UNIVERSIDAD AUTÓNOMA METROPOLITANA

Casa abierta al tiempo

UNIDAD IZTAPALAPA
DIVISIÓN EN CIENCIAS BÁSICAS E INGENIERÍA

ÁRBOLES GENERADORES CON PESO MÍNIMO Y GRADOS ACOTADOS

Tesis que presenta:
Maria Elena Martinez Cuero
para obtener el grado de
Maestra en Ciencias
(Matemáticas Aplicada e Industriales)

Director de Tesis:
Dr. Eduardo Rivera Campo.

Jurado

Presidente: Dra. Virginia Urrutia Galicia.
Secretario: Dr. Eduardo Rivera Campo.
Vocal: Dra. Mucuy-Kak del Carmen Guevara Aguirre.

27 de Febrero de 2015

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Casa abierta al tiempo

UNIDAD IZTAPALAPA
DIVISIÓN EN CIENCIAS BÁSICAS E INGENIERÍA

ÁRBOLES GENERADORES CON PESO MÍNIMO Y GRADOS ACOTADOS

Tesis que presenta:
Maria Elena Martinez Cuero
para obtener el grado de
Maestra en Ciencias
(Matemáticas Aplicada e Industriales)

Director de Tesis:
Dr. Eduardo Rivera Campo.

Jurado

Presidente: Dra. Virginia Urrutia Galicia.
Secretario: Dr. Eduardo Rivera Campo.
Vocal: Dra. Mucuy-Kak del Carmen Guevara Aguirre.

27 de Febrero de 2015

Agradecimientos

Con sincera gratitud deseo dedicar este trabajo a todos aquellos que hicieron posible la culminación de esta investigación. A la Universidad Autónoma Metropolitana por brindarme la oportunidad de estudiar un posgrado y a CONACYT por el soporte dado.

Agradezco a mi madre, quien siempre ha sido mi aliada y amiga; a mi hermano Jorge y a mi padre por su comprensión y apoyo. A Julieta, por su compañía y su siempre motivación. A la lealtad de mi Terry quien se ha desvelado conmigo y soportado la luz encendida por las noches.

A mis profesores de la UAM-I quienes me han compartido su conocimiento durante las clases y fuera de ellas. A Raquel, Héctor Manuel, Leticia, Víctor Manuel, Juan Luis, Victoria, Adalberto, Marco Antonio, Marlene, Liz, Alfonso, Paty, Misael, Karina y Julio; por ser más que mis compañeros. Gracias Jorge Matadamas, por tu valiosa ayuda y paciencia.

Agradezco a la Dra. Mucuy-Kak del Carmen Guevara Aguirre y a la Dra. Virginia Urrutia Galicia sus valiosos consejos y orientación académica. Su participación como sinodales se refleja indudablemente en el mejoramiento de la tesis.

Deseo agradecer muy especialmente la dirección de tesis al Dr. Eduardo Rivera Campo, a quien admiro y respeto profundamente, por su enseñanza, orientación y paciencia. Sin él no habria sido posible sacar adelante este trabajo. Gracias.

¡Gracias Dios por mostrarme en estos momentos tu infinita misericordia al permitirme continuar celebrando la vida junto a mis seres queridos!

Índice general

Introducción

Sean G una gráfica completa con $n \geq 2$ vértices, k un entero con $1 \leq k \leq n-1$ y $\varphi : E(G) \to \mathbb{R}^+$ una función de peso en las aristas de G.

Si en un árbol generador T de G la suma de los pesos de las aristas es lo mínimo posible, entonces T resulta ser un árbol generador de G de peso mínimo. A lo largo de la literatura existen diversos algoritmos que nos permiten encontrar árboles generadores de peso mínimo de una gráfica conexa. Por ejemplo el algoritmo de Prim y el algoritmo de Kruskal.

Fekete et al [5] estudiaron el problema de encontrar un árbol generador S de una gráfica G, con el menor peso posible y tal que los grados de todos los vértices de S sean menores o iguales a cierto entero.

Un árbol generador T de G es un *k-árbol* generador de G si $d_T(u) = 1$ o $d_T(u) \geq k$, $\forall$ $u \in V(T) = V(G)$. En esta tesis estudiamos el problema de encontrar k-árboles generadores de G con el menor peso posible.

En el Capítulo 2, desarrollamos un algoritmo exacto, es decir un algoritmo que, para cualquier instancia del problema, produce un k- árbol generador de peso mínimo. Sin embargo, por la naturaleza del problema, dicho algoritmo resulta ser muy lento comparado con el tamaño de la gráfica.

En el Capítulo 3 consideramos el caso en que φ cumple con la desigualdad del triángulo, es decir $\varphi(uw) \leq \varphi(uv) + \varphi(vw)$ $\forall$ $u, v, w \in V(G)$. En este caso describimos una heurística o algoritmo de aproximación que transforma, de manera eficiente, un árbol generador T de G con peso mínimo en un k-árbol generador de G con peso relativamente bajo.

Para situar el problema y las técnicas usadas, incluimos en el Capítulo 1 algunas definiciones y resultados preliminares.

Capítulo 1

Preliminares

En este capítulo introducimos algunos conceptos básicos de Teoría de Gráficas, principalmente aquellos relacionados con árboles generadores. Para mas detalles nos referimos a los libros "Graph Theory with Applications"[1] e "Introduction to Graph Theory"[3].

1.1. Definiciones

Definición 1 *Una gráfica o gráfica simple G consta de una pareja de conjuntos $(V(G), E(G))$ en la que $V(G)$ es un conjunto finito no vacío de elementos denominados vértices y $E(G)$ es un conjunto finito de pares no ordenados de elementos de $V(G)$ llamados aristas.*

Una gráfica G se puede representar por medio de un dibujo de la siguiente manera: los vértices de G se representan por medio de puntos en el plano y las aristas de G por medio de arcos de curvas cuyos extremos son los vértices correspondientes.

En la siguiente figura representamos la gráfica G dada por $V(G) = \{a, b, c, d, e, f, g, h, i\}$ y $E(G) = \{ab, bc, cd, de, ef, fg, gh, ha, bh, hi, ig, ic, cf, df\}$.

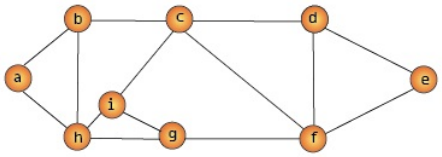


Figura 1: Gráfica G.

Definición 2 *El grado $d_G(u)$ de un vértice u de una gráfica simple G es el número de aristas de G que inciden en u.*

Siempre que no haya confusión escribiremos $d(u)$ en lugar de $d_G(u)$.

En la gráfica G de la Figura 1 tenemos $d(a) = d(e) = 2$, $d(b) = d(d) = d(g) = d(i) = 3$ y $d(c) = d(f) = d(h) = 4$.

Definición 3 *Sea n un entero positivo. Una gráfica G con n vértices es completa si todo par de vértices de G son adyacentes, es decir, que uv es una arista de G para todo par de vértices u y v de G.*

La gráfica completa con n vértices tiene $\frac{1}{2}n(n-1)$ aristas y se denota por K_n. La gráfica de la Figura 2 es una gráfica completa con n =6 vértices y $m = 15$ aristas.

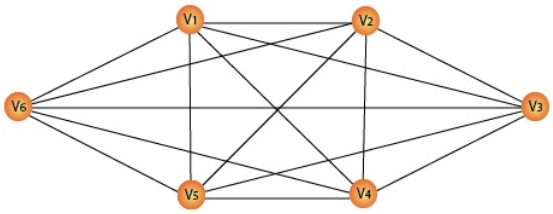


Figura 2: Gráfica completa $G = K_6$

Definición 4 *Una trayectoria en una gráfica G es una sucesión $u = u_0, u_1, u_2, ..., u_k = v$ de vértices distintos de G junto con las aristas u_iu_{i+1} para $i = 0, 1, \ldots, k-1$.*

Los vértices *a,b,c,d* y *e* junto con las aristas *ab, bc, cd* y *de* forman una trayectoria en la gráfica G de la Figura 1.

Definición 5 *Una gráfica G es conexa si para cada par de vértices u, v de G, la gráfica G tiene una trayectoria con extremos u y v. Las componentes conexas de una gráfica G son las subgráficas conexas de G máximas con esa propiedad.*

Definición 6 *Un ciclo de una gráfica G es una trayectoria $u = u_0, u_1, u_2, ..., u_k = v$ junto con la arista vu.*

La trayectoria a, b, c, i, g, h, junto con la arista ha es un ciclo de la gráfica de la Figura 1.

Definición 7 *Un árbol T es una gráfica simple conexa que no contiene ciclos.*

La gráfica T de la Figura 3 es un árbol. En todo árbol cualquier nueva arista crea exactamente un ciclo y cualquier par de vértices está conectado por una trayectoria única.

