
# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

## Un Algoritmo Local para el Problema del Árbol Generador de Peso Mínimo

Que para optar por el grado de:
Maestra en ciencias (computación)

Presenta:
Marcela Concepción Soriano Orozco

Tutor:
Dr. Jorge Urrutia Galicia
Instituto de Matemáticas

Co-Tutor:
Dr. José David Flores Peñaloza
Facultad de Ciencias

MÉXICO, D. F. JUNIO 2014

A mi familia y amigos, les dedico este trabajo ya que gracias a su apoyo incondicional y comprensión he podido culminar este proyecto. En especial a mis padres, Raúl e Hilda, quienes han sido los cimientos para que yo pudiese llegar a esta étapa y culminarla.

**Agradecimientos**

Este trabajo no se habría podido realizar sin la colaboración de muchas personas que me han brindado su ayuda, sus conocimientos y su apoyo. A todos ellos les doy mi más sentido reconocimiento y agradecimiento por su tiempo y dedicación para que este trabajo saliera adelante de la mejor manera posible.

Quiero agradecer profundamente a mi tutor, Dr. Jorge Urrutia, por la colaboración, paciencia, apoyo y sobre todo el gran conocimiento que compartió para la realización de este trabajo.

Un agradecimiento especial para mi cotutor, Dr. David Flores y para el Dr. Carlos Velarde por sus valiosos conocimientos, sugerencias, colaboración y guía para la realización de este trabajo.

# Índice general

# Índice de figuras

# Índice de cuadros

# Capítulo 1

# Introducción

## 1.1. El Contexto

Una red de sensores consiste en un conjunto de sensores autónomos que monitorean condiciones físicas o ambientales, tales como temperatura, sonido, presión, etc. La comunicación entre este conjunto de nodos de sensores se lleva a cabo de forma inalámbrica, con la cual permite formar redes ad hoc, es decir, redes sin infraestructura física preestablecida ni administración centralizada. Las redes de sensores y redes ad hoc, se caracterizan por tener un alto grado de movilidad, lo cual hace que este tipo de redes sean naturalmente dinámicas. [18].

Una red geométrica [1] es una red en la cual los nodos son representados por puntos en un plano, y las aristas son segmentos de líneas rectas que conectan a dos de estos. En los últimos años, se ha trabajado en problemas algorítmicos en redes geométricas, debido a que en muchas aplicaciones de la vida real, los nodos de una red se encuentran en posición fija en el plano, o bien, ésta puede ser obtenida continuamente mediante un GPS[2]. Por lo anterior, las redes geométricas son ideales para modelar redes en las que la posición es conocida, tal es el caso de las redes ad hoc, redes celulares y redes de sensores [17].

Muchos algoritmos funcionan bajo la suposición de que se cuenta con información global y centralizada sobre la topología de una red. Para los fines de este trabajo, no se considerarán los algoritmos de éste tipo, ya que el uso de la memoria y el mantenimiento de información actualizada (cuando la red cambia de forma dinámica) es

[1] También conocida como gráfica geométrica.
[2] Por sus siglas en inglés Global Position System

demasiado costoso [18]. Como consecuencia, la aplicación de algoritmos con conocimiento global de la red resulta ser ineficiente para este tipo de escenarios [15]. Más aún, la construcción y mantenimiento distribuido (o local por nodo) es preferible debido a los recursos limitados y movilidad de los nodos [18].

El crecimiento del tiempo de ejecución de un algoritmo global, típicamente está relacionado con el tamaño de sus datos de entrada [16], es decir, toma más tiempo encontrar el camino más corto en una gráfica con mil nodos, que en una gráfica con cien nodos.

Existe una alternativa a los algoritmos globales o de conocimiento centralizado, es decir, existen problemas que pueden ser resueltos sin la necesidad del conocimiento general de la red.

Un algoritmo local es un algoritmo distribuido que se ejecuta en un número constante de rondas de comunicación, independientemente del número de nodos que existan en la red [16]. Un algoritmo local toma decisiones basado en información local, es decir, solo considera información de todos sus vecinos a $k$-saltos de distancia; en general para muchos algoritmos locales $k$ es uno o dos [18]. Con lo cual, la garantía de eficiencia y el tiempo de ejecución de un algoritmo local, es independiente del número de nodos en la red [16].

Los algoritmos locales no sólo son altamente escalables, sino también tolerantes a fallas y a cambios sobre la topología de la red. Debido a ésto, los algoritmos locales son empleados para mantener una solución factible en redes altamente dinámicas [16].

A lo largo de los años se han propuesto diferentes aproximaciones de algoritmos locales que resuelven problemas bien conocidos tales como el problema de ruteo [5, 11], el problema de obtener una gráfica plana inmersa [18], el problema de calcular un conjunto dominante [17, 6], el problema de calcular el diagrama de Voronoi [17, 18], el problema de calcular una gráfica generadora [18, 8], el problema de obtener un árbol generador mínimo [aqui falta referencia al algoritmo de urrutia de la arista de entrada al poligono], por mencionar algunos.

Calcular el árbol generador de peso mínimo es uno de los problemas típicos y más conocidos en optimización combinatoria, además es de gran interés debido a que dicho problema puede resolverse de manera eficiente en tiempo polinomial, lo cual hace que sea aplicable directamente en diferentes áreas como redes de datos, telecomunicaciones, sistemas de agua, redes de transportación, redes eléctricas y procesamiento de imágenes, por mencionar algunas [9].

## 1.2. El Problema

Aunque las principales aproximaciones al problema del árbol generador de peso mínimo son los algoritmos de Kruskal [12] y Prim [14], los orígenes del problema hacen referencia al primer algoritmo propuesto por Otakar Borůvka [3, 4, 13]. Los tres algoritmos pueden clasificares como algoritmos *greedy* que obtienen una solución óptima y se ejecutan en tiempo polinomial. Estos tres algoritmos han jugado un papel muy importante en la historia del problema, no obstante existe una gran cantidad de variantes de los algoritmos cuya gran diferencia recae en la implementación [9].

El problema de estos tres algoritmos es que tienen una naturaleza centralizada y suponen el conocimiento general de la gráfica, por lo que calcular el árbol generador de peso mínimo para una red ad hoc o red de sensores puede ser bastante ineficiente, debido a que todo el computo de la red se concentra en un solo nodo. Por otra parte, tomando en cuenta que la red cambia constantemente en el tiempo, mantener actualizados los datos centralizados sobre la topología de la gráfica agrega carga en los canales de comunicación.

## 1.3. La aproximación

Dada una red geométrica plana $G$, puede calcularse el árbol generador de peso mínimo $T$ de manera local, es decir, no es necesario conocer en su totalidad la topología de la red para determinar las aristas que conforman al árbol generador de peso mínimo.

Este trabajo presenta un algoritmo local para determinar si una arista $e$ pertenece al conjunto de las aristas que conforman el árbol generador mínimo de una gráfica geométrica plana. Es fácil ver que a partir de este algoritmo se puede calcular de manera distribuida el árbol generador de peso mínimo.

Sea $e \in G$ una arista, la pregunta ¿Es $e$ una arista del árbol generador de peso mínimo?, se contesta localmente considerando sólo la información de los vecinos de $e$, y toma tiempo $O(n)$.

# Capítulo 2

# Marco Teórico

En este capítulo se presenta una breve introducción a teoría de gráficas y algoritmos locales.

## 2.1. Teoría de Gráficas

Muchas situaciones del mundo real pueden ser representadas convenientemente como un diagrama, consistente de un conjunto de puntos y líneas que relacionan ciertos pares de ellos. Una abtracción matemática de situaciones de este tipo da pie al concepto de gráfica [2].

Una gráfica $G$ se define como un par ordenado $(V, E)$, que consiste de un conjunto $V$ de vértices y un conjunto $E$ de aristas, juntos con una función de incidencia $\psi_G$ que asocia cada arista de $G$ con un par no ordenado de vértices (no necesariamente distintos) de $G$ [2].

Sea $e$ es una arista y $u, v$ dos vértices tales que $\psi_G(e) = \{u, v\}$ entonces se dice que $e$ une a $u$ y $v$, y los vértices $u$ y $v$ son llamados los extremos de $e$ [2].

Las gráficas son también así nombradas porque pueden ser representadas gráficamente mediante puntos que representan vértices y líneas (aristas) que los interconectan, lo cual ayuda a entender muchas de sus propiedades [2].

