríamos elegido a u_{i+k} en lugar de u_{i+1} como sucesor de u_i en el camino p. Por lo tanto, los vértices $u_1, u_3, ..., u_{2\lceil \frac{m}{2} \rceil - 1}$ no son vecinos entre ellos. Según el *packing lemma*[11], la longitud del camino óptimo está acotada por $O(d^2)$. Así, tenemos que la complejidad de RG es $O(d^2)$. □

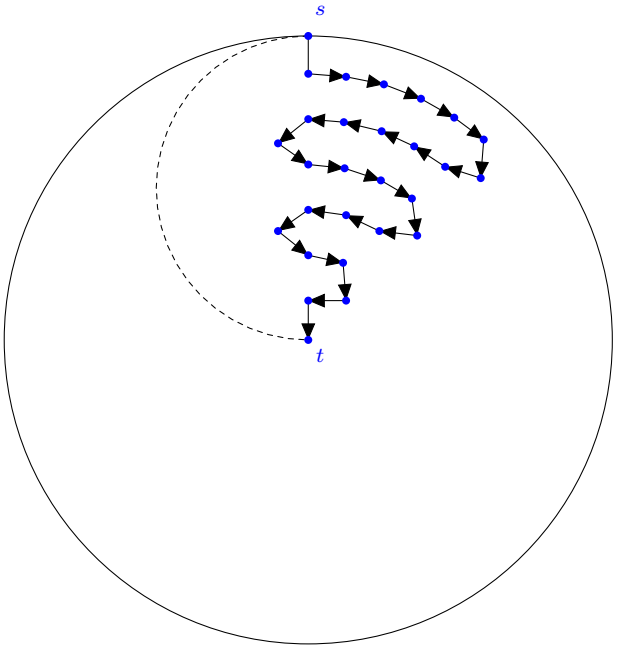


FIGURA 2.7: *Packing Lemma*

[11]Es decir, en una gráfica $G(V, E)$, si todos sus vértices están a distancia $1 - \epsilon$ con $\epsilon > 0$, entonces $|V| \leq r^2$ con r el radio de la circinferencia con menor área que contiene a G.

En la siguiente sección, el ruteo glotón se utiliza como subrutina de un algoritmo de ruteo de manera que, tanto en el caso promedio como en el peor de los casos, el rendimiento mejora.

2.3.2. Ruteo por caras

En la sección anterior, nos dimos cuenta de que el ruteo glotón no siempre logra enviar la información al destinatario. En esta sección introducimos un tipo de ruteo geográfico que, en contraste, siempre logra entregar el mensaje al destinatario si la red cumple con que, el nodo fuente y el nodo destino están conectados por un camino: ruteo basado en caras.

El primer algoritmo para ruteo geográfico que siempre entrega el mensaje al nodo destinatario se introdujo en [11] como *ruteo por caras RC*.

La técnica central de este algoritmo está basada en *caras*, las cuales son regiones contiguas separadas por las aristas de una gráfica plana. El algoritmo empieza en el vértice s y determina la primer cara, que es la cara incidente a s e intersecada por la línea $\overline{st}$ que une a s con el vértice t. Elige cualquiera de las dos aristas incidentes a s y empieza a recorrer la frontera de la cara guardando la información de las aristas de esa cara intersecadas por $\overline{st}$. Al terminar su recorrido por la cara actual, el algoritmo se dirije a la intersección más cercana a t y repite el mismo procedimiento. Si s y t están conectados, el ruteo por caras siempre encuentra un camino entre dichos vértices. Así, este algoritmo toma a lo más $O(n)$ pasos donde n el el número total de vértices en la gráfica.

2.3.3. Ruteo por caras adaptado

El algoritmo de *ruteo por caras adaptado RCAD*, a diferencia del algoritmo anterior, tomará en cuenta la distancia $|\overline{st}|$ entre los vértices s y t: limitará la complejidad del algoritmo con respecto a la longitud del camino más corto entre s y t [18].

En esta sección explicaremos esta nueva técnica como preámbulo al algoritmo principal. Observemos que el problema en el ruteo por caras es que necesita recorrer la frontera completa de cada cara y por lo tanto no podemos acotar la complejidad

de este algoritmo por el costo de algún camino óptimo entre s y t. Sin embargo, podemos hacer lo siguiente: suponiendo que conocemos la longitud de un camino óptimo entre s y t, la búsqueda se restringe a un área específica, en particular a una elipse cuyo tamaño se elige de manera que contenga un camino óptimo entre s y t. Si durante el recorrido que hace el algoritmo a través de la frontera de una cara interseca a la elipse, entonces da vuelta atrás y continúa su recorrido en sentido contrario hasta que vuelva a intersecar con la elipse, lo cual completa la búsqueda en la cara actual. A este algoritmo se le conoce como *ruteo por caras acotado* RCA. De manera breve: como RCA no recorre una arista más de un número constante de veces y como la elipse acotada (recordemos que estamos en el modelo $\Omega(1)$) no contiene más de $O(|\overline{st}|^2)$ aristas, la complejidad del RCA es $O(c(p^*)^2)$, donde p^* es un camino óptimo de s a t y $c(\cdot)$ puede estar basada en las métricas de enlace, euclidiana o de energía.

Así, utilizando el algoritmo anterior podemos plantear el *ruteo por caras adaptado*: RCA inicia con una elipse de tamaño estimado según un camino óptimo entre s y t. Si RCA falla en llegar a t, entonces regresa a s y reincia el proceso con una elipse de tamaño doble. Si s y t están conectados, entonces RCAD llegará a t eventualmente. Esta iteración está asintóticamente dominada por la complejidad de la cantidad de pasos usados para el proceso de la última elipse, cuya área es, a lo más, proporcional al cuadrado del costo de un camino óptimo. Como consecuencia, también la complejidad del RCAD estará acotada por $O(c(p^*)^2)$.

En la sección 2.3.4 mostraremos que dentro del modelo $\Omega(1)$ no existe un algoritmo local para ruteo geográfico que tenga un mejor desempeño: RCAD es asintóticamente óptimo.