Si elegimos un vértice v de $V(T)$ y lo nombramos como vértice raíz r de T, entonces T es un árbol con raíz r y cada arista del árbol con raíz r va a tener una orientación de la raíz hacia fuera, por lo que si v_iv_j esta orientada de v_i a v_j, entonces el vértice v_i es considerado el padre del vértice v_j.

Cada $v \epsilon V(T)$ está en un nivel de T, el vértice raíz r de T esta en el nivel 0, los vértices hijos de r están en el nivel 1, los vértices adyacentes a los hijos de r están en el nivel 2 y así sucesivamente.

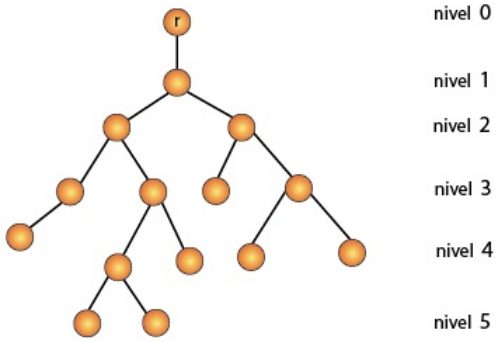


Figura 3: Arbol T.

Definición 8 *Un árbol generador de una gráfica G es una subgráfica T de G tal que T es un árbol y $V(T) = V(G)$.*

En la Figura 4 mostramos un árbol generador de la gráfica G de la Figura 1.

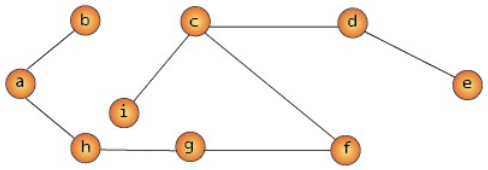


Figura 4: Árbol generador de la gráfica de la Figura 1.

Definición 9 *Un bosque B es una gráfica en la que cada componente conexa es un árbol. Un bosque generador de una gráfica G es una subgráfica B de G tal que B es bosque y $V(B) = VG)$.*

En la Figura 5 mostramos un bosque B con tres componentes conexas y que es bosque generador de la gráfica de la Figura 1.

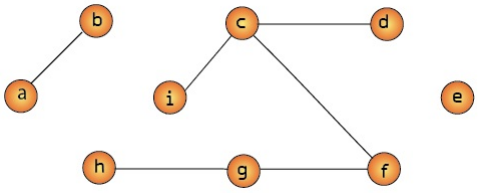


Figura 5: Bosque generador de la gráfica de la Figura 1

Definición 10 *Una gráfica ponderada (G, φ) es una gráfica G junto con una función de peso φ definida en las aristas de G.*

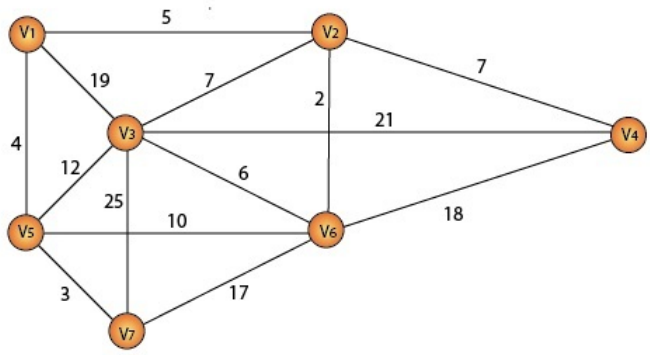


Figura 6: Gráfica ponderada.

Definición 11 *Dada una gráfica ponderada (G, φ), el peso $\varphi(F)$ de una subgráfica F de G es la suma de los pesos de las aristas de F. Un árbol generador T de G es árbol generador de peso mínimo si $\varphi(T) \leq \varphi(S)$ para todo árbol generador S de G.*

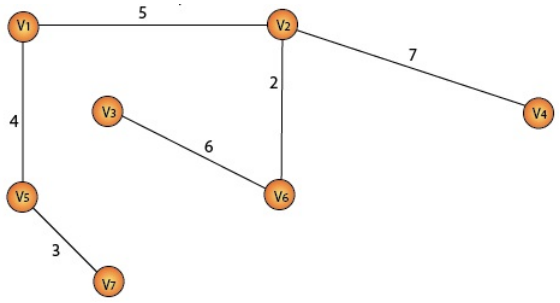


Figura 7: Árbol generador con peso mínimo de la gráfica de la Figura 6.

Definición 12 *Sea $k \geq 2$ un entero. Un árbol generador T de una gráfica G es un k-árbol de G si el grado de cada vértice u de T es 1 o es mayor o igual a k.*

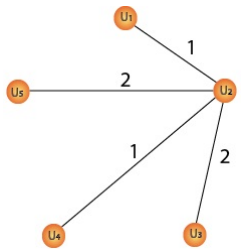


Figura 8: Un 3-árbol.

1.2. Árboles Generadores de Peso Mínimo

Uno de los principales destinos turísticos de México es Yucatán por sus zonas arqueológicas. Supongamos que el Gobierno del Estado contrata a una de las mejores compañías constructoras del país para hacer una red de carreteras que conecte las distintas zonas. Dado que el costo se eleva demasiado si se construyen todas las posibles vías, entonces la compañía debe construir solo las carreteras necesarias que conecten las zonas turísticas al menor costo posible.

Al problema de la construcción de la red de carreteras se le puede asociar una gráfica G completa y una función de peso φ definida en las aristas de G , donde los vértices sean las zonas arqueológicas, las aristas las posibles carreteras que conecten la zona arqueológica u con la zona v y el peso $\varphi(uv)$ en cada arista uv represente el costo de construción de la carretera de u a v. La solución al problema anterior resulta ser equivalente a encontrar un árbol generador T de G de peso mínimo.

Dentro de los algoritmos más usados para encontrar un árbol generador de peso mínimo de una gráfica ponderada G, están el Algoritmo de Kruskal y el Algoritmo de Prim descritos a continuación. Para mayor detalle nos referimos al libro "Introduction to Algorithms", Sección 24.2 [4].

1.2.1. Algoritmo de Kruskal

El algoritmo de Kruskal describe una sucesión de bosques generadores $B_0, B_1, \ldots B_{n-1}$ de G , en donde B_0 está formado por una componente conexa C_j por cada vértice v_j de G y para $i = 0, 1, \ldots, n-2$, el bosque B_{i+1} se obtiene del bosque B_i añadiendo una arista $e_{i+1} = uv$ con el menor peso posible tal que sus extremos u y v están en componentes conexas distintas C_j y C_k de B_i.

El bosque final B_{n-1} es un árbol generador de G con peso mínimo.

1.2.2. Algoritmo de Prim

El algoritmo de Prim parte de un vértice v_1 de G y construye un árbol T de G que va creciendo, añadiendo en cada iteración una arista wz de G con el menor peso posible tal que $w \in V(T)$ y $z \in V(G) \setminus V(T)$.

Para j=0, $V(T) = \{v_1\}$ y $E(T) = \emptyset$.

Si para $j = k$, el subárbol T de G esta formado por $V(T) = \{v_1, v_2, ..., v_k, v_{k+1}\}$ y $E(T) = \{e_1, e_2, ..., e_k\}$, entonces en la iteración $j = k + 1$ se elige la arista $e_{k+1} = wz$ de $E(G) \setminus E(T)$ de menor peso posible con $w \in V(T)$ y $z \in V(G) \setminus V(T)$. El subárbol T se actualiza añadiendo la arista e_{k+1}=wz y el vértice $v_{k+2} = z$.

El algoritmo para, cuando T es un árbol generador de G.

El árbol generador con peso mínimo de una gráfica no es necesariamente único pero el resultado del peso mínimo si lo es, no importando el algoritmo que se utilice.