Se dice que los vértices extremos en una arista son incidentes a ella misma, y viceversa. Dos vértices que son incidentes a una arista en común, son adyacentes a dos aristas que son incidentes a un mismo vértice, y a dos vértices adyacentes

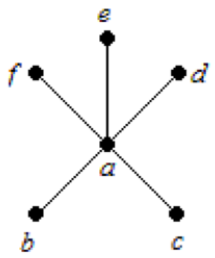


Figura 2.1: Grados de un vértice

distintos, se les llama vecinos. El conjunto de vecinos de un vértice $v$ en una gráfica $G$ se denota como $N_G(v)$ [2].

Una gráfica es simple si no contiene aristas paralelas, es decir, que exista dos o mas aristas cuyo par de vértices sea el mismo, ni contiene aristas blucle, la cual es una arista que va de un vértice y finaliza en éste mismo [2].

Una gráfica *completa*, es una gráfica simple en la que cualesquiera dos vértices son adyacentes [2].

**Definición 1.** *(grado de un vértice)*
*Se le llama el grado de un vértice al número de aristas que inciden al vértice. El grado de un vértice $v$ se denota por $grado_G(v)$ o $grado(v)$.*

**Definición 2.** *(grado máximo) (grado mínimo)*
*El grado máximo entre todos los vértices de una gráfica $G$ se denota por $\Delta(G)$. El grado mínimo se denota por $\delta(G)$*

Para la gráfica de la figura 2.1, se tiene que:
$grado(a) = 5$, $grado(b) = 1$, $grado(c) = 1$, $grado(d) = 1$, $grado(e) = 1$, $grado(f) = 1$.

El grado máximo es $\Delta(G) = 5$.

El grado mínimo es $\delta(G) = 1$.

**Definición 3.** *(vértice par o impar)*
*Un vértice $v$ es llamado par o impar de acuerdo a si $grado(v)$ es par o impar.*

**Definición 4.** *(subgráfica) Una gráfica $H$ es una subgráfica de una gráfica $G$ si $V(H) \subseteq V(G)$ y $A(H) \subseteq A(G)$ [2].*

**Definición 5.** *(supergráfica) Cualquier gráfica $G'$ para la cual $G$ es una subgráfica, es llamada una supergráfica de $G$ [2].*

**Definición 6.** *(subgráfica generadora)*
*Una subgráfica generadora $H$ de $G$ es cualquier subgráfica que tenga el mismo número de vértices.*

**Definición 7.** *(subgráfica inducida)*
*Una subgráfica inducida $H$ de $G$ es aquella que cumple que $V(H) \subseteq V(G)$ y que $A(H)$ es el conjunto de todas las aristas de $G$ entre vértices de $H$.*

### 2.1.1. Caminos, paseos, trayectorias y ciclos.

**Definición 8.** *(camino)*
*Sea $v_0, e_0, v_1, e_1, v_2, e_2, ..., v_{n-1}, e_{n-1}, v_n$ la secuencia de aristas y vértices que denotan un camino en $G$, donde $v_i$ hace referencia al $i$-ésimo vértice del camino, y $e_i$ denota la $i$-ésima arista del mismo.*

Se dice que dos vértices están conectados si existe un camino que vaya de uno a otro, de lo contrario estarán desconectados. Dos vértices pueden estar conectados por varios caminos. El número de aristas dentro de un camino es su longitud.

**Definición 9.** *(paseo)*
*Un paseo en una gráfica es un camino en el que no repite aristas.*

**Definición 10.** *(trayectoria)*
*Una trayectoria es un camino o (paseo) en el que no se repiten vértices.*

**Definición 11.** *(circuito)*
*Un paseo en el cual $u = v$, y que contiene al menos tres aristas es llamado circuito. Un circuito debe terminar en el mismo vértice en el que se empezó.*

**Definición 12.** *(ciclo)*
*Un circuito en el que no repiten vértices (excepto el primero y el último) es llamado ciclo.*

La longitud de un camino o un ciclo, es el número de sus aristas. Un camino o ciclo de longitud $k$ es llamada un $k - camino$ o $k - ciclo$, respectivamente.

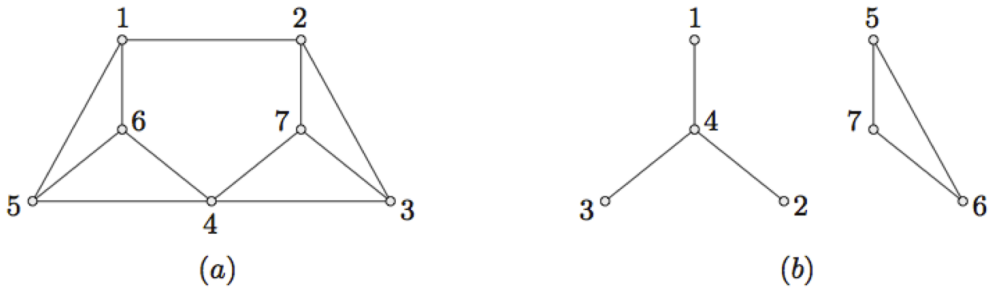


Figura 2.2: (a) Gráfica conexa, y (b) una gráfica no conexa

### 2.1.2. Conexidad

**Definición 13.** *(vértices conectados)*
*Dos vértices $u$ y $v$ en una gráfica $G$ están conectados si $u = v$ o si $u \neq v$ y existe un camino de $u$ a $v$ en $G$.*

**Definición 14.** *(gráfica conexa)*
*Una gráfica $G$ es conexa si dados cualesquiera dos vértices, siempre están conectados.*

**Definición 15.** *(gráfica disconexa)*
*Una gráfica $G$ es disconexa si existen en ella dos vértices no conectados.*

**Definición 16.** *(componente conexa)*
*Una subgráfica conexa $H$ de una gráfica $G$, es una componente de $G$ si $H$ no es una subgráfica propia de una subgráfica conexa de $G$.*

Una gráfica es conexa si y solo si el número de sus componentes es uno. Denotaremos por $C(G)$ a las componentes de $G$.

**Definición 17.** *(vértice de corte)*
*Sea $G$ una gráfica, se dice que $v$ es un vértice de corte de $G$ si $C(G-v) > C(G)$.*

**Definición 18.** *(arista de corte)*
*Sea $G$ una gráfica, se dice que $a$ es una arista de corte (o puente) de $G$ si $C(G-a) > C(G)$.*

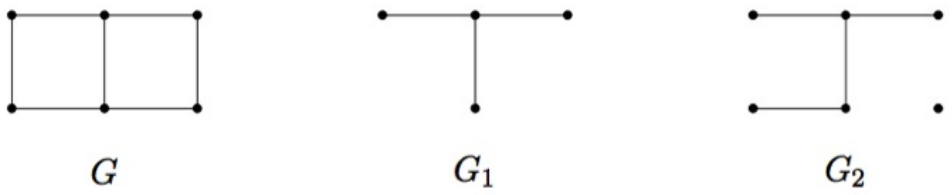


Figura 2.3: (a) $G_1$ y $G_2$ son subgráficas de $G$, pero sólo $G_2$ es expansiva

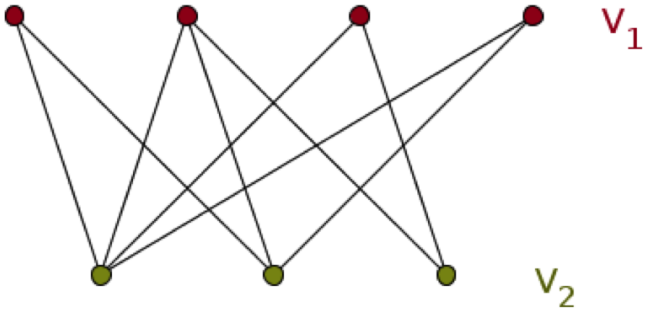


Figura 2.4: Gráfica bipartita

**Definición 19.** *(gráfica expansiva)*
*Una subgráfica $G'$ de una gráfica $G$ es una gráfica que tiene todos sus vértices y aristas en $G$, de manera que toda arista de $G'$ incida en vértices de $G$. Se dice que una subgráfica es expansiva cuando contiene todos los vértices de la gráfica de partida.*

**Definición 20.** *(gráfica bipartita)*
*Una gráfica $G$ es bipartita si existe una partición de los vértices de $G$ en dos conjuntos (no vacíos) $V_1$ y $V_2$ de forma que cada arista de $G$ tenga un extremo en $V_1$ y el otro en $V_2$*

En la figura 2.4 se presenta una gráfica bipartita.