2.3.4. Otro Ruteo Por Caras Adaptado

Como vimos en la sección 2.3.1, el ruteo glotón tiene buen desempeño cuando logra alcanzar el nodo destino. Una manera natural de sacar ventaja de este algoritmo es combinarlo con el ruteo por caras. Como primer intento, el ruteo glotón y el RCAD pueden ser combinados de manera sencilla: empezamos implementando el ruteo glotón tal que si llegamos a un mínimo local, entonces pasamos al RCAD para solucionar este problema. Ha sido demostrado en [22] que esta combinación deja de ser asintóticamente óptima. Sin embargo, una variante del RCAD llamada *Otro ruteo por caras adaptado ORCAD* fue diseñada de manera que su combinación

con el ruteo glotón finalmente logra un algoritmo que es asintóticamente óptimo en el caso promedio.

Similarmente a la descripción que hicimos en la sección anterior con el RCAD, explicaremos el algoritmo otro ruteo por caras adaptado en tres pasos: ORC, ORCA Y ORCAD.

Otro ruteo por caras ORC, difiere del ruteo por caras de la siguiente manera: en lugar de cambiar a la siguiente cara según la *mejor* intersección de la frontera de la cara y $\overline{st}$, ORC regresa (después de explorar por completo la frontera de la cara actual) al punto en la frontera (o uno de los puntos) más cercano al vértice t (ver figura 2.8).

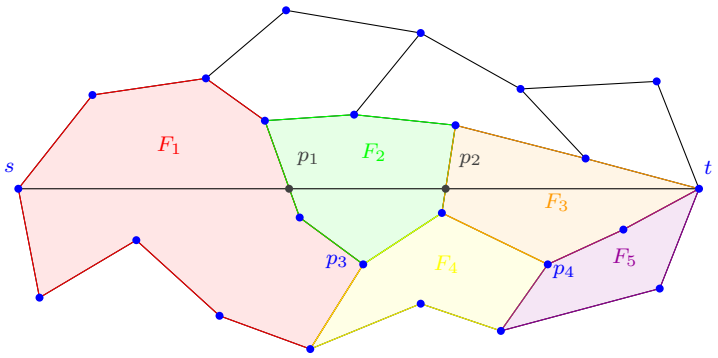


FIGURA 2.8: El ruteo por caras RC empieza en el vértice s, explora la frontera de F_1 y encuentra a p_1 en $\overline{st}$, enseguida explora la frontera F_2 y encuentra a p_2, por último recorre la frontera F_3 y encuentra a t que es el vértice destino. En contraste, el otro ruteo por caras ORC encuentra a p_3 que es el punto más cercano a t en la frontera de de F_1, continúa explorando la frontera de F_4 y encuentra p_4, para finalmente encontrar t recorriendo la frontera de F_5.

Algoritmo 6 Otro ruteo por caras ([22], *Other Face Routing OFR*)

1: Empieza en s y recorre la frontera de la cara inmediata F que contenga un fragmento del segmento $\overline{st}$.
2: Una vez terminada la exploración de la frontera de F, regresar al punto p más cercano a t.
3: Cambiar a la siguiente cara contigua que contenga a $\overline{pt}$ y repetir el paso 1.
4: Repetir hasta llegar al vértice t.

En el siguiente lema demostramos que el número de pasos en ORC está acotado:

Lema 2.15 ([22, Lema 4.1])**.** *ORC siempre termina en $O(n)$ pasos, donde n es el número de vértices. Si s y t están conectados, ORC siempre llega al vértice t; de otra manera, detectará desconexión.*

Demostración. Sea $F_1, F_2, ..., F_k$ una sucesión de caras que son visitadas durante le ejecución de ORC. Supongamos que s y t están conectados. Como el cambio entre caras siempre sucede en algún punto de la frontera más cercano a t y como la siguiente cara es elegida de manera que los puntos de su frontera están más cercanos a t que los de la cara anterior, ninguna cara puede ser visitada dos veces. Además, sean $p_1, p_2, ..., p_t$ el rastro de la ejecución de ORC, donde p_i, $i \geq 1$ es el punto cuya distancia con t desde la frontera de la cara F_i es la mínima. Como ninguna cara es recorrida dos veces, tenemos que $\forall i > j : |\overline{p_i t}| < |\overline{p_j t}|$. Por lo tanto, si s y t están conectados, eventualmente llegaremos a la cara que tenga a t en su frontera. (De otra manera, existiría una i para la cual $p_i = p_{i+1}$, lo cual implicaría que la gráfica es disconexa.)

Como cada una de las caras es recorrida una sola vez, cada arista es recorrida a lo más 4 veces. Como toda gráfica plana corresponde a la proyección de un poliedro en el plano, podemos usar la fórmula de Euler para poliedros: $n - m + f = 2$, donde n, m y f corresponden al número de vértices, aristas y caras, respectivamente. Además, observemos que para $n > 3$ cada cara es delimitada por al menos tres aristas y cada arista es adyacente a lo más a dos caras; esto implica que $3f \leq 2m$. Usando la fórmula de Euler, tenemos que $3m - 3n + 6 = 3f \leq 2m$ y por lo tanto $m \leq 3n - 6$. Así, ORC termina después de $O(n)$ pasos. □

Aclaremos que el algoritmo puede darse cuenta de que s y t no están conectados: si para algún $i \geq 1$ se tiene que $p_i = p_{i+1}$ entonces no existe camino entre dichos vértices. Esto es informado al vértice s usando el mismo algoritmo ORC en dirección contraria.

Si aplicamos el ORC a una subgráfica de Gabriel, el algoritmo puede ser simplificado de la siguiente manera: en lugar de cambiar de caras en el *punto* de la frontera más cercano a t, podemos tomar el *vértice* más cercano a t. Esta modificación no afecta al lema 2.15 puesto que en el paso 2 siempre se cambia de una cara a otra. Conforme las definiciones y explicaciones se vuelvan más claras, usaremos esta modificación del algoritmo en la descripción de los siguientes algoritmos.