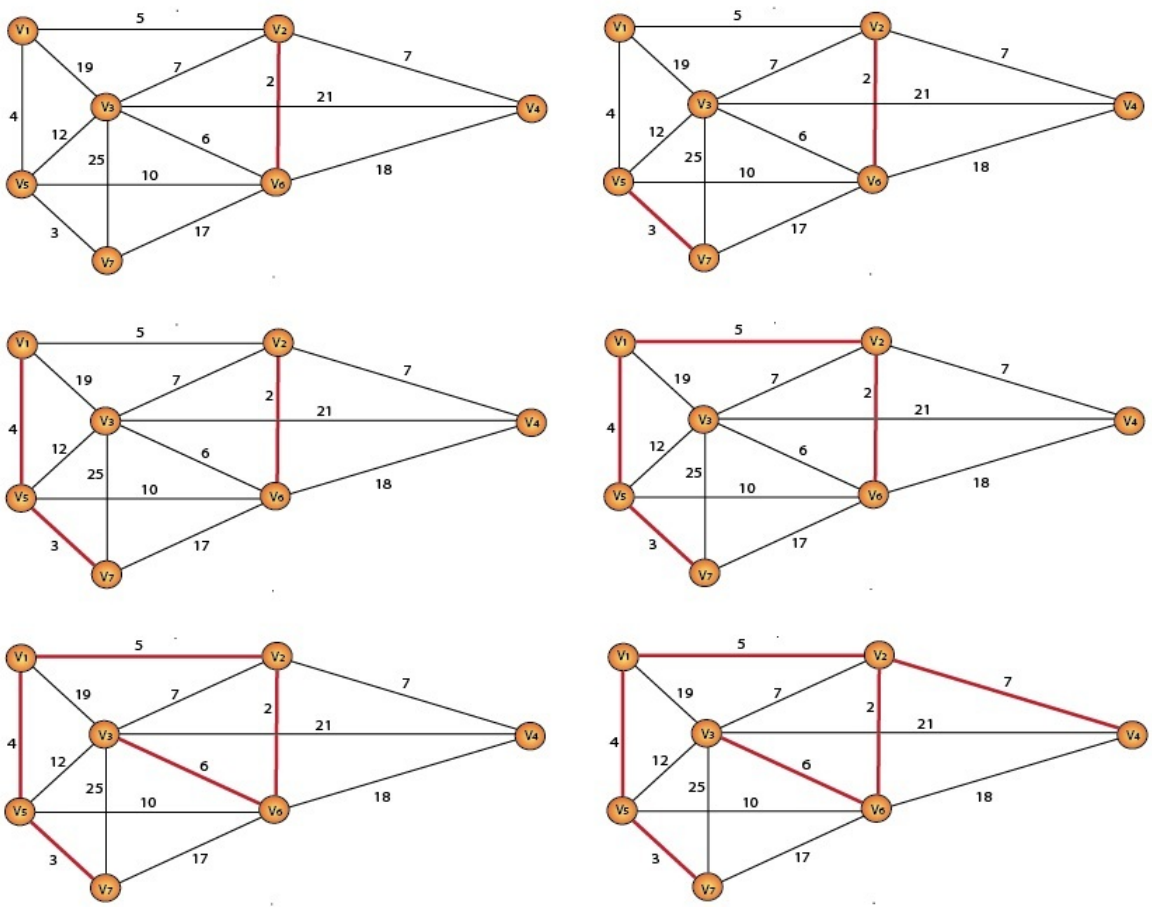


Figura 9: Aplicación del algoritmo de Kruskal.

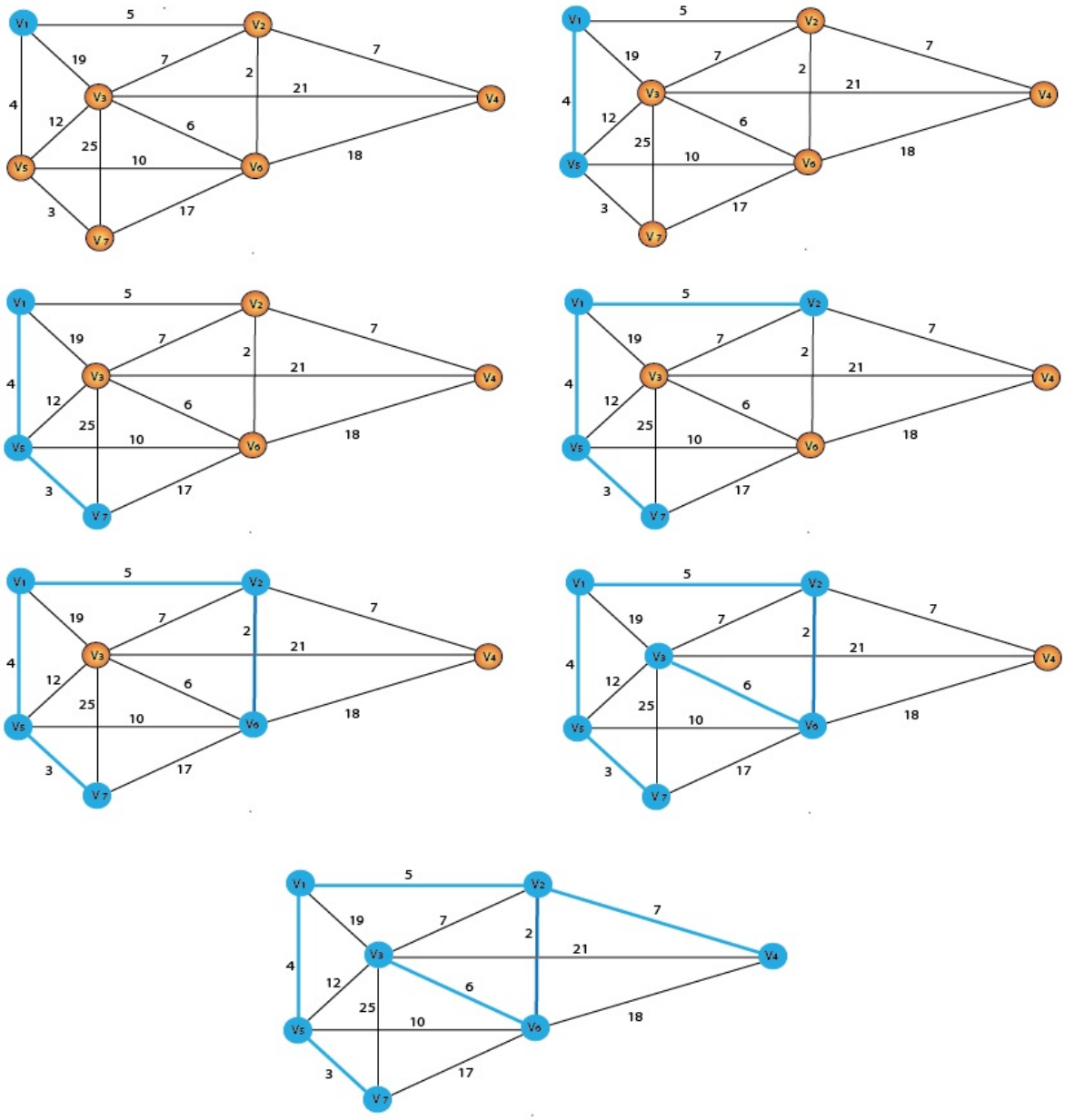


Figura 10: Aplicación del algoritmo de Prim.

1.3. El Problema del Agente Viajero

Un viajero desea visitar n ciudades partiendo de una de ellas, visitando las otras $n-1$ ciudades restantes una sola vez y volviendo a la ciudad inicial, pero necesita hacerlo de tal forma que el recorrido total sea lo más corto posible.

A cada instancia de este problema se le asocia una gráfica completa K_n con un vértice v_i por cada ciudad y una función de peso φ en las aristas de G tal que $\varphi(v_iv_j)$ representa la distancia de la ciudad i a la ciudad j.

La solución al problema del agente viajero equivale a encontrar un ciclo hamiltoniano o ciclo generador de K_n de menor peso posible.

El problema del árbol generador con peso mínimo ayuda a obtener una cota inferior al peso de la solución del problema del agente viajero. Observemos que si se toma cualquier ciclo hamiltoniano C de la gráfica completa ponderada $G = K_n$ y se elimina un vértice v de C, se obtiene una trayectoria generadora (árbol generador) de la gráfica $G - v$.

De esta manera si se considera el peso del árbol generador de peso mínimo de la gráfica $G - v$ y se le suma los pesos de las dos aristas de menor peso incidentes en v se encuentra una cota inferior a la solución del problema del agente viajero (ver [**?**]).

En una variante del problema del agente viajero, el viajero debe visitar las n ciudades con el recorrido mas corto posible sin necesidad de regresar a la ciudad de partida.

La solución del problema del agente viajero modificado es equivalente a encontrar un árbol generador de peso mínimo de la gráfica G con la restricción adicional de que el grado de cada uno de sus vértices no sea mayor que dos (trayectoria generadora). Claramente el peso de un árbol de peso mínimo de G es cota inferior para la solución del problema del agente viajero modificado correspondiente.

1.3.1. Algoritmo de Christofides

En el libro "Graph Theory: An Algorithmic Approach", Sección 6.2 [2], Nicos Christofides describe un algoritmo de busqueda por árbol de decisión para resolver de manera exacta el Problema del Agente Viajero Modificado.

Sea G la gráfica de la Figura 11 con función de peso φ dada por la Tabla 1.

En la tabla 1 se muestra el peso $\varphi(v_iv_j)$ de cada arista v_iv_j de la gráfica G.

Usando este ejemplo describimos a continuación el algoritmo de Nicos Christofides para encontrar una trayectoria generadora de G con peso mínimo.

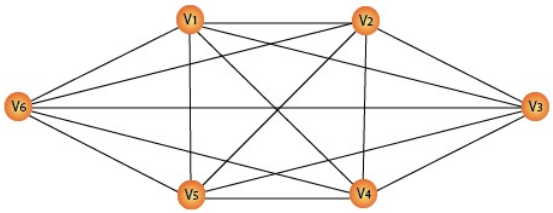


Figura 11: Gráfica completa $G = K_6$

	1	2	3	4	5	6
1	0	1	2	4	3	6
2	1	0	1	4	8	1
3	2	1	0	4	1	5
4	4	4	4	0	1	6
5	3	8	1	1	0	5
6	6	1	5	6	5	0

Figura 12: Tabla 1.