**Definición 21.** *(árbol)*
*Sea $G$ una gráfica, se dice que $G$ es un árbol si $G$ es una gráfica conexa y no tiene ciclos.*

**Definición 22.** *(bosque)*
*Sea $G$ una gráfica, se dice que $G$ es un bosque si $G$ es una gráfica que no contiene ciclos.*

### 2.1.3. Planaridad

**Definición 23.** *(gráfica aplanable)*
*Una gráfica aplanable es una gráfica que puede ser dibujada sobre el plano de tal manera que ninguna arista se cruce con otra excepto en los vértices.*

**Definición 24.** *(gráfica plana)*
*Una gráfica plana es aquella gráfica aplanable dibujada sobre el plano de tal manera que las aristas no se intercectan.*

*Sea G=(V,E) una gráfica plana*

*Sea $v_0, e_0, v_1, e_1, v_2, e_2, ..., v_{n-1}, e_{n-1}, v_n$ la secuencia de aristas y vértices que denotan un camino en G, donde $v_i$ hace referencia al i-ésimo vértice del camino, y $e_i$ denota la i-ésima arista del mismo.*

**Definición 25.** *(caras de una gráfica)*
*Una cara (o región) de una gráfica aplanable se define como una área del plano que está acotada por aristas y no se puede continuar dividiéndose en subáreas.*

Si una gráfica conexa y plana se traza en el plano, el plano se divide en caras contiguas. Una cara queda caracterizada por el ciclo que forma su *frontera*.

En una gráfica plana conexa podemos definir dos tipos de caras, las *caras internas* y la *cara externa*. Una cara es interna si su región queda totalmente acotada por su frontera (un conjunto de aristas que forman un ciclo). La única región no acotada del plano es denominada cara externa.

### 2.1.4. Gráficas Geométricas

**Definición 26.** *(gráfica geométrica)*
*Una gráfica geométrica es una gráfica en la cual los vértices o nodos son representados por puntos en el plano, y las aristas son segmentos de líneas rectas que conectan a dos de estos.*

Una gráfica geométrica también es conocida como red geométrica. El término que se utilizará en éste trabajo será el de red geométrica.

En años recientes se ha prestado mucha atención al estudio de problemas algorítmicos en redes geométricas, debido a que en muchas aplicaciones de la vida real los nodos de una red se encuentran en una posición fija en el plano, o esta puede ser obtenida continuamente por GPS. Por lo tanto, las redes geométricas son ideales

para modelar redes en las que la posición es conocida, tal como redes ad hoc, redes celulares y redes de sensores. [1]

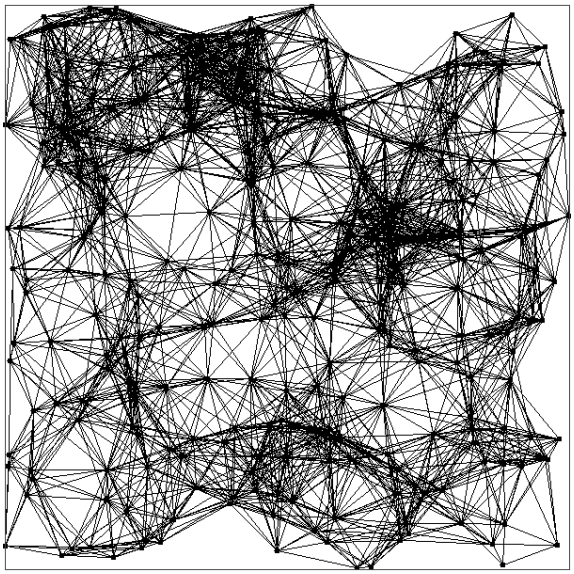

Figura 2.5: Gráfica geométrica no plana. Cada nodo *conoce* sus coordenadas de posición en el plano.

El identificador único de cada vértice en una red geométrica puede estar dado por su posición en el plano, supondremos que no tenemos dos vértices sobre un mismo punto, en [2] se especifica un orden parcial de los nodos de una red basado en la posición, de la siguiente manera: Dados dos procesadores $p$,$q$,

$$p < q, si\ p_y < q_y,\ o\ si\ p_y = q_y\ entonces\ p_x < q_x$$

Bajo el orden parcial definido note que dos procesadores no pueden compartir un identificador, por lo que cualesquiera dos procesadores son comparables.

[1] J. Urrutia. Local Solutions For Global Problems in Wireless Networks, 2006
[2] S. Rajsbaum, J. Urrutia, Some Problems in Distributed Computational Geometry, 2012

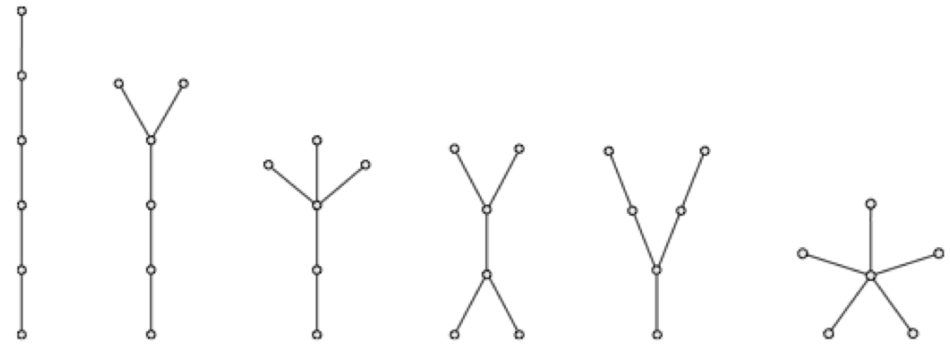

Figura 2.6: Árboles de seis vértices.

## 2.2. Árboles

### 2.2.1. Bosques y Árboles

Recordemos que una gráfica *acíclica* es aquella que no contiene ciclos. Un árbol es una gráfica en la que cualesquiera dos vértices están conectados por exactamente un único camino simple. De acuerdo a ésta definición, cada componente de una gráfica acíclica es un árbol. Por ésta razón, las gráficas acíclicas son usualmente llamadas *bosques*. Un bosque es una unión desjunta de árboles.

Las condiciones siguientes son equivalentes, lo que quiere decir que si se cumple una, las demás también se cumplirán.

- 1. $G$ es conexa y no tiene ciclos.
- 2. $G$ no tiene ciclos y, si se añade alguna arista se forma un ciclo.
- 3. $G$ es conexa y si se le quita alguna arista deja de ser conexa.
- 4. Dos vértices cualesquiera de $G$ están conectados por un único camino simple.

Si un árbol tiene un número finito de vértices $n$, entonces tiene $n - 1$ aristas.

**Definición 27.** *(árbol dirigido)*
*Un árbol dirigido es una gráfica dirigida que sería un árbol si no se consideraran las direcciones de las aristas.*

**Definición 28.** *(árbol con raíz)*
*Un ábol recibe el nombre de árbol con raíz si a cada vértice le ha sido asignado una raíz, en cuyo caso las aristas tienen una orientación natural hacia o desde la raíz.*

Un árbol con raíz $T(x)$ es un árbol $T$ con un vértice $x$ específico, llamado la raíz de $T$.

**Definición 29.** *(árbol etiquetado)*
*Un árbol etiquetado es un árbol en el que cada vértice tiene una única etiqueta. Los vértices de un árbol etiquetado de n vértices reciben normalmente las etiquetas* $\{1, 2, ..., n\}$

**Definición 30.** *(árbol regular)*
*Un árbol regular u homogéneo es un árbol en el que cada vértice tiene el mismo grado.*

A continuación se da la prueba de la equivalencia de las condiciones (1) y (4):

**Teorema 1.** *$T$ es un árbol si, y sólo si, todo par de vértices de $T$ está unido por un solo camino.*

**Prueba.** Sea $T$ un árbol. Como, por definición, $T$ es conexo, entonces existe un camino elemental entre todo par de vértices de $T$. Ahora bien, este camino es único: si $u$ y $v$ son dos vértices cualesquiera de $T$ y suponemos que existen dos caminos distintos de $u$ a $v$, al unir dichos caminos se formaría un subgrafo de $T$ que contendría un ciclo, debido a que existen dos caminos diferentes, hecho que nos lleva a una contradicción. Recíprocamente, si todo par de vértices de $T$ está unido por un solo camino elemental, obviamente $T$ es un grafo conexo y, además, $T$ no contiene ciclos porque si existiera un ciclo todos los pares de vértices en él se podrían unir con dos caminos distintos. □

### 2.2.2. Árboles Generadores

**Definición 31.** *(subárbol)*
*Un subárbol de una gráfica es una subgráfica que es un árbol.*

**Definición 32.** *(árbol generador)*
*Si subárbol es una subgráfica generadora, entonces se le llama árbol generador de la gráfica.*

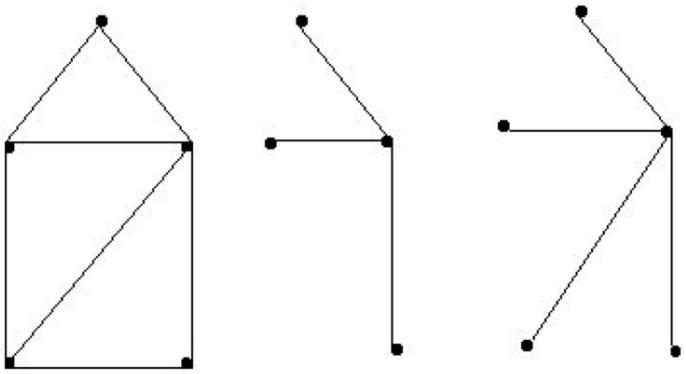

Figura 2.7: Gráfica $G$ - Árbol de la gráfica $G$ - Árbol generador de $G$

La figura 2.7 se muestra un ejemplo de un árbol generador.