La principal desventaja de ORC es que podría tener que recorrer una cara demasiado grande cuya exploración sea demasiado costosa respecto a un camino óptimo entre s y t. Con el fin de evitar esta desventaja, se usará la técnica implementada en el algoritmo RCA que acota el área de búsqueda por una elipse. A este algoritmo lo llamamos *Otro ruteo por caras acotado ORCA*.

Por simplicidad supondremos que s y t están conectados y que l es una estimación de la longitud Euclidiana de un camino de longitud mínima ente s y t. Además, sea ε la elipse con focos s y t y con eje mayor de longitud l (en otras, palabras, ε contiene todos los caminos de s a t de longitud a lo más l).

Algoritmo 7 Otro ruteo por caras acotado ORCA ([22], *Other Bounded Face Routing OBFR*)

1: Paso 1 del algoritmo 6.
2: Paso 2 del algoritmo 6, pero en lugar de recorrer por completo a F, cuando interseque por primera vez con ε dará vuela atrás y recorrerá la cara en dirección contraria hasta que interseque a la elipse por segunda vez.
3: Paso 3 del algoritmo 5 con una pequeña modificacion: si el vértice más cercano a t en la frontera de la cara F es el mismo que en la iteración anterior, esto es, no hubo avance en el paso 2, el algoritmo comunicará a s una falla a través del mismo ORCA.

Las figuras 2.9 y 2.12 muestran la ejecución de ORCA, si la elipse es elegida muy pequeña y si la elipse contiene un camino de s a t, respectivamente. La complejidad de ORCA puede ser acotada como se muestra a continuación:

Teorema 2.16 ([22, Lema 4.2])**.** *Si la longitud l del eje mayor de ε es al menos la longitud de un camino mínimo de s a t (respecto a la métrica euclidiana), ORCA logra enviar la información al vértice t. De otra manera, ORCA reporta al vértice s que hubo una falla. En cualquiera de los casos, la complejidad de ORCA no es mayor que $O(l^2)$.*

Demostración. Supongamos que l es al menos la longitud de un camino mínimo p^*, esto es, p^* está completamente contenido en ε. Como el recorrido de ORCA se mantiene en el interior de ε, sólo nos interesa analizar la parte de la gráfica que está dentro de ε. Observemos que si una cara es intersecada por ε, las regiones resultantes serán consideradas como caras también.

En búsqueda de una contradicción, supongamos que ORCA reporta un fallo al vértice s, esto es, el algoritno no tuvo progreso en el paso 3. Esto sólo es posible si

la frontera de la actual cara que está siendo recorrida corta el interior de la elipse en dos regiones ajenas, una que contiene a s y otra a t, ver figura 2.9. En este caso, ε no contendría ningún camino entre s y t, lo cual contradice la hipótesis y por lo tanto la primer parte del lema es verdadera.

Finalmente, resta probar que la complejidad del algoritmo es $O(l^2)$. Si ε contiene un camino entre s y t, cada cara es visitada a lo más una vez (por la misma razón que pasa esto en ORC). De otra manera, cada cara es visitada a lo más dos veces (la cara donde el algoritmo encuentre una falla puede ser recorrida dos veces). Además, durante el recorrido sobre la frontera de una cara, una arista puede ser visitada a lo más 4 veces. Como consecuencia de esto, toda arista es recorrida a lo más un número constante de veces durante la ejecución completa de ORCA.

Debido a la planaridad de la gráfica considerada, el número de aristas es lineal respecto al número de vértices. Además, según el modelo $\Omega(1)$, los círculos con radio $\frac{d_0}{2}$ alrededor de cada vértice no se intersecan entre ellos. Como la longitud del semieje mayor de la elipse ε es $\frac{l}{2}$, y como el área de ε es más pequeña que $\pi \cdot \frac{l}{2}^2$, el número de vértices n' en el interior de ε está acotado por:

$$n' \leq \frac{\pi \cdot \frac{l}{2}^2}{\pi \cdot (\frac{d_0}{2})^2} = \frac{l^2}{d_0^2} \in O(l^2).$$

Teniendo esta cota superior para el número de mensajes enviados, el último enunciado del lema se sigue del lema 2.12. □

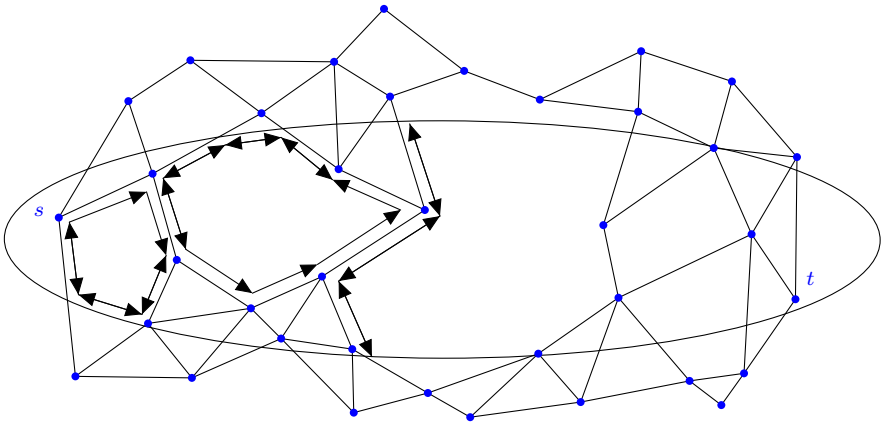


FIGURA 2.9: Ejecución del algoritmo otro ruteo por caras acotado si la elipse elegida no es lo suficientemente grande.