Consideremos el árbol generador T de G con peso mínimo (Figura 13).

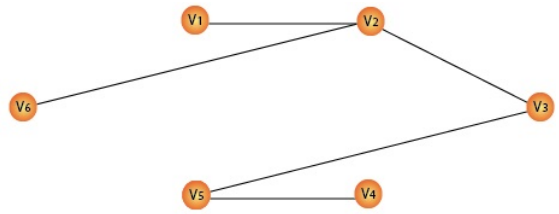


Figura 13: Árbol T, $\varphi(T) = 5$

Ya que $d_T(v_2)$>2, para cada trayectoria generadora T^* de G, al menos una de las aristas de T, incidentes en v_2 no es arista de T^*. Prohibimos de una en una las aristas de T incidentes en v_2.

Sin pérdida de generalidad elegimos a la arista v_1v_2, colocando ∞ en las entradas (1,2) y (2,1,) de la Tabla 1 para obtener la Tabla 2 (Figura 14).

A continuación encontramos un árbol generador A de G de peso mínimo con los pesos dados por la Tabla 2 (Figura 15).

Hecho todo lo anterior toca el turno de prohibir la arista v_2v_3 de T reemplazando el valor de las entradas (2,3) y (3,2) por ∞ en la Tabla 1, obteniendo la Tabla 3 (Figura 16).

	1	2	3	4	5	6
1	0	∞	2	4	3	6
2	∞	0	1	4	8	1
3	2	1	0	4	1	5
4	4	4	4	0	1	6
5	3	8	1	1	0	5
6	6	1	5	6	5	0

Figura 14: Tabla 2.

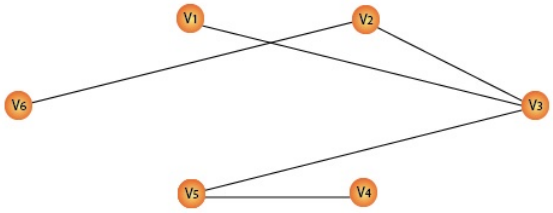


Figura 15: Árbol A, $\varphi(A) = 6$

	1	2	3	4	5	6
1	0	1	2	4	3	6
2	1	0	∞	4	8	1
3	2	∞	0	4	1	5
4	4	4	4	0	1	6
5	3	8	1	1	0	5
6	6	1	5	6	5	0

Figura 16: Tabla 3.

Usamos de nuevo un algoritmo para encontrar un árbol generador B de G de peso mínimo con los pesos dados por la Tabla 3 (Figura 17).

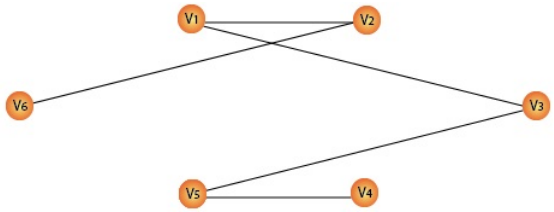


Figura 17: Árbol B, $\varphi(B) = 6$

Finalmente prohibimos la arista v_2v_6 de T de la misma manera que lo realizamos con las aristas v_1v_2 y v_2v_3 (Figura 18.)

Obtenemos un árbol generador C de G de peso mínimo de acuerdo a la Tabla 4, (Figura 19).

El algoritmo de Nicos Christofides genera una estructura Δ con raíz T, donde los árboles generadores A, B y C que obtuvimos con la ayuda de las tablas 1, 2 y 3 respectivamente son hijos de T (Figura 20).

De los vértices terminales de la estructura Δ, elegimos el árbol Z de menor peso. Si Z es una trayectoria, entonces Z es solución al problema del agente viajero modificado de la gráfica completa G. De lo contrario, el algoritmo itera prohibiendo aristas incidentes en algún vértice u de Z con $d_Z(u) > 2$.

En nuestro ejemplo B es un vértice terminal de Δ con el menor peso posible y es una trayectoria. Por lo tanto B es la trayectoria generadora de peso mínimo, que da solución al problema del agente viajero modificado correspondiente a la gráfica G.

	1	2	3	4	5	6
1	0	1	2	4	3	6
2	1	0	1	4	8	∞
3	2	1	0	4	1	5
4	4	4	4	0	1	6
5	3	8	1	1	0	5
6	6	∞	5	6	5	0

Figura 18: Tabla 4.

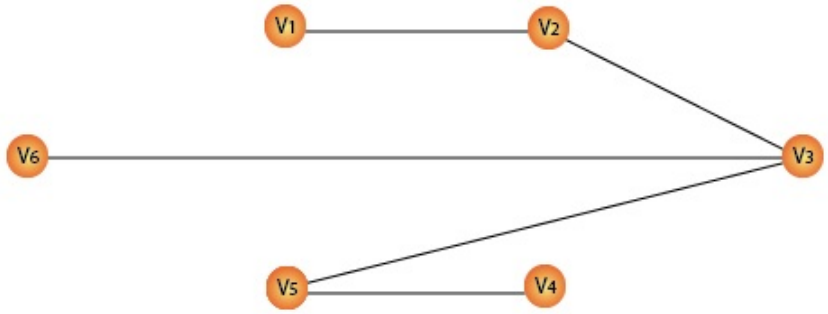


Figura 19: Árbol C, $\varphi(C) = 9$.

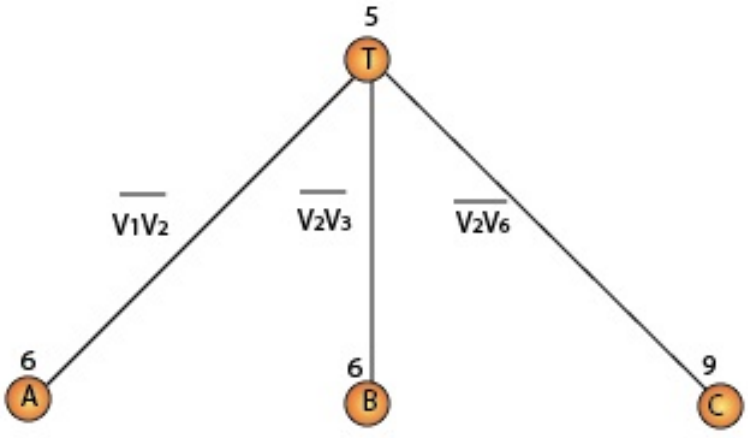


Figura 20: Estructura Δ

Capítulo 2

Algoritmo Exacto

Sea G una gráfica completa con $n \geq 2$ vértices y φ: $E(G) \rightarrow \mathbb{R}^+$ una función de peso en las aristas de G.

Dado un entero k con $1 < k < n$, un árbol generador Z de G es un *k-árbol generador* de G si $d_Z(u) \geq k$ ó $d_Z(u) = 1$ para todo vértice u de Z.

En este capítulo describiremos un algoritmo exacto que nos permita encontrar un k-árbol generador de G con el menor peso posible.

El algoritmo inicia encontrando un árbol generador R de G de peso mínimo. Si $d_R(u) \geq k$ ó $d_R(u) = 1$ para todo vértice u de R, entonces R es el k-árbol generador de G con peso mínimo que se busca. De lo contrario el algoritmo itera construyendo una estructura Δ, cuya gráfica subyacente es un árbol con raíz R y en donde cada uno de sus vértices es un árbol generador de G.

Para cada vértice Z de Δ vamos a considerar los siguientes dos conjuntos: $O(Z)$ formado por las aristas uv de G que están obligadas a formar parte del árbol Z y de todos los descendientes de Z y $P(Z)$ formado por las aristas uv de G que están prohibidas en Z y en todos los descendientes de Z.

Un vértice Z en Δ es vértice *fértil* si ocurre cualquiera de las dos siguientes situaciones:

1.- El árbol Z es un k-aŕbol generador de G.

2.- El árbol Z tiene un vértice u con $1 < d_Z(u) < k$ tal que:

i) Existen una arista $uv \in E(G) \setminus E(Z)$ y un árbol generador X de G que contiene a todas las aristas en $O(Z) \cup \{uv\}$ y que no contiene a ninguna arista en $P(Z)$; ó

ii) Existen una arista $uw \in E(Z)$ y un árbol generador Y de G que contiene a todas las aristas en $O(Z)$ y que no contiene a ninguna arista en $P(Z) \cup \{uw\}$.

Después de cada iteración el algoritmo actualiza la estructura Δ añadiendo vértices hijos adyacentes a cierto vértice terminal fértil.

La condición de paro en el algoritmo ocurre cuando un vértice terminal fértil de Δ con menor peso es un k-árbol generador de G.

A continuación describiremos el algoritmo exacto usando como ejemplo la gráfica G de la Figura 21 con $k = 3$.