**Definición 33.** *(rama de un árbol)*
*Una rama de un árbol es una arista de la gráfica que es un árbol.*

**Definición 34.** *(enlace de un árbol)*
*Un enlace o cuerda de un árbol es una arista de la gráfica que no está en el árbol.*

**Definición 35.** *(complemento de un árbol)*
*Se le llama complemento del árbol al conjunto de enlaces o cuerdas de un árbol.*

Una gráfica conexa siempre contiene un árbol generador. Si una gráfica es conexa y no contiene ciclos entonces es un árbol. Si la gráfica contiene uno o más ciclos, se puede eliminar una arista de los ciclos y aún tener una subgráfica conexa.

Si una gráfica $G$ tiene un árbol generador $T$, entonces $G$ es conexa porque cualesquiera dos vértices de $G$ están conectados por un camino en $T$, y por lo tanto en $G$. Por el otro lado, si $G$ es una gráfica conexa la cual no es un árbol, y $e$ es una arista de un ciclo de $G$, entonces $G \setminus e$ es una subgráfica generadora de $G$, la cual es también conexa, por la proposición 2, $e$ no es una arista de corte de $G$. Mediante la repetición de éste proceso, de ir eliminando aristas en los ciclos hasta que cada arista que se mantiene, es una arista de corte, entonces obtenemos un árbol generador de $G$. Así, tenemos el siguiente teorema, que proporciona otra caracterización de las gráficas conexas.

**Teorema 2.** *Una gráfica es conexa si y sólo si, tiene un árbol generador.*

**Definición 36.** *(conjunto de corte)*
*Un conjunto de corte es un conjunto (mínimo) de aristas en una gráfica tal que la eliminación del conjunto incrementa el número de componentes conexas en la subgráfica restante, en tanto que la eliminación de cualquier subconjunto propio de éste, no lo haría.*

De la definición 36 se tiene que en una gráfica conexa, la eliminación de un conjunto de corte divide a la gráfica en dos partes.

Una gráfica $G$ puede tener varios árboles generadores diferentes. Podemos asignarle a cada arista de $G$ un número llamado *peso*, el cual, es un número que representa el valor que tiene una arista. El peso de un árbol generador es la suma de los pesos de las ramas del árbol.

### 2.2.3. Árbol Generador de Peso Mínimo

**Definición 37.** *(árbol generador de peso mínimo)*
*Un árbol generador de peso mínimo es un árbol generador de peso, menor o igual al de cualquier otro árbol generador de la gráfica.*

Cualquier gráfica no dirigida (no necesariamente conexa) tiene un *bosque generador mínimo*, el cual, es la unión de los árboles generadores de peso mínimo para sus componentes conexas.

## 2.3. Algoritmos Locales

**Definición 38.** *(algoritmo local)*
*Un Algoritmo local es un algoritmo distribuido que se ejecuta en un número constante de rondas de comunicación, independientemente del número de nodos o vértices que existan en la red.*[3]

Un algoritmo local toma decisiones basado en información local, es decir, en un nodo solo considera información de todos sus vecinos a $k$-saltos de distancia de él; en general, para muchos algoritmos locales $k$ es uno o dos, [4] con lo cual la garantía

[3]Jukka Suomela, Survey of Local algorithms, 2013
[4]D. Wagner, R. Wattenhofer. Algorithms for Sensor and AdHoc Networks. 2007

de eficiencia y el tiempo de ejecución de un algoritmo local es independiente del número de nodos en la red.

Los algoritmos locales no solo son altamente escalables, sino también tolerantes a fallas y cambios sobre la topologá de la red. Debido a esto, los algoritmos locales son empleados para mantener una solución factible en redes altamente dinámicas

**Aproximación local**: Un algoritmo de que produce la salida mas factible. y la utilidad de la salida.

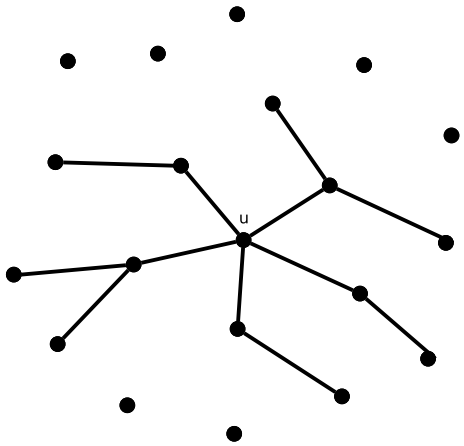


Figura 2.8: Vecindad de un nodo $u$, tal que $k$ es a lo mas distancia dos.

Resulta que si los nodos de una gráfica geométrica plana conocen su posición en el plano, muchos problemas pueden ser resueltos con algoritmos locales, tales como ruteo, en el cual un agente desea viajar de un nodo $u$ a un nodo $v$ solo recordando una cantidad constante de información y los nodos de inicio y fin, es decir, no es necesario conocer la topología en su totalidad, no está permitido dejar marcas en los vértices de la gráfica, vea [5], [6] tal que información global de la red, como tablas de ruteo o la cantidad de nodos en la red, no es necesaria.

Una gráfica se dice plana, si se puede dibujar en el plano de tal forma que no tenga dos aristas que se crucen mas que en los puntos finales de ellas (vértices), una gráfica que no cumple con ésta característica tiene una gráfica plana inmersa la

[5] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia. Routing with Guaranteed Delivery in AdHoc Networks, 1999

[6] E. Kranakis, H. Singh, J. Urrutia. Compass Routing on Geometric Networks. 1999

cual generalmente se calcula con el algoritmo de gráficas de Gabriel (del cual existe una versión local que se muestra en [7]). Dicho algoritmo garantiza la planaridad y conexidad de la gráfica resultante. Existen otros algoritmos locales que calculan una gráfica plana, tales como gráficas de Yao y gráficas de vecindario relativo, por mencionar algunos.

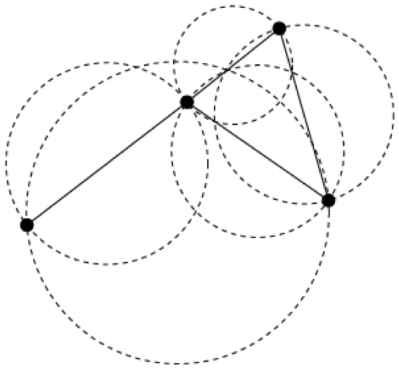

Figura 2.9: Gráfica de Gabriel.

Otro problema que puede resolverse con algoritmos locales es el problema del conjunto dominante, es decir, dada una gráfica con un conjunto de vertices $V(G)$, un conjunto dominante es un subconjunto $S \subset V(G)$ tal que cada vértice de $G$ esta en $S$ o es adyacente a un vértices en $S$. En [8] se presenta un algoritmo local para obtener un conjunto dominante en una red inalámbrica UDG.

### 2.3.1. ¿Qué Puede Ser Calculado De Manera Local?

No obstante, no todos los problemas en gráficas pueden ser resueltos de manera local. Intuitivamente, todo aquel problema que necesite información global de la red, o todo aquel cómputo que necesite almacenamiento proporcional al tamaño de la gráfica, se dice que es inherentemente no local. Un buen ejemplo de este tipo de problemas es la construcción de un árbol generador de peso mínimo, en el que es necesario el conocimiento global de la red para la solución, sin embargo, existen aproximaciones locales para resolverlo. El Cuadro 2.1 menciona una serie de problemas, que se ha demostrado no tienen solución mediante un algoritmo local determinístico.