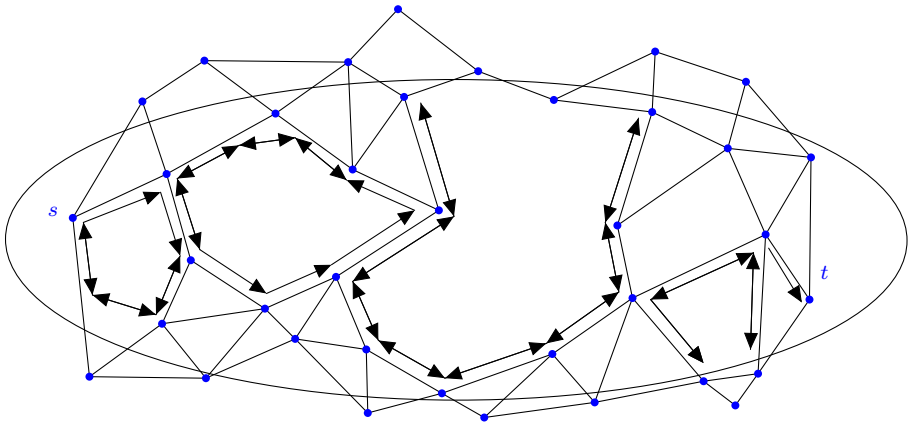


FIGURA 2.10: Ejecución del algoritmo otro ruteo por caras acotado si la elipse elegida es lo suficientemente grande para contener un camino de s a t.

Como usualmente no tenemos información *a priori* sobre la longitud del camino óptimo entre s y t, iniciaremos (como en RCAD) con una pequeña estimación sobre el tamaño de la elipse e iterativamente aumentaremos su tamaño si es necesario, hasta alcanzar al vértice t.

Algoritmo 8 Otro ruteo por caras adaptado ORCAD ([22], *Other Adaptive Face Routing OAFR*)

1: Iniciamos con una elipse ε con focos en s y t y cuyo eje mayor tiene longitud $2 \cdot |\overline{st}|$.
2: Ejecutamos ORCA con la elipse ε.
3: Si el vértice t no ha sido alcanzado, duplicamos la longitud del eje mayor de la elipse ε y repetimos el paso 2.

Como ORCA es capaz de distinguir la diferencia entre los casos donde la elipse no es suficientemente grande y la desconexión entre s y t, también lo será ORCAD. Además, la complejidad de ORCAD es acotada como se demuestra a continuación:

Teorema 2.17 ([22, Lema 4.3])**.** *Si s y t están conectados, ORCAD llega a t con una complejidad de $O(c(p^*)^2)$, donde p^* es un camino óptimo entre s y t. Si entre dichos vértices no existe ningún camino, ORCAD lo detecta y envia mensaje al vértice s.*

Demostración. Denotemos al primer estimado de la longitud del camino óptimo como l_0 y a los siguientes como $l_i := 2^i l_0$. Además, definimos al número $k > 0$ tal

que $l_{k-1} < c_{dist}(p^*) \leq l_k$. Para $c(\text{ORCA}[l])$, la complejidad de ORCA acotada por una elipse ε con eje mayor de longitud l, tenemos que $c(\text{ORCA}[l]) \in O(l^2)$ y por lo tanto $c_{enlace}(\text{ORCA}[l]) \leq \lambda \cdot l^2$ para una constante λ. La complejidad de ORCAD está acotada finalmente por:

$$\begin{aligned} c_{enlace}(\text{ORCAD}) \leq \sum_{i=0}^{k} c_{enlace}(\text{ORCA}[l]) &\leq \sum_{i=0}^{k} \lambda(2^i l_0)^2 \\ &= (\lambda l_0)^2 \frac{4^{k+1}-1}{3} < \frac{16}{3}\lambda(2^{k-1}l_0)^2 \\ &< \frac{16}{3}\lambda \cdot c_{dist}(p^*)^2 \in \text{O}(c_{dist}(p^*)^2). \end{aligned} \tag{2.7}$$

□

Observemos que la complejidad para ORCAD (y por lo tanto también para RCAD) de detectar que s y t no están conectados está acotada por $O(n \cdot log(n))$, donde n es el número de vértices en la componente conexa que contiene a s: El número de veces que se ejecuta ORCA es a lo más $O(log(n))$, mientras que la complejidad de cada una de estas ejecuciones es, a lo más, lineal en n. Como se ilustra en la figura 2.11, existe una gráfica para la cual ORCAD tiene una complejidad de $\Theta(n \cdot log(n))$.

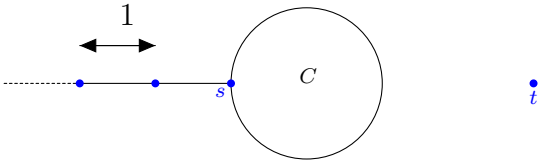


FIGURA 2.11: Si $\frac{n}{2}$ vértices están en el cúmulo C (representado por el disco gris) y $\frac{n}{2}$ vértices forman el segmento de recta de la izquierda, antes de que ORCAD detecte que s y t no están conectados, ejecuta la subrutina ORCA $\Theta(log(n))$ veces, donde cada ejecución tiene una complejidad de $\Theta(n)$ si los vértices en C forman un laberinto.[12]

[12]Podemos ver un laberinto como una gráfica si consideramos cada coyuntura (intersección) del laberinto como un vértice, y añadimos aristas a la gráfica entre coyunturas adyacentes que no estén bloqueadas por una pared.

Como hemos visto, el algoritmo ORCAD llega al nodo destino con un costo de $O(c^2)$ donde c es el camino más corto entre el vértice que representa al nodo fuente y el vértices que represental al nodo destino. Una pregunta que surge de manera intuitiva es si esta es una buena aproximación o si existen algoritmos que tengan un mejor desempeño. El siguiente teorema muestra que no existe un algorimo de ruteo geográfico según la definición 2.13 que tenga mejor desempeño que ORCAD. Esto se demuestra construyendo una cota inferior como en el siguiente teorema:

Teorema 2.18. *Sea c el costo del mejor camino entre dos vértices s y t. Entonces, existen gráficas donde cualquier algoritmo de ruteo geográfico ad hoc determinista (aleatorio) tiene una cota esperada de $\Omega(c^2)$ para cualquier métrica según la definición 1.2.*