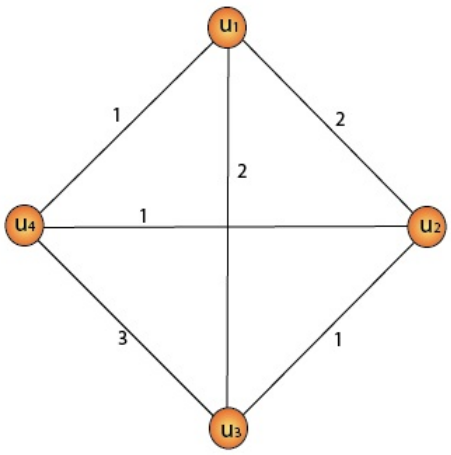


Figura 21: Los pesos de G están indicados junto a las aristas.

Iniciamos el algoritmo con la estructura Δ formada por un árbol generador R de G con peso mínimo como vértice raíz y en donde $O(R) = P(R) = \emptyset$.

Si $d_R(u) \geq k$ para todo vértice no terminal $u \, \epsilon \, V(R)$, entonces R es el k-árbol generador de peso mínimo de G que estamos buscando. De lo contrario R es un vértice fértil en Δ y existe un vértice u de R con $1< d_R(u) < k$.

Recordemos que todo k-árbol generador X de G debe satisfacer que $d_X(u) = 1$ o $d_X(u) \geq k$. Para lograr lo anterior tenemos dos opciones:

Opción 1.- Aumentar el grado de u considerando árboles generadores de G que contengan alguna arista $e \, \epsilon \, E(G) \backslash E(R)$ incidente en u.

Opción 2.- Disminuir el grado de u considerando árboles generadores de G que no contengan alguna arista $f \, \epsilon \, E(R)$ incidente en u.

En nuestro ejemplo R es el árbol de la Figura 22 en donde podemos observar que los vértices no terminales de R no cumplen con la restricción en los grados, por lo cual R no es un 3-árbol. Sin embargo R es fértil, por lo que dentro de los vértices no terminales que no cumplen con la restricción en los grados, elegimos sin pérdida de generalidad a $u = u_2$.

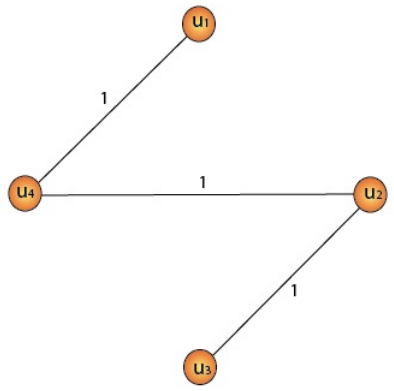


Figura 22: Árbol generador R de G con peso mínimo.

Para la primera opción, u_2u_1 es la única arista en $E(G) \setminus E(R)$ incidente en u_2. Encontramos un árbol generador A de G con el de menor peso posible tal que $u_2u_1 \epsilon E(A)$, (Figura 23 izquierda).

Para la opción 2, u_2u_3 y u_2u_4 son las únicas aristas en $E(R)$ incidentes en u_2. Encontramos un árbol generador B de G con el menor peso posible tal que $u_2u_3 \notin E(B)$ (Figura 23 centro) y un árbol generador C de G con la condición que $u_2u_4 \notin E(C)$ (Figura 23 derecha).

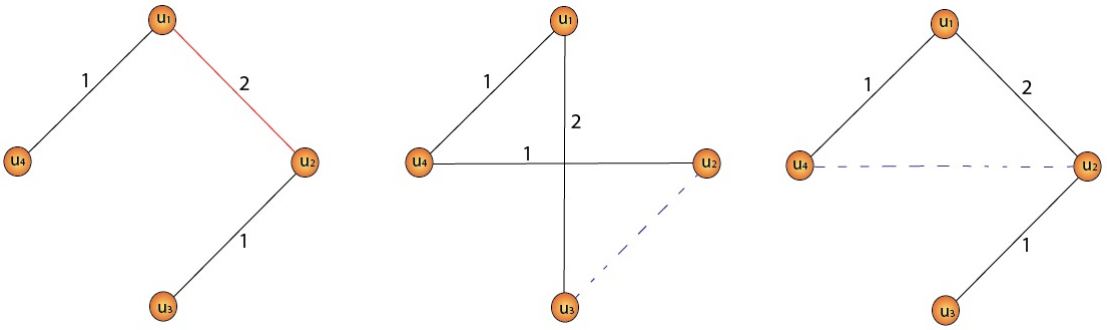


Figura 23: A, $\varphi(A)$=4 B, φ (B)=4, C, $\varphi(C)$=4

A continuación actualizamos a Δ haciendo A, B y C hijos de R, (Figura 24). Recordemos que cada vértice en Δ tiene asociado dos conjuntos en este caso $O(A) = \{u_2u_1\}$ y $P(A) = \emptyset$; $O(B) = \emptyset$ y $P(B) = \{u_2u_3\}$ y $O(C) = \emptyset$ y $P(C) = \{u_2u_4\}$.

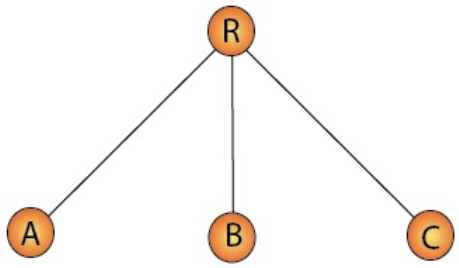


Figura 24: Δ actualizada.

Con la estructura Δ actualizada elegimos un vértice terminal fértil Z de Δ de menor peso. Si Z es un k-árbol, entonces Z es un k-árbol generador de G de menor peso.

Si Z no es un k-árbol, entonces existe un vértice $u \epsilon V(Z)$ tal que $1 < d_Z(u) < k$. De nuevo tenemos dos opciones:

Opción 1.- Considerar los árboles generadores de G que contengan todas las aristas en $O(Z)$ y ninguna arista en $P(Z)$ y que contengan alguna arista $e \epsilon E(G) \backslash E(Z)$ incidente en u.

Opción 2.- Considerar los árboles generadores de G que contengan todas las aristas en $O(Z)$ y ninguna arista en $P(Z)$ y que no contengan alguna arista $f \epsilon E(Z)$ incidente en u.

En nuestro ejemplo A, B y C son vértices terminales fértiles de Δ con $\varphi(A) = \varphi(B) = \varphi(C) = 4$. Sin perder generalidad tomamos $Z = C$ y $u = u_1$. La arista u_1u_3 es la única arista en $E(G) \setminus E(C)$ incidente en u_1 y las aristas u_1u_2 y u_1u_4 son las aristas en $E(C)$ incidentes en u_1.

Para la primera opción encontramos un árbol generador D de G con el menor peso posible que contenga a la arista u_1u_3 y no contenga a la arista u_2u_4 (Figura 25 izquierda).

Considerando la opción 2, encontramos un árbol generador F de G con el menor peso posible que no contenga a las aristas u_2u_4 y u_1u_2 (Figura 25 centro) y un árbol generador H de G ambién con el menor peso posible, que no contenga a las aristas u_2u_4 y u_1u_4 (Figura 25 derecha).

Actualizamos la estructura Δ haciendo D, F y H hijos de C con $O(D) = \{u_1u_3\}$, $P(D) = \emptyset$, $O(F) = \emptyset$, $P(F) = \{u_1u_2, u_2u_4\}$, $O(H) = \emptyset$ y $P(H) = \{u_1u_4, u_2u_4\}$, (Figura 26).

Una vez que Δ ha sido actualizada, iteramos considerando un vértice terminal fértil Z de Δ con el menor peso posible. Si Z es un k-árbol, entonces Z es un k-árbol generador de G con peso mínimo.