[7] D. Wagner, R. Wattenhofer. Algorithms for Sensor and AdHoc Networks, 2007

[8] S. Funke, A. Kasselman, M. Segal, U. Meyer. A Simple Improved Distributed Algorithm For Minimum CDS in Unit Disk Graphs, 2005

| Problema | Familia de la Gráfica |
|---|---|
| Cojunto Maximal Independiente | Ciclo |
| Apareamiento Maximal | Ciclo |
| 3-Coloreo de Vértices | Ciclo |
| $\Delta$-Coloreo de Vértices | Árbol $(\Delta + 1)$-Coloreado |
| Coloreo de Aristas | Ciclo |
| Coloreo debil | $2k$-regular |

Cuadro 2.1: Lista de problemas no resueltos con algoritmos locales determinísticos.

Aunque algunos problemas no puedan ser calculados localmente, se ha trabajado sobre algoritmos locales que entregan soluciones aproximadas a la óptima, en [9] se muestra una cota mínima de aproximación para el problema del conjunto mínimo de cobertura de vértices. Dicho resultado se extiende también para el problema del mínimo conjunto dominante.

Existen más aproximaciones locales para una amplia variedad de problemas conocidos tales como, aproximaciones a árboles generadores de peso mínimo euclidiano, diagramas de Voronoi, gráficas generadoras, estas ultimas están muy cercanamente relacionadas con la planaridad, factor de estiramiento constante y un número lineal de aristas (con respecto a los vértices) de una gráfica geométrica.

Como puede observar, el campo de los algoritmos locales, ha tomado gran auge a raíz de la aplicabilidad en problemas de la vida real. También bajo la motivación de encontrar algoritmos cuya complejidad y eficiencia no dependen del tamaño de la red. Por otro lado, las redes geométricas son cada vez mas empleadas, debido a los avances tecnológicos en la reducción del tamaño de los procesadores y la fiabilidad de los sistemas de posicionamiento globales. Mas aún, las investigaciones han encontrado una gran cantidad de preguntas abiertas, que llaman la atención a dicho campo. En [10] se propone como trabajo a futuro, una adaptación (en el mejor de los casos) de los algoritmos de ruteo, o una propuesta para lidiar con el problema de ruteo sobre redes no planas. Debido a que el modelo de redes geométricas es aplicable a redes de sensores y redes ad-hoc, deben considerarse también factores como el ahorro de energía de los nodos, lo cual está relacionado a problemas clásicos como

[9] F. Kuhn, T. Moscibroda, R. Watternhofer, What Cannot Be Computed Locally, 2008

[10] J. Urrutia. Local Solutions for Global Problems in Wireless Networks, 2006

árboles generadores mínimos, diagramas de Voronoi y gráficas generadoras de factor de estiramiento constante, entre otras.

# Capítulo 3

# Trabajo Relacionado

Durante este capítulo se presentarán algunos algoritmos que se han estudiado en los últimos años para la solución del problema del Árbol Generador de Peso Mínimo.

En [1] se presenta un algoritmo que requiere de $O(E + VlogV)$ mensajes con un tiempo $O(V)$, donde se considera un modelo estándar de una red asíncrona, esto es, una comunicación de punto a punto descrita por una gráfica $G = (V, E)$ no dirigida, donde el conjunto de nodos $V$ representa los procesadores de la red, y el conjunto de aristas $E$ representa la comunicación entre ellos. Cada nodo tiene una copia de la ejecución de su algoritmo. El algoritmo especifica qué cálculos deben ser realizados y qué mensajes deben ser enviados. Así, el algoritmo comienza de forma independiente en cada nodos, y esto puede hacerlo en tiempos diferentes. Al comienzo, cada nodo ignora la topología de la red excepto por sus propias aristas. Al terminar el algoritmo, cada nodo tiene conocimiento de sus vecinos en el árbol generador mínimo.

El algoritmo consiste de 2 etapas, la primera la refieren como etapa de *conteo*. En esta etapa encuentra *algún* árbol generador y calcula el número de nodos en la red. La segunda etapa, la llaman etapa de *MST*, donde recibe como entrada el número de nodos en la red y, usando esa información, encuentra el Árbol Generador de Peso Mínimo. Ambas etapas requieren de $O(E + VlogV)$ mensajes, con un tiempo de ejecución de $O(V)$.

En la etapa de conteo, se utiliza una rutina para encontrar un árbol generador, no importa que no sea el de mínimo peso puesto que el objetivo principal de ésta etapa es obtener el número de nodos que contiene la gráfica.

El autor asume que todas las aristas están asignadas con pesos distintos y están

ordenadas conforme a éstos. Ésta condición es la que garantizará que existirá solo un árbol generador de peso mínimo [7].

Para la etapa llamada *MST*, se asume el conocimiento del conjunto de nodos $V$, el número total de nodos, el cual es obtenido en la etapa de *conteo*. La etapa MST es realizada en 2 fases. La primera fase ejecuta un algoritmo idéntico al presentado en [7], las nuevas ideas algorítmicas son presentadas hasta la fase 2, donde la primera solución fue presentada en [7] usando la técnica de *niveles*, la diferencia recae principalmente en que [1] actualiza los niveles de manera más precisa, lo cual impide que pequeños árboles esperen por árboles más grandes, ésto hace que el algoritmo se acelere. Éste método de actualización de niveles requiere de más mensajes, sin embargo no incrementa la complejidad de comunicación dado que el número de árboles es pequeño, a lo más $logV$. La razón principal por la cual, la etapa de conteo es necesaria es que sin el conocimiento del conjunto de vértices $V$, no se sabría cuándo la primera fase de la etapa MST terminaría. La innovación que presenta el autor en [1] a diferencia de [7] es que en lugar de actualizar un nivel por medio del conteo del número total de nodos, el nivel es actualizado localmente basado en el número estimado de nodos en un subárbol. Presenta 2 nuevos mecanismos para actualizar los niveles, llamados *Root-Update* y *Test-Distance*. A grandes rasgos, *Root-Update* incrementa el nivel de la raíz si dentro de determinado tiempo no se encuentra la mejor arista; *Test-Distance* incrementa el nivel de nodos en un subárbol cuya raíz está lo suficientemente lejos de la raíz del árbol resultante. La meta principal es garantizar que el periodo de tiempo en el cual un nivel es el nivel más pequeño en la red está acotado en a lo más $O(2^t)$.

Cuando un nodo $r$ llega a ser una raíz de un árbol $T$ con nivel $l$, transmite un mensaje de inicialización conteniendo a $r$ y $l$ como parámetros sobre $T$. Este mensaje a su vez es transmitido a los árboles que se conectan a sí mismos en un árbol $T$. Un nodo, al recibir el mensaje de inicialización, de acuerdo a aquellos parámetros en campos locales *Level(j)*, *Root(j)*, y comienza la ejecución de un procedimiento de búsqueda local. Este procedimiento encuentra la arista de peso mínimo saliente del nodo $j$ al nodo $i$ que no están $T$, o bien declara que no existe tal nodo, es decir, todas las aristas incidentes a $j$ son aristas internas en $T$.

Hacia ese objetivo, nodo $j$ explora sus aristas incidentes, aun sin saber que son aristas internas, en orden decreciente de sus pesos. Explorar una arista consiste de mandar un mensaje especial de prueba al nodo $k$ al otro lado de la arista y obtener la respuesta de $k$. La respuesta será negativa si *Root(k)* $= r$ (es decir, si $k \in T$) y es positiva en caso contrario. La propiedad importante del algoritmo es que solo los nodos con un nivel mayor o igual a $l$ responderán inmediatamente a tal mensaje.

Un nodo $k$ con *Level(k)* $< l$ tarda en responder a aquel mensaje hasta que *Level(k)* alcance $l$. Si este se nivel incrementa en $k$ es debido a la conexión del nodo $k$ al árbol $T$, entonces $k$ tendrá *Root(k)* $= r$; así la respuesta es negativa y la arista a $k$ es marcada en $j$ como *interna*. En este caso, el procedimiento de búsqueda se reanuda con la siguiente arista a ser explorada. De otra manera, si la respuesta es positiva, entonces la arista $(j, k)$ es declarada para ser la candidata local para la mejor arista de $T$.