Demostración. Construyamos la gráfica que prueba el teorema como sigue. Dado un entero positivo k, definamos la gráfica con métrica euclidiana de la siguiente manera (VER FIGURA): en un disco, distribuimos $2k$ vértices tal que la distancia entre cualesquiera dos es exactamente 1. Por lo tanto, el disco tiene radio $r \approx \frac{k}{\pi}$. Ahora, consideremos sólo la mitad de estos vértices de manera que entre dos seleccionados haya uno que no lo está. Para cada uno de estos vértices construyamos una cadena de $\lceil \frac{r}{2} \rceil - 1$ vértices en línea recta con dirección hacia el centro del disco. La distancia entre cuales quiera de estos vértices en las cadenas es exactamente 1. Sea w un vértices arbitario en el disco, sólo para este vértice vamos a contruir una cadena que consista de $\lceil r \rceil$ vértices con distancia 1 entre ellos. Observemos que el último vértice de la cadena de w es el centro del disco y que la arista entre el centro y el penúltimo nodo podría tener longitud distinta de 1.

La gráfica de disco unitario consistirá de todas las aristas en el disco que unan a los vértices con sus vértices vecinos (izquierda y derecha) y las aristas que unen a los vértices en cada cadena. En particular, no habrá aristas entre dos cadenas puesto que todas las cadenas (excepto la que empieza en w) terminan estrictamente afuera del disco de radio $\frac{r}{2}$. Así, la gráfica tiene k cadenas con $\Theta(k)$ vértices cada una. Supongamos que queremos rutear desde un vértice s en el disco hacia el vértice que es el centro del disco t. Un camino óptimo entre s y t seguiría el camino más corto hacia w y después seguiría directamente por la cadena de w hasta encontrar el centro del disco t. Esto tendría un costo c tal que $c \leq k + r + 1 \in O(k)$. Un algoritmo de ruteo que use tablas de ruteo, podría fácilmente encontrar este camino.

En contraste, un algoritmo de ruteo geográfico, necesita buscar el vértice w y para esto necesita explorar, dado cierto orden, la cadena de cada vértice que va encontrando en el disco. Por lo tanto, si suponemos que el vértice s se elige de tal manera que el orden del recorrido nos haga revisar la cadena de w hasta el final, el algoritmo llegaría al destino después de visitar $\Theta(k^2)$ vértices.

De la misma manera, en algorimos aleatorios, podemos forzar a que en el peor caso se visiten, con probabilidad constante, $\Omega(k)$ cadenas antes de encontrar al vértice w. Por lo tanto, el costo esperado según la métrica de enlace es $\Theta(k^2)$.

Como todas las aristas de nuestra gráfica (excepto una) tienen longitud 1, los costos según las métricas euclidiana, de enlace y de energía, son iguales. Así, según la definición 1.2, para cualquier costo $c'(\cdot)$ se tiene que $c'(1)$ es también una constante, entonces la cota inferior $\Omega(c^2)$ se mantiene para todas las métricas. □

Dada esta cota inferior, podemos afirmar que ORCAD es asintóticamente óptimo para gráficas de disco unitario en el model $\Omega(1)$.

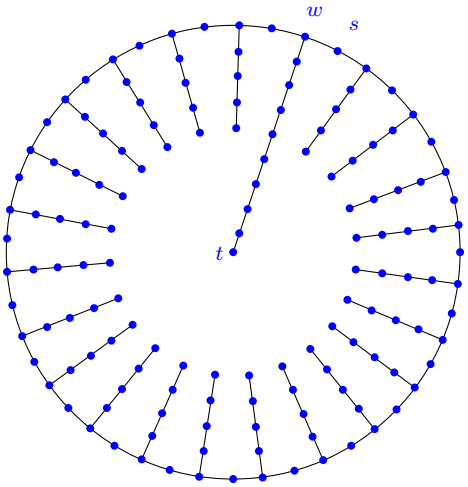


FIGURA 2.12: Gráfica en el peor de las casos para los vértices s y t.

Teorema 2.19. *Sea c el costo de una camino óptimo entre dos vértices en una gráfica de disco unitario en el modelo $\Omega(1)$. En el peor caso, el costo de aplicar ORCAD a dicha gráfica para encontrar una aproximación al camino óptimo es $\Theta(c^2)$. Esta cota es aintóticament óptima.*

Demostración. Este teorema es consecuencia inmediata de los teoremas 2.17 y 2.18. □

2.3.5. Combinación entre ruteo glotón y ruteo por caras

El algoritmo para el ruteo glotón, presentado en la sección 2.3.1, es considerado por su simplicidad conceptual y de implementación. La sencillez de este algoritmo contrasta con lo inflexible de la búsqueda sobre la frontera de las caras en el ruteo por caras. Con fines prácticos, se intentó mejorar el desempeño del ruteo por caras aprovechando las ventajas del ruteo glotón.

En esta sección, resumiremos el algoritmo diseñado combinando el ruteo glotón y el ruteo por caras, llamado ORCADG+[13]. Este algoritmo es una combinación en el siguiente sentido: mientras sea posible, el algoritmo intentará rutear de manera glotona hasta que se encuentre con un mínimo local respecto a la distancia con el vértice t, en cuyo caso, implementará el ruteo por caras. Cuando ORCADG+ esté en modo ruteo por caras, el algoritmo sólo buscará en un área particular como lo hace ORCAD.

El comportamiento antes descrito está fundamentado en el hecho de que el ruteo glotón es más eficiente que el ruteo geográfico en el caso promedio. Debido a esto es que ORCADG+ intenta matenerse en modo ruteo glotón mientras le sea posible. Sin embargo, se ha demostrado que esto no puede suceder de forma simplista [20, 21]: por ejemplo, en el caso en el que el algoritmo en modo ruteo por caras esté más cerca del vértice t, conviene más avanzar hacia él que salir del mínimo local. En caso contrario, el algoritmo dejaría de ser asintóticamente óptimo.