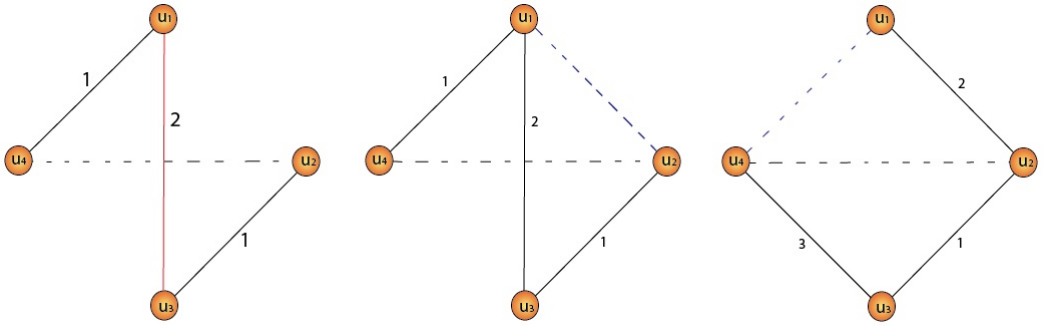


Figura 25: D, $\varphi(D)$=4 F, φ (F)=4, H, $\varphi(H)$=6

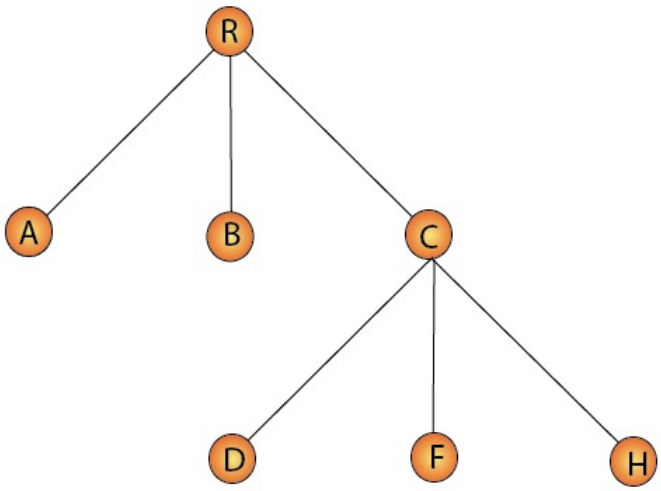


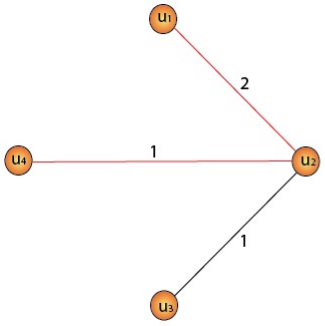
Figura 26: Δ actualizada.

En el ejemplo, A es un vérice terminal fértil de Δ con el menor peso posible y u_2 es tal que $d_A(u_2) = 2$. Procedemos con las opciones 1 y 2 correspondientes:

Para la primera opción, encontramos un árbol generador de menor peso I de G que contenga a todas las aristas en $O(A) \cup \{u_2u_4\} = \{u_2u_1, u_2u_4\}$ y ninguna arista en $P(A) = \emptyset$ (Figura 27 izquierda). De la misma forma para la segunda opción buscamos un árbol generador de menor peso posible J que contenga a todas las aristas en $O(A) = \{u_2u_1\}$ y ninguna arista en $P(A) \cup \{u_2u_3\} = \{u_2u_3\}$ (Figura 27 derecha).

Actualizamos la estructura Δ, colocando a I y J como hijos del vértice A con sus respectivos conjuntos $O(I) = O(A) \cup \{u_2u_4\} = \{u_2u_1, u_2u_4\}$, $P(I) = P(A) = \emptyset$, $O(J) = O(A) = \{u_2u_1\}$ y $P(J) = P(A) \cup \{u_2u_3\} = \{u_2u_3\}$, (Figura 28).

En el ejemplo, el vértice I es un vértice fértil terminal de Δ con el menor peso posible. I es un 3-árbol de G y por lo tanto es un 3-árbol generador de peso mínimo de G.



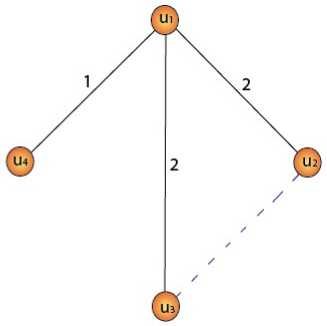


Figura 27: I, $\varphi(I) = 4$ J, $\varphi(J) = 5$

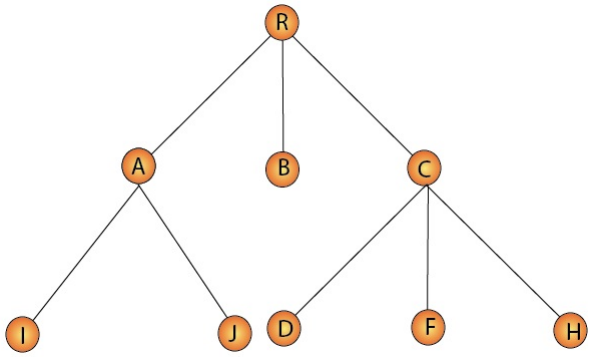


Figura 28: Δ actualizada.

En lo que resta del capítulo demostramos que el algoritmo es correcto, es decir que para toda instancia (G, φ), el algoritmo produce un k-árbol generador de G con peso mínimo. Iniciamos con un lema.

Lema 1. *En todo momento la estructura Δ contiene al menos un vértice terminal fértil.*

Demostración.

Sea N un k-árbol generador cualquiera de G y sea Z un vértice de Δ en el nivel más grande posible tal que:

$$O(Z) \subset E(N) \text{ y } P(Z) \subset E(N^c)$$

El vértice Z está bien definido pues la raíz R de Δ es tal que $O(R) = P(R) = \emptyset$.

Si Z es k-árbol, entonces Z es vértice terminal fértil de Δ. Si Z no es k-árbol, entonces existe $u \in V$ con $1 < d_Z(u) < k$. Notemos que $d_N(u) \geq k$ o $d_N(u) = 1$, por lo tanto existe una arista $e = uv \in E(N) \setminus E(Z)$ o existe una arista $f = uw \in (E(Z) \setminus E(N))$.

En el primer caso, notemos que N es un árbol generador de G tal que $O(Z)\cup\{uv\} \subset E(N)$ y$P(Z) \subset E(N^c)$. Por lo tanto Z es vértice teminal fértil de Δ o Z tiene un hijo X en Δ con $O(X) = O(Z) \cup \{uv\} \subset E(N)$ y $P(X) = P(Z) \subset E(N^c)$, lo cual no es posible de acuerdo a la elección de Z.

En el segundo caso, N es un árbol generador de G tal que $O(Z) \subset E(N)$ y$P(Z)\cup\{uw\} \subset E(N^c)$. Por lo tanto Z es vértice teminal fértil de Δ o Z tiene un hijo Y en Δ con $O(Y) = O(Z)$ y $P(Y) = P(Z) \cup \{uw\} \subset E(N^c)$, lo cual tampoco no es posible de acuerdo a la elección de Z.■

Recordemos que el número de aristas de una gráfica completa es $\binom{n}{2}$.

Sea Z un vértice de Δ en el nivel t. El número de aristas en $O(Z) \cup P(Z)$ es exactamente t, lo cual implica que Δ tiene a lo más $1 + \binom{n}{2}$ niveles. Esto junto al lema anterior implica que el algoritmo termina con un vértice terminal fértil que es k-árbol generador de G. El siguiente teorema muestra que el algoritmo termina con un k-árbol generador de G con peso mínimo.

Teorema 1. *Sea T un vértice terminal fértil en Δ tal que $\varphi(T) \leq \varphi(Z)$ para todo vértice terminal fértil Z en Δ. Si T es un k-árbol, entonces T es k-árbol generador de G con peso mínimo.*

Demostración.

Sea M un k-árbol generador de G con peso mínimo y sea Z un vértice de Δ en el nivel más grande posible tal que:

$$O(Z) \subset E(M),\ P(Z) \subset E(M^c) \text{ y } \varphi(Z) \leq \varphi(M)$$

El vértice Z está bien definido pues la raiz R es tal que $O(R) = P(R) = \emptyset$ y $\varphi(R) \leq \varphi(M)$.

Caso 1.- Z es k-árbol.

En este caso Z es vértice terminal fértil de Δ. Por hipótesis $\varphi(T) \leq \varphi(Z)$ y por la elección de Z, $\varphi(Z) \leq \varphi(M)$. Por lo tanto T es k-árbol generador de G con peso mínimo.

Caso 2.- Z no es k-árbol.

Sea $u \in V$ tal que $1 < d_Z(u) < k$. Notemos que $d_M(u) \geq k$ o $d_M(u) = 1$ pues M si es k-árbol.

Si $d_M(u) \geq k$, entonces existe una arista $uv \in E(M) \setminus E(Z)$. En este caso M es un árbol generador de G tal que $O(Z) \cup \{uv\} \subset E(M)$ y$P(Z) \subset E(M^c)$. Por lo tanto Z es vértice

fértil de Δ.

Si Z es vértice terminal, entonces $\varphi(T) \leq \varphi(Z) \leq \varphi(M)$ y por lo tanto T es k-árbol generador de G con peso mínimo. Si Z no es terminal, entonces Z tiene un hijo X en Δ con $O(X) = O(Z) \cup \{uv\} \subset E(M)$, con $P(X) = P(Z) \subset E(M^c)$ y con $\varphi(X) \leq \varphi(M)$ lo cual no es posible por la elección de Z.

Si $d_M(u) = 1$, entonces existe una arista $uv \in E(M) \setminus E(T)$. En este caso M es un árbol generador de G tal que $O(Z) \subset E(M)$ y$P(Z) \cup \{uw\} \subset E(M^c)$. Por lo tanto Z es vértice fértil de Δ.