El nombre de las candidatas locales son guardadas en la raíz. Si ninguno de los nodos del árbol pudo encontrar una candidata local, entonces el algoritmo termina, dado que el árbol genera la red completa y así, es un árbol generador mínimo. De otra manera, la raíz escoge a aquella arista con peso mínimo, digamos que la arista $(v, w)$ con $v$ siendo el extremo, como la mejor arista de $T$. Posteriormente, un mensaje especial viaja al otro extremo $v$ de esa arista, regresando todos los puntos padre a su camino y transformando $v$ en una raíz de $T$. Cuando $v$ recibe ese mensaje, éste actúa de la siguiente forma: Si la arista $(v, w)$ es una arista núcleo y $v$ es su extremo más grande, $v$ no se conecta a sí mismo, dado que es la raíz del árbol resultante; su nivel es incrementado en 1, y transmite su propio mensaje de inicialización sobre el árbol resultante. Este procedimiento continúa hasta que la etapa MST eventualmente termina.

El algoritmo presentado por [1] es un algoritmo distribuido para encontrar un $AGPM$, como comenta el autor, requiere de una etapa de *conteo*, la cual, permitirá obtener el conocimiento de todos los nodos de la red para posteriormente utilizar dicha información en la siguiente etapa, llamada $MST$. Como bien lo describeel autor, para encontrar el AGPM, es necesario tener el conocimiento de la red. El algoritmo propuesto en este trabajo de tesis, a diferencia del propuesto por [1], no es necesario tener el conocimiento de la topología de la red, ni de los nodos y aristas.

Otro algoritmo propuesto fue propuesto por [Li]:

Dado un vértice $v \in G$, sea $H(v, k)$ la subgráfica inducida de $G$ por los vértices de $G$ a distancia menor o igual a $k$ desde $v$, incluyendo el mismo $v$. Dada una constante $k$, se realiza lo siguiente:

Primero se define un orden lineal de aristas de $G$ como sigue: una arista $(u, v)$ es más pequeña que una arista $(r, s)$ si la longitud de $(u, v)$ es menor que la longitud de $(r, s)$. En el caso de que ambas tengan la misma longitud, se toma la arista con el vértice superior más a la derecha como la mayor. Si ambas tienen la misma longitud

y el mismo vértice más a la derecha, entonces se tomará vértice superior más a la izquierda como la mayor.

**Algoritmo 1** Algoritmo $LocalMST_k$

1: cada nodo $v$ calcula el árbol generador mínimo $T_{k,v}$ de la subgráfica de $H(v,k)$ (de acuerdo al orden lineal definido arriba).
2: una arista $uv$ de $G$ se mantiene si aparece en ambos $T_{k,u}$ y $T_{k,v}$.

En [Li, Localized Delaunay triangulation] se prueba que el peso de la subgráfica obtenida está dentro de un factor constante del árbol generador de peso mínimo de $H$. En [E.Chavez, S.dobrev, E. Kranakis. Local Construction of Plannar Spanners] se prueba que el peso de la solución obtenida es a lo más $\frac{k+1}{k-1}$ veces el de la solución óptima, $k \geq 2$.

# Capítulo 4

# La Propuesta

Supongamos que tenemos una red geometrica plana y conexa $G = (V, E)$, con un conjunto de nodos $V = \{v_1, v_2, ..., v_n\}$ y un conjunto de aristas $E = \{e_1, e_2, ..., e_m\}$, tal que cada arista es de la forma $e_k = (v_i, v_j)$.

Para cada elemento $e_k \in E$, se cuenta con un cierto peso asociado $c(e_k) > 0$. Por simplicidad suponga que todos los pesos de las aristas son diferentes.

Por conveniencia se desea controlar la topología de $G$ y construir la red menos costosa posible, con pocos enlaces pero que garantice la conexidad entre cualquier par de nodos.

El problema es encontrar un subconjunto de las aristas $T \subseteq E$ de modo que la gráfica $G' = (V, T)$ sea conexa y que el costo total $\sum_{e \in T} c_e$ sea el mínimo posible[10].

Una solución factible es obtener el árbol generador de peso mínimo de la red $G$, sin embargo dado el contexto del problema donde los nodos cuentan con características dinámicas de movilidad y conexidad, implementar un algoritmo para obtener el AGPM de manera centralizada sería ineficiente y costoso.

Este capítulo presenta un algoritmo local que calcula si una arista $e = (u, v)$ de la gráfica $G$ pertenece al AGPM. Note que cada vértice de la gráfica puede calcular localmente cada una de las aristas incidentes a él, y construir el AGPM de una forma distribuida.

## 4.1. El Algoritmo Local

Sea $G = (V, E)$ una gráfica plana conexa, tal que todos los pesos de las aristas son diferentes y sea $e = (u, v)$ una arista de $E$.

Considere el algoritmo de Kruskal, este construye un Árbol Generador de Peso Mínimo (AGPM) de una gráfica $G$. Debido a que todas las arista de $G$ tienen pesos diferentes se garantiza que solo existe un único AGPM [7].

El algoritmo de Kruskal toma las aristas de la gráfica y las ordena de manera creciente de acuerdo a su peso asociado, después toma la de menor peso y la agrega al AGPM de $G$ sólo si la arista no crea algún ciclo, en caso contrario elimina dicha arista. El algoritmo procede con la siguiente arista de acuerdo al orden de pesos, eventualmente el algoritmo agota las aristas y obtiene el AGPM de $G$.

La construcción del árbol puede verse de la siguiente manera. Se tienen $n$ componentes disjuntas de cadinalidad uno en vértices, en cada iteración del algoritmo se agrega una arista que une a dos componentes. Al final del algoritmo de Kruskal se tiene una componente de cardinalidad $n$, con $n-1$ aristas.

**Observación 1:** El algoritmo de Kruskal en cada iteración verifica que la arista que agrega no genera algún ciclo. Es decir, el algoritmo de Kruskal no agrega aristas que unan vértices de una misma componente.

De acuerdo con la Observación 1 puede demostrarse el siguiente lema.

**Lema 1.** *Cada arista $(u, v)$ que se agrega al AGPM de $G$ en una iteración del algoritmo de Kruskal, está formada por dos vértices $u \in A$ y $v \in B$, donde $A$ y $B$ son dos componentes diferentes.*

*Demostración.* Por contradicción supongamos que $(u, v)$ conecta a dos vértices tales que $u, v \in A$.

Al agregar la arista $(u, v)$ al AGPM de $G$ se unen dos vértices $u, v$. Pero los vértices $u, v$ pertenecen a la misa componente $A$. Por lo tanto existe un camino $P$ que los conecta. Más aún $P+(u, v)$ induce necesariamente un ciclo en la componente $A$. Pero $A \subseteq$ AGPM y un árbol no tiene ciclos. □

De acuerdo con la Observación 1 y el Lema 1 puede demostrarse el siguiente teorema.

**Teorema 3.** *La arista $(u, v) \in AGPM$ de $G$ si y sólo si, en la subgráfica de $G$ inducida por las aristas de peso menor que $c(u, v)$ no existe un camino $P = \{p_1, p_2, p_3, ..., p_k\}$, que va de $u$ a $v$.*

*Demostración.* Por contradicción, supongamos que existe un camino $P = \{p_1, p_2, p_3, ..., p_k\}$ en la subgráfica de $G$ inducida por las aristas de peso menor que $c(u, v)$.

Por el algoritmo de Kruskal, cada arista de $P$ fue agregada en una iteración del algoritmo, por lo tanto al menos han pasado $k$ iteraciones.

Sin pérdida de generalidad, supongamos que en la iteración $k + 1$ se escoge la arista $(u, v)$ para ser agregada al AGPM de $G$.

La arista $(u, v)$ une a los vértices $u$ y $v$. Por Lema 1, $u$ y $v$ necesariamente pertenecen a componentes diferentes. Pero existe $P$ que va de $u$ a $v$. Por lo tanto $u$ y $v$ ya pertenecían a una misma componente. Lo cual contradice el Lema 1. □

La idea principal del Algoritmo 2 se basa en el Teorema 3. Dada una arista $(u, v)$ se inicia un recorrido por el vértice $u$ sobre la subgráfica de $G$ inducida por las aristas de peso menor que $c(u, v)$. El recorrido (Algoritmo 3) puede terminar en el vértice inicial $u$ o en el vértice $v$. En caso de terminar en el vértice $u$ implica que no se encontró un camino que conduzca al vértice $v$, por lo tanto la arista $(u, v)$ pertenece al AGPM. En caso de finalizar en el vértice $v$ implica que el recorrido encontró un camino $P$ de aristas con pesos menores a $c(u, v)$ que va de $u$ a $v$, por lo tanto la arista $(u, v)$ no pertenece al AGPM.