Con el fin de preservar dicha propiedad, el algoritmo ORCADG+ utiliza dos contadores p y q. El contador p lleva la cuenta de los vértices visitados durante la fase actual del ruteo por caras que se encuentran más cerca del destino. El contador q lleva la cuenta de cuántos vértices están al menos tan lejos del destino como el punto de partida de la fase actual del ruteo por caras. En cuanto se cumpla una determinada condición, ORCADG+ continúa en modo glotón.

[13] Expresando la combinación del algoritmo glotón y el ruteo por caras, el acrónimo ORCADG significa *otro ruteo por caras adaptado glotón*. El signo "+" indica que ORCADG+ es una mejora de un algoritmo similar definido anteriormente con el nombre de ORCADG [22].

El algoritmo ORCADG+ usará los paramétros ρ_0, ρ y σ definidos antes de iniciar, cuyos valores pemanecen constantes durante la ejecución. Para que el algoritmo trabaje adecuadamente, los parámetros deben de cumplir las siguientes condiciones:

$$1 \leq \rho_0 < \rho \text{ y } 0 < \sigma.$$

Algoritmo 9 Otro ruteo por caras adaptado glotón ORCADG+ ([20], ORCADG+)

1: Empieza en s. Iniciamos con una elipse ϵ con focos en s y t y con eje mayor de longitud $l := \rho_0|\overline{st}|$.
2: **(Modo ruteo glotón)** Ejecutar el ruteo glotón hasta llegar a t o a un mínimo local. En el primer caso el algoritmo termina, en el segundo caso avanza al siguiente paso. Siempre que sea posible, se reduce el radio ($l := \frac{l}{\rho}$) siempre y cuando el vértice visitado permanezca en el interior de C.
3: **(Modo ruteo por caras)** Sea u_i el mínimo local visitado actualmente. Empezar a explorar el contorno de F_i, la cara que tiene a la línea $\overline{u_i t}$ en el entorno inmediato de u_i. Al completar la exploración de la cara F_i, regresa a u_i y avanza a uno de los vértices ya visitados tal que hasta el momento sea el más cercano a t y continúa al paso 1. Si ninguno de los vértices visitados está más cerca de t que u_i, entonces el algoritmo reporta a la fuente que s y t no están conectados (usando ORCADG+). Durante la exploración de la frontera F_i se usan dos contadores p y q para llevar la cuenta del número de vértices visitados en la frontera de F_i: p cuenta los vértices más cercanos a t que u_i y q los vértices que no se encuentran más cerca de t que u_i. Se realiza una acción especial si se cumple una de las siguientes condiciones:
4: Al llegar a ϵ por primera vez, vuelve atrás y continúa explorando el contorno de F_i en la dirección opuesta.
5: ϵ es alcanzado por segunda vez: Si ninguno de los vértices visitados está más cerca de t que u_i, se amplía ϵ ($l := \rho l$) y se continúa al paso 2 como si se partiera de u_i. En caso contrario, se avanza al vértice visitado hasta el momento más cercano a t y continuar con el paso 1.
6: Si $p > \sigma q$, es decir, hemos visitado (hasta un factor constante σ) más vértices en la frontera de F_i que están más cercanos a t que vértices que no lo están, avanzamos al vértice visto hasta ahora más cercano a t (si no es el vértice actualmente visitado) y se continúa con el paso 1.

Se puede demostrar que el algoritmo ORCADG+ conserva la propiedad de ORCAD de ser asintóticamente óptimo, en particular, se garantiza alcanzar el destino con una complejidad de a lo más $O(c(p^*)^2)$, donde p^* es un camino óptimo entre el origen y el destino [20].

Por otra parte, se demostró, mediante una simulación en redes generadas por nodos acomodados aleatoriamente en un campo determinado, que la combinación de ORCADG+ entre ruteo glotón y ruteo geográfico, es también benéfica para el ruteo en *gráficas de caso promedio* [20, 21, 35].

En esta sección se ha demostrado que utilizando el ruteo por caras mejorado por una limitación a un área de búsqueda y una técnica de contador, el algoritmo ORCADG+ requiere como máximo $O(c^2)$ pasos, donde c es el costo del camino más corto que conecta al origen con el destino. Junto con la correspondiente gráfica acotada inferiormente, se demuestra que este algoritmo es asintóticamente óptimo. Utilizando el ruteo glotón, el algoritmo también se vuelve eficiente en gráficas de caso promedio, como se demuestra por simulación y comparación con algoritmos similares. En este sentido, el algoritmo OCARDG+ puede considerarse una síntesis en simplicidad y eficiencia en casos promedios por un lado, y por el otro en cuanto a la corrección y la optimización asintótica en el peor de los casos.

2.3.6. Conclusión

Las primeras propuestas de ruteo geográfico -sugeridas hace más de dos décadas- eran de naturaleza puramente glotona [5, 7, 27]. Todas estas propuestas (incluyendo una contribución realizada posteriormente en [11]) tienen en común que pueden no llegar al nodo destino. El primer algoritmo de ruteo geográfico que sí garantiza la entrega es el ruteo por caras basado en la planarización de la gráfica e introducido en [11]. Han existido sugerencias posteriores de algoritmos con garantía de entrega de mensajes [10, 13, 15, 17, 16]; sin embargo, cada uno de ellos tiene sus problemas o, al menos en el mejor de los casos, no supera al ruteo por caras original. En [19] podemos encontrar un enfoque específico en los algoritmos relacionados al ruteo geográfico. En [18] se propuso el ruteo por caras adaptado RCAD, cuya complejidad de tiempo está acotada con respecto al costo de un camino óptimo. Han habido propuestas con fines prácticos para combinar el ruteo glotón con el ruteo por caras [14, 15, 17], aunque sin garantías competitivas en el peor de los casos. En [22] se introdujo el algoritmo ORCADG; hasta donde sabemos, este fue el primer algoritmo que combinó el ruteo glotón y el ruteo por caras de una manera óptima en el peor de los casos; la variante de dicho algoritmo, ORCADG+ en [20] sigue siendo asintóticamente óptima en el peor de los casos, mientras que mejora la eficiencia del ORCAD en el caso promedio. En los últimos

años se han realizado los primeros experimentos de ruteo geográfico. En particular, de ruteo por caras en redes prácticas [30, 31]. Otros enfoques han estudiado el ruteo geográfico aligerando las premisas [32, 37] o reforzándolas [24, 26, 34].