Si Z es vértice terminal, entonces $\varphi(T) \leq \varphi(Z) \leq \varphi(M)$ lo cual implica que T es k-árbol generador de G con peso mínimo. Si Z no es terminal, entonces Z tiene un hijo Y en Δ con $O(Y) = O(Z)$, con $P(Y) = P(Z) \cup \{uw\} \subset E(M^c)$ y con $\varphi(Y) \leq \varphi(M)$, lo cual tampoco es posible de acuerdo a la elección de Z. ■

Capítulo 3

Algoritmo de Aproximación

Sea G una gráfica completa con $n \geq 2$ vértices, k un entero con $1 < k < n$ y φ: $E(G) \rightarrow \mathbb{R}^+$ una función de peso en las aristas de G. En este capítulo consideramos el caso en que φ satisface la desigualdad del triángulo, es decir $\varphi(uw) \leq \varphi(uv) + \varphi(vw)\ \forall\ u, v, w \in V(G)$.

Para este caso desarrollamos una heurística o algoritmo de aproximación que busca de manera eficiente un k-árbol generador de G de bajo peso a través de una serie de transformaciones desde un árbol generador T de G de peso mínimo.

3.1. Operación de Adopción

Antes de dar inicio a la descripción del algoritmo de aproximación necesitamos presentar la *Operación de Adopción* dada por Fekete et al [5]. Esta operación formará parte del algoritmo de aproximación con el objetivo de encontrar un k-árbol generador de G con bajo peso.

En la operación de adopción $Adop(T, u, v, x)$, T es un árbol generador de G y u, v y x son vértices de T.

$Adop(T, u, v, x)$ significa que u adopta a x, x es adyacente a v y no se encuentra en la única trayectoria de u a v en T.

El vértice u adopta a x añadiendo a T la arista $ux \in E(G) \setminus E(T)$ y quitando la arista $vx \in E(T)$, evitando que se forme un ciclo. Observemos que el grado de v debe ser al menos dos (Figura 29).

Cuando se hace la operación de adopción $Adop(T, u, v, x)$, el grado del vértice u se incrementa en una unidad y el grado del vértice v por el contrario decrece en uno.

El aumento (disminución) en el peso del árbol resultante T' está dado por:

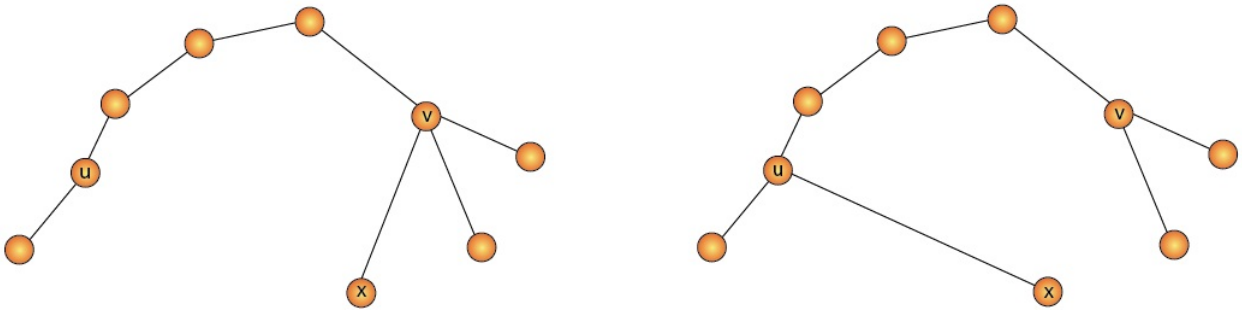


Figura 29: *Adop*(T, u, v, x)

$$\varphi(T') - \varphi(T) = \varphi(ux) - \varphi(vx) \leq (\varphi(uv) + \varphi(vx)) - \varphi(vx) = \varphi(uv)$$

Por lo tanto $\varphi(vu)$ es cota superior para el costo de la adopción $Adop(T, u, v, x)$, para todo vértice x, adyacente a v y que no se encuentra en la única trayectoria de u a v en T.

3.2. Descripción del Algoritmo de Aproximación

Consideremos un árbol generador T de G con peso mínimo y con un vértice r asignado como raíz. Orientamos las aristas de T de tal manera que para cada vértice u de T, la única trayectoria de r a u en T esté orientada de r a u. Con esta orientación de las aristas de T, para cada vértice u distinto de T existe un único vértice $p(u)$, adyacente a u en T, tal que la arista $p(u)u$ está orientada de $p(u)$ a u. El vértice $p(u)$ se denomina el padre de u.

Por medio de ciertas adopciones, el algoritmo $\text{Alg}(T, r, k)$ va transformando (si es necesario) el árbol T hasta obtener un k-árbol generador de G con bajo peso, dando una solución aproximada al problema de encontrar un k-árbol generador de G con peso mínimo.

Dado un vértice z de T el algoritmo $\text{Alg}(T, z, k)$ trabaja en el subárbol T_z de T formado por el vértice z y todos sus descendientes, transformándolo en un k-subárbol de G con los mismos vértices.

Dado un vértice u de T el ex-grado $d^+(u)$ de u es el número de aristas incidentes en u que están orientadas hacia fuera de u. Análogamente el in-grado $d^-(u)$ de u es el número de aristas incidentes en u que están orientadas hacia u. Claramente $d(u) = d^+(u) + d(^-(u)$.

Para cada vértice z de T sea $U_z = \{x \in V(T) \mid z = p(x) \text{ y } d^+(x) \geq 1\}$. Notemos que U_z puede ser el conjunto vacío o distinto de él.

Caso 1. $U_z = \emptyset$.

Si $d^-(z) + d^+(z) < k$, entonces $p(z)$ adopta a todos los hijos de z dejando a z como un vértice de grado 1. Si $d^-(z) + d^+(z) \geq k$, entonces $d(z) \geq k$ y todos los hijos de z tienen

grado 1.

Caso 2. Si $U_z \neq \emptyset$.

Si $d^-(z)+d^+(z) < k$, el algoritmo elige un vértice $x \epsilon U_z$, bajo la condición que $\varphi(zx)$=mín $\{\varphi(zx_i) \mid x_i \epsilon U_z\}$. Seleccionado x procede a aplicar la adopción Adop(T,z,x,y), donde el vértice z adopta al vértice y tal que y es hijo del vértice x por lo cual se incrementa el grado del vértice z en una unidad. Si $d^-(z) + d^+(z) \geq k$, entonces se llama al algoritmo de aproximación Alg(T, x, k) para cada uno de los vértices $x \epsilon U_z$.

Alg(T, r, k)

$z \leftarrow r$
$U_z \leftarrow \{x \epsilon V(T) \mid z = p(x)$ y $d(x)^+ \geq 1\}$
si $U_z = \varnothing$
    si $d^-(z) + d^+(z) < k$
        Adop(T, $p(z)$, z, x) $\forall$ x que es hijo de z
        salir
    otro $(d^-(z) + d^+(z) \geq k)$
    salir con T
otro $(U_z \neq \varnothing$)
    si $d^-(z) + d^+(z) < k$
        Elegir x tal que $\varphi(zx) =$ mín $\{\varphi(zx_i) \mid x_i \epsilon U_r$ $\}$
        Elegir y tal que $x = p(y)$
        Adop(T, z, x, y)
            si $d^+(y) \geq 1$ y $d^+(x) \geq 1$
                $U_z \leftarrow U_z \cup \{$ y $\}$
            si $d^+(y) = 0$ y $d^+(x) = 0$
                $U_z \leftarrow U_z \setminus \{$ x $\}$
            si $d^+(y) = 0$ y $d^+(x) \geq 1$
                $U_z \leftarrow U_z$
            si $d^+(y) \geq 1$ y $d^+(x) = 0$
                $U_z \leftarrow \{$ $U_z \setminus \{$ x $\}$ $\} \cup \{$ y $\}$
        $d^+(z) = d^+(z)+1$
    otro $(d^-(z) + d^+(z) \geq k)$
        Alg(T, x, k) $\forall$ $x \epsilon U_z$
Fin

A continuación describiremos el algoritmo de aproximación con un ejemplo de una gráfica completa G con n=15 y con $k = 5$.

Los pesos de las aristas de G están dados de la siguiente forma: sea T el árbol generador de G mostrado en la Figura 30; el peso de cada una de las aristas de T es igual a uno y el peso de cualquier otra arista $\varphi(v_m v_n)$ que se encuentra en $E(G) \setminus E(T)$ es igual al número de aristas de la única trayectoria de v_m a v_n que se encuentra en T.