**Algoritmo 2** Algoritmo local que determina si la arista $(u, v)$ pertenece al AGPM de $G$

1: Efectúa recorrido local a partir de la arista $(u, v)$, por la subgráfica $G'$ inducida por las aristas de peso menor que $c(u, v)$, aplicando el algoritmo 3.
2: **si** recorrido finaliza en $u$ **entonces**
3: **devolver** $(u, v) \in$ AGPM.
4: **si no**, **si** recorrido finaliza en $v$ **entonces**
5: **devolver** $(u, v) \notin$ AGPM.
6: **fin si**

El recorrido por la gráfica inducida $G'$ se lleva a cabo de manera local como lo muestra el Algoritmo 3, solo recordando la arista inicial y la arista actual del recorrido. El recorrido emplea la bien conocida Regla de la Mano Derecha [5], la cual afirma que es posible visitar todas las paredes de un laberinto, manteniendo su mano derecha sobre la pared mientras camina hacia adelante.

Tome en cuenta que la gráfica inducida $G'$ es una gráfica bidireccional no dirigida, por lo que una arista $(x, y)$ corresponden a dos aristas dirigidas del mismo peso que van en sentidos opuestos, del lado izquierda la arista dirigida $(x, y)$ y del lado derecha la arista $(y, x)$.

**Algoritmo 3** Algoritmo local de recorrido por una gráfica $G'$, a partir de una arista (u,v).

```
1: para toda arista (u, w) incidente a u tal que c(u, w) < c(u, v) hacer
2:    Sea (x, y) = (u, w).
3:    repetir
4:       Toma la arista (y, z) que forme el menor ángulo con la arista (x, y) en
         sentido de las manecillas del reloj tal que c(y, z) < c(u, v).
5:       (x, y) = (y, z).
6:       si y = u and (y, z) ≠ (w, u) entonces
7:          (x, y) = (y, x).
8:       fin si
9:    hasta que (y = u and (x, y) = (w, u)) or y = v
10:   si y = v entonces
11:      devolver v
12:   fin si
13: fin para
14: devolver u;
```

Puede ver que si cada nodo de una red ejecuta el Algoritmo 2 de manera independiente, entonces de manera distribuida se construye el árbol generador de peso mínimo de la gráfica $G$.

Note que el Algoritmo 3 considera tres casos que se dan de acuerdo al recorrido local. El primer caso se da cuando un recorrido inicia y finaliza en la misma arista (Figura 4.2). El segundo caso se da cuando el recorrido sigue un camino que va de $u$ a $v$ (Figura 4.3). Y el tercer caso se da cuando un recorrido regresa al vértice inicial pero a través de una arista diferente a la de inicio (Figura 4.5).

Considere la gráfica geométrica plana $G$ que muestra la Figura 4.1, puede observar que todas las arista tiene un peso asociado diferente. La Figura 4.1 muestra también el AGPM de la gráfica $G$, Las aristas sólidas forman el conjunto de aristas que pertenecen al AGPM, las aristas punteadas forman el conjunto excluído del AGPM.

Considere la arista $e$ de peso 5 que se muestra en la Figura 4.2. Suponga que se desea conocer si la arista $e$ pertenece o no al AGPM de $G$. El Algoritmo 2 lleva a cabo un recorrido por las aristas incidentes a $u$ cuyo peso sea menos que $c(u, v)$, en el ejemplo de la Figura 4.2 puede verse que el recorrido inicia por la arista de peso 1, el recorrido da vuelta por la misma arista ya que no se encuentran aristas

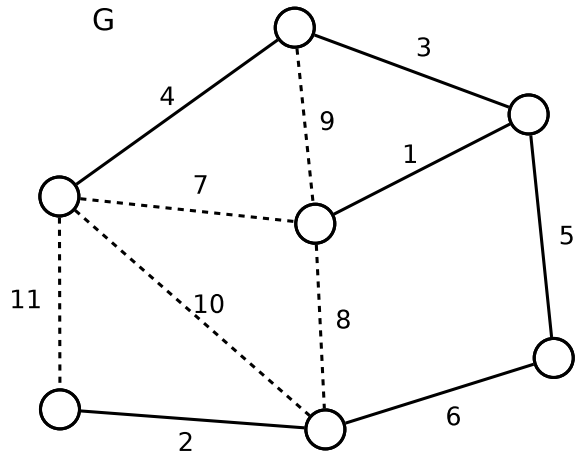


Figura 4.1: Gráfica $G$ geométrica plana, las lineas sólidas muestran las aristas que conforman al AGPM de $G$.

adyacentes con peso menor que $c(e)$. Finalmente el recorrido regresa al vértice $u$ por la arista de inicio, con lo cual se considera que el recorrido referente a la arista de peso 1 ha finalizado.

Un segundo recorrido inicia ahora por la siguiente arista de peso 3 incidente a $u$, continúa por la arista de peso 4 y da vuelta sobre esta misma hasta que llega de nuevo al vértice $u$. Hasta este punto todas las aristas incidentes a $u$ cuyo peso es menor que $c(e)$ ya han sido recorridas, por lo que se concluye que no existe un camino que conduzca a $v$ con aristas de pesos menores a $c(e)$, por lo tanto la arista $e$ pertenece al AGPM de $G$.

Ahora considere la arista $f$ que muestra la figura 4.3. Suponga que, de manera independiente a $e$, se desea saber si $f$ pertenece al AGPM (mismo que muestra la Figura 4.1). El algoritmo 2 realiza un recorrido por las aristas incidentes a $u$, con lo cual se inicia el recorrido por la arista de peso 7, la siguiente arista que toma es la de peso 4, el recorrido continúa por la arista de peso 3, sigue por la arista de peso 5 y finaliza con la arista de peso 6, en este punto se consigue llegar al vértice $v$, lo cual implica que se encontró un camino de aristas con pesos menores a $c(f)$ que va de $u$ a $v$, por lo tanto $f$ no pertenece al AGPM de $G$.

Como puede ver, en la Figuras 4.2 sucede el caso en el que el recorrido vuelve al vértice $u$ por la arista de inicio. Mientras que en el ejemplo de la Figura 4.3 se consigue un camino que va del vértice $u$ al vértice $v$. Sin embargo puede suceder un tercer caso en el que el recorrido consigue regresar al vértice $u$ pero a través de una

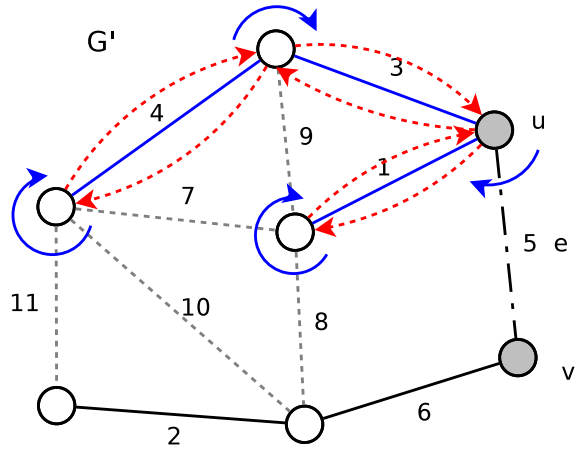


Figura 4.2: Recorrido con la regla de la mano derecha, solo pasando por aristas de peso menor a $c(e)$, iniciando en el vértice $u$.

arista diferente a la de inicio. Para este propósito considere la gráfica $H$ y el AGPM que muestra la Figura 4.4.

Sea $e$ la arista que se desea saber si pertenece al AGPM de la gráfica $H$. En la Figura 4.5 se puede ver que el algoritmo inicia un recorrido por la arista de peso 7, posteriormente se toma la arista de peso 4, seguida de la arista de peso 3 y finalmente se llega a la arista de peso 1, la cual hace que el recorrido regrese al vértice $u$ pero por una arista diferente a la de inicio. En este caso el recorrido da vuelta por la arista de peso 1 y eventualmente vuelve al vértice $u$ mediante la arista de inicio, en cuyo caso el algoritmo finaliza el recorrido correspondiente. Note que el recorrido que inicia por la arista de peso 1 pasa por el mismo camino asociado a la arista de peso 7, con la diferencia que son recorridos en sentidos opuestos.

## 4.2. Complejidad

El Algoritmo 2, que localmente determina si una arista $(u, v)$ pertenece al AGPM, emplea al Algoritmo 3 para recorrer la gráfica inducida $G'$ siguiendo la regla de la mano derecha, dicho recorrido toma tiempo lineal ya que pasa a lo mas dos veces por cada lado de las aristas de la gráfica. Dado que $G$ es una gráfica plana y conexa entonces $|E| \leq 3n - 6$. Por lo tanto el recorrido es de $O(12n - 24)$.