Capítulo 3

Conclusiones

En esta tesis, discutimos la utilidad de los conjuntos dominantes e independientes en las gráficas que modelan a las redes de comunicación inalámbricas. La discusión se hizo desde una perspectiva local, tanto teórica como práctica de las gráficas. La investigación fue motivada por los escenarios realistas que pueden representar las redes inalámbricas *ad hoc*. En primer lugar, hemos discutido las posibilidades de modelar las redes de comunicación inalámbricas mediante gráficas geométricas y hemos considerado la propiedad de crecimiento (polinómicamente) acotado. Disponer de una representación geométrica difiere en gran medida del caso en el que sólo está presente la información de adyacencia de una gráfica. Presentamos un esquema de aproximación en tiempo polinomial para los problemas de encontrar un conjunto independiente maximal y un conjunto mínimo dominante, basado en vecindades locales de los vértices de gráficas geométricas.

Las redes inalámbricas *ad hoc* con pocos recursos son intrínsecamente locales. Los nodos suelen carecer de suficiente memoria y potencia de cálculo para obtener y mantener una visión global de la topología de la red. En este ámbito, los conjuntos independientes maximales desempeñan un papel importante, tanto en la teoría como en la práctica. Este enfoque crea implícitamente una red basada en un conjunto independiente maximal. Este tipo de organizacion y agrupación permiten que los nodos tengan periodos de reposo relativamente largos, manteniendo al mismo tiempo una estructura conexa y dominante que mantiene la red global operativa y eficiente.

Aunque esta tesis presenta las respuestas positivas diseñadas para solucionar los problemas de encontrar un conjunto independiente maximal y un conjunto dominante mínimo en gráficas de disco unitario, todavía hay interesantes cuestiones abiertas en esta área.

La parte fundamental de este trabajo consiste en presentar los modelos y algoritmos diseñados mediante los cuales podamos comprender mejor las dificultades de los problemas de ruteo en gráficas geométricas, en particular, en gráficas de disco unitario. Lás técnicas presentadas tienen por objetivo mejorar el ruteo en gráficas que modelan redes con enlaces inestables que pueden cambiar durante dicho ruteo.

En particular, se analizó el ruteo por caras con el objetivo de identificar las condiciones en las que puede garantizar la entrega de los mensajes. Por lo tanto, se presentan algoritmos deterministas y se proporcionan pruebas teóricas para garantizar la entrega de mensajes en los modelos que nos interesan.

3.1. Trabajo a futuro

Durante el estudio de los algoritmos de ruteo en gráficas geométricas hemos observado que deben cumplirse ciertas restricciones, como el que la gráfica sea plana y conexa. Al respecto, se puede investigar bajo cuáles restricciones adicionales se puede garantizar la entrega de los mensajes en una red de comunicación.

El trabajo sobre dicho ruteo ha supuesto que el nodo destino es estacionario. Investigar sobre el problema de que el nodo destino sea móvil es de interés para trabajar a futuro: es posible que el ruteo por caras pueda combinarse con un esquema eficiente de actualización de la ubicación para garantizar la entrega.

En el caso de los algoritmos para calcular soluciones aproximadas, el siguiente paso sería estudiar las distintas variantes de conjuntos independientes y dominantes. Por ejemplo, considerar los conjuntos dominantes conexos y los conjuntos independientes maximales que no dependan de información geométrica y tratar de dar algoritmos de aproximación.

En general, si bien es importante estudiar los problemas de la teoría de gráficas en gráficas generales, también es fundamental estudiar estos problemas en clases de gráficas que modelan topologías que se dan en problemas prácticos.

Bibliografía

[1] K Ruben Gabriel y Robert R Sokal. «A new statistical approach to geographic variation analysis». *Systematic zoology* 18.3 (1969), págs. 259-278.

[2] David S Johnson. «Approximation algorithms for combinatorial problems». *Journal of computer and system sciences* 9.3 (1974), págs. 256-278.

[3] László Lovász. «On the ratio of optimal integral and fractional covers». *Discrete mathematics* 13.4 (1975), págs. 383-390.

[4] Michael Ian Shamos y Dan Hoey. «Closest-point problems». *16th Annual Symposium on Foundations of Computer Science (1975)*. IEEE. 1975, págs. 151-162.

[5] Ting-Chao Hou y Victor Li. «Transmission range control in multihop packet radio networks». *IEEE Transactions on Communications* 34.1 (1986), págs. 38-44.

[6] Michael Luby. «A simple parallel algorithm for the maximal independent set problem». *SIAM journal on computing* 15.4 (1986), págs. 1036-1053.

[7] Gregory G Finn. *Routing and addressing problems in large metropolitan-scale internetworks*. Inf. téc. University of Southern California Marina del Rey Information Sciences Inst., 1987.

[8] Madhav V Marathe, Heinz Breu, Harry B Hunt III, Shankar S Ravi y Daniel J Rosenkrantz. «Simple heuristics for unit disk graphs». *Networks* 25.2 (1995), págs. 59-68.

[9] Uriel Feige. «A threshold of ln n for approximating set cover». *Journal of the ACM (JACM)* 45.4 (1998), págs. 634-652.

[10] Prosenjit Bose y Pat Morin. «Online routing in triangulations». *International symposium on algorithms and computation*. Springer. 1999, págs. 113-122.

[11] Evangelos Kranakis, Harvinder Singh y Jorge Urrutia. «Compass Routing on Geometric Networks». *Proceedings 11 Th Canadian Conference on Computational Geometry*. 1999, págs. 51-54.

[12] Franz Aurenhammer y Rolf Klein. «Voronoi Diagrams.» *Handbook of computational geometry* 5.10 (2000), págs. 201-290.

[13] Prosenjit Bose, Andrej Brodnik, Svante Carlsson, Erik D Demaine, Rudolf Fleischer, Alejandro López-Ortiz, Pat Morin y J Ian Munro. «Online routing in convex subdivisions». *International Symposium on Algorithms and Computation*. Springer. 2000, págs. 47-59.