Una vez hecho el recorrido, el Algoritmo 2 toma tiempo constante para decidir si la arista $(u, v)$ pertenece o no al AGPM, ya que solo requiere verificar el resultado que devuelve la rutina de recorrido. Por lo tanto toma tiempo constante $O(1)$.

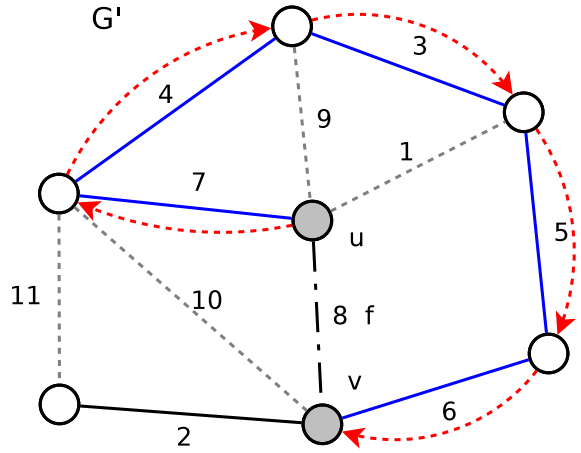


Figura 4.3: Recorrido con la regla de la mano derecha que consigue llegar al vértice $v$.

En conclusión el Algoritmo 2 asintóticamente es de orden $O(n)$.

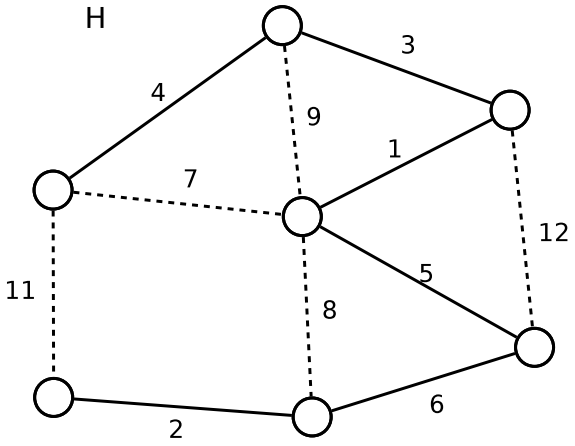


Figura 4.4: Gráfica $H$ geométrica plana y conexa.

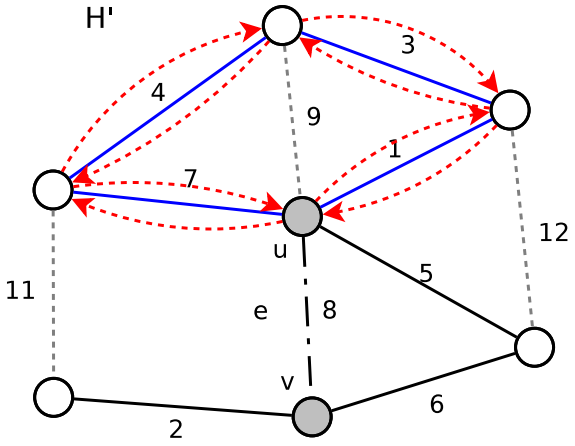


Figura 4.5: Recorrido con la regla de la mano derecha, que llega al vértice $u$ por una arista diferente a la de inicio.

# Capítulo 5

# Conclusiones

## 5.1. Ventajas

Los Algoritmos 2 y 3 solo almacenan un número constante de variables, por lo que el tamaño de la memoria que emplean es de orden constante $O(1)$. Lo anterior permite que los cambios dinámicos tanto en la topoligía como en la disponibilidad de enlaces de la red no afecte el rendimiento del algoritmo, ya que no es necesario actualizar datos por cada cambio que se efectúe en la red.

Por otro lado, evitar centralizar el computo en uno o pocos nodos agrega mayor cantidad de comunicaciones sobre la red, además hace propenso a que los nodos encargados del computo se mantengan siempre activos y obligando a que deban ser robustos a fallos y desconexiones. Considerando lo anterior en un ambiente de redes ad-hoc o redes de sensores, delegar computo a uno o pocos nodos haría que la vida de la batería de estos nodos se vea afectada, además agregar robustez a estos nodos incrementaría la complejidad de estos sistemas.

Los algoritmos que se proponen permiten explotar el paralelismo y la distribución ya que un nodo no necesita esperar a que los demás nodos de la red calculen algín resultado para obtener el propio.

El Algoritmo 3 puede ejecutarse de manera independiente para cada una de las aristas incidentes a un vértice, ya que el algoritmo no presenta dependencia en los resultados de otros recorrido, con lo cual perfectamente puede ejecutarse de forma paralela ganando así eficiencia en el desempeño del algoritmo. Además el algoritmo 2 puede ejecutarse de manera distribuida en cada uno de los nodos de la red, debido a que el cálculo de cada arista no tiene dependencia de otros cálculos de aristas o

nodos en la red. Por lo tanto si el algoritmo 2 se ejecuta de manera distribuida, globalmente se construye el árbol generador de peso mínimo.

## 5.2. Desventajas

Si se desea obtener el árbol generador de peso mínimo de una gráfica arbitraria empleando los algoritmos propuestos, primero debe someterse la grfica a un algoritmo que obtenga una subgráfica plana, lo cual agrega computo a los nodos de la red. Ademas los algoritmos que aplanan una gráfica no consideran el peso de las arista, lo cual puede afectar en la construcción del AGPM.

Considerando que de manera distribuida cada uno de los nodos debe ejecutar los algoritmos propuestos con objetivo de obtener el AGPM de una gráfica plana, puede ver que cada nodo toma tiempo $O(n)$ para calcular sus aristas incidentes, pero si se suma el computo que lleva a cabo en su totalidad los nodos de la red se obtiene que el algoritmo visto de forma global toma tiempo $O(n^2)$ para obtener el AGPM de una gráfica, lo cual es de orden mayor que los algoritmos que se conocen en la literatura.

## 5.3. Conclusiones

El problema del árbol generador de peso mínimo se agrega como uno de los problemas que pueden ser calculados de manera local, es decir, no es necesario contar con el conocimiento global de la red para construir el AGPM de una gráfica plana dada.

# Bibliografía


[1] Baruch Awerbuch. Optimal distributed algorithm for minimum weight spanning tree. *ACM*, 1987.

[2] J. A. Bondy and U.S.R. Murty. *Graph Theory*. Springer, 2008.

[3] Otakar Borůvka. O jistém problému minimálním (about a certain minimal problem). *Práce mor. přírodověd. spol. v Brně III (in Czech, German summary)*, 3:37–58, 1926.

[4] Otakar Borůvka. Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí (contribution to the solution of a problem of economical construction of electrical networks). *Elektronický Obzor (in Czech)*, pages 153–154, 1926.

[5] P./ Bose, P. Morin, I Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc networks. *Proc, of 3rd ACM Int. Workshop on Descrete Algorithms and Methods for Mobile Computing and Comunications DIAL M99, Seattle*, pages 48–55, 1999.

[6] S. Funke, A. Kasselman, and M. Segal U. Meyer. A simple improved distributed algorithm for minimum cds in unit disk graphs. *1st IEEE International Conference on Wireless and Mobile Computing, Networking and Comunications WiMob, Montreal*, 2005.

[7] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum weight spanning trees. *ACM Trans. On Program. Lang. And Systems*, 5:66 – 77, 1983.

[8] J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. *ACM Symposium on Mobile Ad Hoc Networks and Computing MobiHoc, Long Beach, California*, pages 45–55, 2001.

[9] R. L. Graham and Pavol Hell. On the history of the minimum spanning tree problem. page 15, Enero 1985.

[10] Jon Kleinberg and Eva Tardos. *Algorithm Design.*

[11] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. *Proc. 11th Canadian Conference on Computational Geometry, Vancouver*, pages 15–18, 1999.

[12] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[13] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar borůvka on minimum spanning tree problem: translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233 (1-3):3–36, 2001.

[14] Robert C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[15] Sergio Rajsbaum and Jorge Urrutia. Some problems in distributed computational geometry. 1(nah):19, 2012.

[16] J. Suomela. Survey of local algorithms. *ACM Computing Surveys, volume 45, issue 2, article 24.*, 2013.

[17] J. Urrutia. Local solutions for global problems in wireless networks. page 19, 2006.

[18] D. Wagner and R. Wattenhofer (Eds). *Algorithms for Sensor and Ad Hoc Networks.* Springer-Verlag, Berlin Heidelberg, 2007.