[14] Brad Karp y Hsiang-Tsung Kung. «GPSR: Greedy perimeter stateless routing for wireless networks». *Proceedings of the 6th annual international conference on Mobile computing and networking*. 2000, págs. 243-254.

[15] Prosenjit Bose, Pat Morin, Ivan Stojmenović y Jorge Urrutia. «Routing with guaranteed delivery in ad hoc wireless networks». *Wireless networks* 7.6 (2001), págs. 609-616.

[16] Jie Gao, Leonidas Guibas, John Hershberger, Li Zhang y An Zhu. «Discrete mobile centers». *Proceedings of the seventeenth annual symposium on Computational geometry*. 2001, págs. 188-196.

[17] Susanta Datta, Ivan Stojmenovic y Jie Wu. «Internal node and shortcut based routing with guaranteed delivery in wireless networks». *Cluster computing* 5.2 (2002), págs. 169-178.

[18] Fabian Kuhn, Rogert Wattenhofer y Aaron Zollinger. «Asymptotically optimal geometric mobile ad-hoc routing». *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*. 2002, págs. 24-33.

[19] Jorge Urrutia. «Routing with guaranteed delivery in geometric and wireless networks». *Handbook of wireless networks and mobile computing* 18 (2002), págs. 393-406.

[20] Fabian Kuhn, Rogert Wattenhofer, Yan Zhang y Aaron Zollinger. «Geometric ad-hoc routing: Of theory and practice». *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. 2003, págs. 63-72.

[21] Fabian Kuhn, Rogert Wattenhofer y Aaron Zollinger. «Ad-hoc networks beyond unit disk graphs». *Proceedings of the 2003 joint workshop on Foundations of mobile computing*. 2003, págs. 69-78.

[22] Fabian Kuhn, Rogert Wattenhofer y Aaron Zollinger. «Worst-case optimal and average-case efficient geometric ad-hoc routing». *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. 2003, págs. 267-278.

[23] Xiang-yang Li, Gruia Calinescu, Peng-jun Wan y Yu Wang. «Localized Delaunay Triangulation with Application in Ad Hoc Wireless Networks». *IEEE Transactions on Parallel and Distributed Systems* 14 (2003), pág. 2003.

[24] Fabian Kuhn, Thomas Moscibroda y Rogert Wattenhofer. «Unit disk graph approximation». *Proceedings of the 2004 joint workshop on Foundations of mobile computing*. 2004, págs. 17-23.

[25] Fabian Kuhn, Thomas Moscibroda y Rogert Wattenhofer. «What cannot be computed locally!» *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*. 2004, págs. 300-309.

[26] Thomas Moscibroda, Regina O'Dell, Mirjam Wattenhofer y Rogert Wattenhofer. «Virtual coordinates for ad hoc and sensor networks». *Proceedings of the 2004 joint workshop on Foundations of mobile computing*. 2004, págs. 8-16.

[27] Kay Romer y Friedemann Mattern. «The design space of wireless sensor networks». *IEEE wireless communications* 11.6 (2004), págs. 54-61.

[28] Karim Seada, Ahmed Helmy y Ramesh Govindan. «On the effect of localization errors on geographic face routing in sensor networks». *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*. IEEE. 2004, págs. 71-80.

[29] Jie Gao, Leonidas J Guibas, John Hershberger, Li Zhang y An Zhu. «Geometric spanners for routing in mobile networks». *IEEE journal on selected areas in communications* 23.1 (2005), págs. 174-185.

[30] Young-Jin Kim, Ramesh Govindan, Brad Karp y Scott Shenker. «Geographic routing made practical». *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. 2005, págs. 217-230.

[31] Young-Jin Kim, Ramesh Govindan, Brad Karp y Scott Shenker. «On the pitfalls of geographic face routing». *Proceedings of the 2005 joint workshop on Foundations of mobile computing*. 2005, págs. 34-43.

[32] Ben Leong, Sayan Mitra y Barbara Liskov. «Path vector face routing: Geographic routing with local face information». *13TH IEEE International Conference on Network Protocols (ICNP'05)*. IEEE. 2005, 12-pp.

[33] Tim Nieberg y Johann Hurink. «A PTAS for the minimum dominating set problem in unit disk graphs». *International Workshop on Approximation and Online Algorithms*. Springer. 2005, págs. 296-306.

[34] Mirjam Wattenhofer, Roger Wattenhofer y Peter Widmayer. «Geometric routing without geometry». *International Colloquium on Structural Information and Communication Complexity*. Springer. 2005, págs. 307-322.

[35] Aaron Zollinger. «Networking unleashed: Geographic routing and topology control in ad hoc and sensor networks». Tesis doct. ETH Zurich, 2005.

[36] Fabian Kuhn, Thomas Moscibroda y Roger Wattenhofer. «The price of being near-sighted». *SODA*. Vol. 6. 10.1145. Citeseer. 2006, págs. 1109557-1109666.

[37] Ben Leong, Barbara Liskov y Robert Tappan Morris. «Geographic Routing Without Planarization.» *NSDI*. Vol. 6. 2006, pág. 25.

[38] Tim Nieberg. *Independent and dominating sets in wireless communication graphs*. University of Twente, Enschede, Netherlands, 2006.

[39] Jorge Urrutia. «Local solutions for global problems in wireless networks». *Journal of Discrete Algorithms* 5.3 (2007), págs. 395-407.

[40] Dorothea Wagner y Roger Wattenhofer. *Algorithms for sensor and ad hoc networks: advanced lectures*. Vol. 4621. Springer, 2007.

[41] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara y Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Vol. 501. John Wiley & Sons, 2009.

[42] Franco P Preparata y Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.

[43] Ramesh K Jallu, Prajwal R Prasad y Gautam K Das. «Distributed construction of connected dominating set in unit disk graphs». *Journal of Parallel and Distributed Computing* 104 (2017), págs. 159-166.