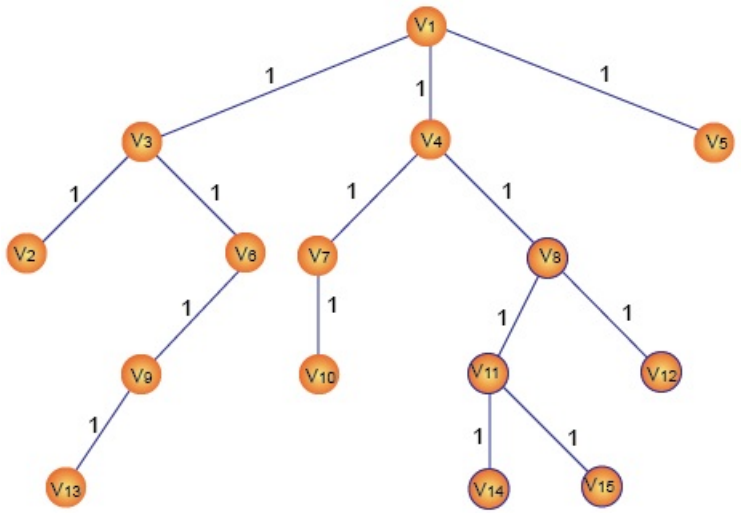


Figura 30: T, $\varphi(T) = 14$

Sin pérdida de generalidad sea $r = v_1$ de T.

Alg$(T, v_1, 5)$

Construimos el conjunto $U_r = U_{v_1} = \{v_3, v_4\}$.

Como $U_{v_1} \neq \emptyset$ y $d^-(v_1) + d^+(v_1) = 0 + 3 < 5$. Elegimos $x \in U_{v_1}$ tal que $\varphi(v_1 x) =$ mín $\{\varphi(v_1 v_3), \varphi(v_1 v_4)\} = \{1, 1\} = 1$, en este caso como el mínimo es uno, entonces podemos elegir a $x = v_3$ o $x = v_4$.

Sea $x = v_3$, elegimos y tal que y es hijo de v_3, sin pérdida de generalidad sea $y = v_2$.

Entonces llamamos a Adop(T, r, x, y) = Adop(T, v_1, v_3, v_2), (Figura 31). Al terminar de aplicar la operación Adop(T, r, x, y), se actualiza el árbol T.

Caemos en el caso $d^+(y) = d^+(v_2)$=0 y $d^+(x) = d^+(v_3) = 1$ por lo que al actualizar U_{v_1} este se queda igual y $d^+(v_1) = d^+(v_1) + 1$.

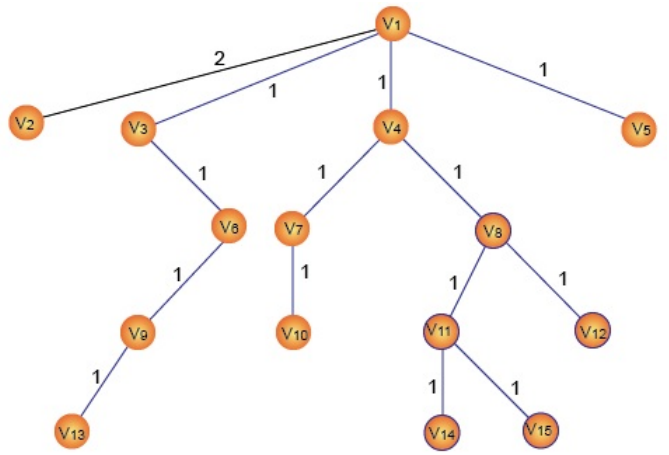


Figura 31: T acualizado, los pesos están indicados junto a las aristas, $\varphi(T) = 15$

Como $U_{v_1} \neq \emptyset$ y $d^-(v_1) + d^+(v_1) = 0 + 4 < 5$, entonces elegimos de nuevo un $x \in U_{v_1}$, dado que U_{v_1} quedó igual, entonces podemos elegir de nuevo $x = v_3$ o $x = v_4$.

Sea $x = v_3$ y $y = v_6$ ya que v_6 es el único hijo de v_3, entonces se llama de nuevo a Adop(T, v_1, v_3, v_6), de nuevo se actualiza T, (Figura 32).

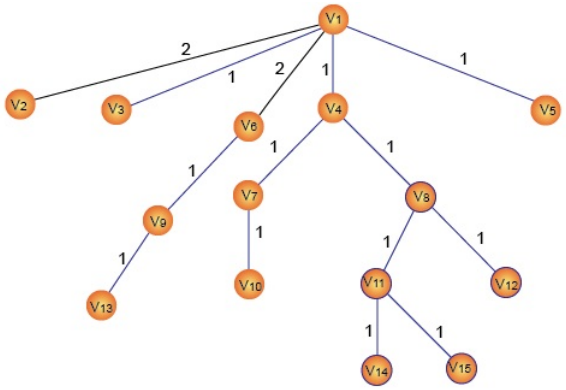


Figura 32: T acualizado, $\varphi(T) = 16$

Caemos en el caso $d^+(y) = d^+(v_6)$=1 y $d^+(x) = d^+(v_3) = 0$ por lo que al actualizar $U_{v_1} = \{U_{v_1} \setminus \{v_3\}\} \cup \{v_6\} = \{v_4, v_6\}$ y $d^+(v_1) = 5$.

Dado que $U_{v_1} \neq \emptyset$ y $d^-(v_1) + d^+(v_1) \geq 5$, caemos en el caso (2ii), donde hacemos un llamado a la recursividad del algoritmo para cada vértice en $U_{v_1} = \{v_4, v_6\}$.

Primero lo hacemos para $\mathbf{Alg}(T, r, k) = \mathbf{Alg}(T, v_4, 5)$.

Construimos el conjunto $U_{v_4} = \{v_7, v_8\}$.

Como $U_{v_4} \neq \emptyset$ y $d^-(v_4) + d^+(v_4) = 1 + 2 < 5$. Elegimos $x \in U_{v_4}$ tal que $\varphi(v_4x) = \text{mín}\{\varphi(v_4v_7), \varphi(v_4v_8)\} = \{1, 1\} = 1$, entonces x puede ser $x = v_7$ o $x = v_8$.

Sea $x = v_7$ y $y = v_{10}$ para poder hacer Adop (T, v_4, v_7, v_{10}) y se actualiza T, (Figura 33).

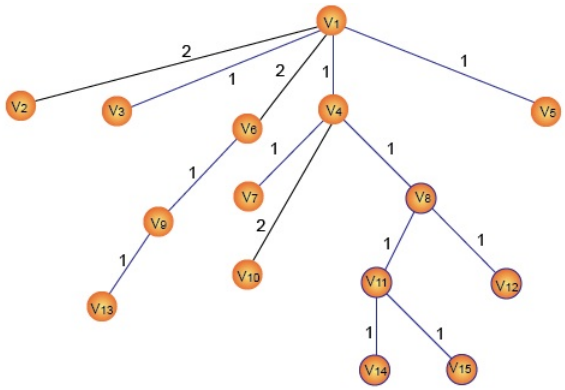


Figura 33: T acualizado, $\varphi(T) = 17$

Tenemos que $d^+(v_{10}) = 0$ y $d^+(v_7) = 0$ por lo que $U_{v_4} = \{U_{v_4} \setminus \{v_7\}\} = \{v_8\}$ y $d^+(v_4) = 3$.

Dado que $U_{v_4} \neq \emptyset$ y $d^-(v_4) + d^+(v_4) = 1 + 3 < 5$. Como $U_{v_4} = \{v_8\}$, entonces $x = v_8$ y sea $y = v_{11}$ para hacer Adop (T, v_4, v_8, v_{11}), (Figura 34).

En consecuencia $d^+(y) = d^+(v_{11}) = 2$ y $d^+(x) = d^+(v_8) = 1$ por lo que $U_{v_4} = \{v_8, v_{11}\}$ y $d^+(v_4) = 4$.

Ya que $U_{v_4} \neq \emptyset$ y $d^-(v_4) + d^+(v_4) = 5$, hacemos uso nuevamente de la recursividad para cada uno de los elementos en U_{v_4}. Sin pérdida de generalidad primero lo hacemos para v_8 y después para v_{11}.

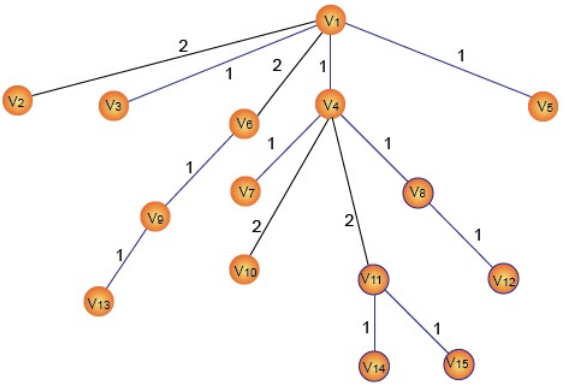


Figura 34: T acualizado, $\varphi(T) = 18$

Alg $(T, v_8, 5)$

Construimos U_{v_8}.

Como $U_{v_8} = \emptyset$ y $d^-(v_8) + d^+(v_8) = 1 + 1 \leq 5$, caemos en el caso (1i), entonces se verifica si $p(v_8)$ existe o no.

El $p(v_8) = v_4$ entonces se llama a $Adop(T, v_4, v_8, v_{12})$, (Figura 35).

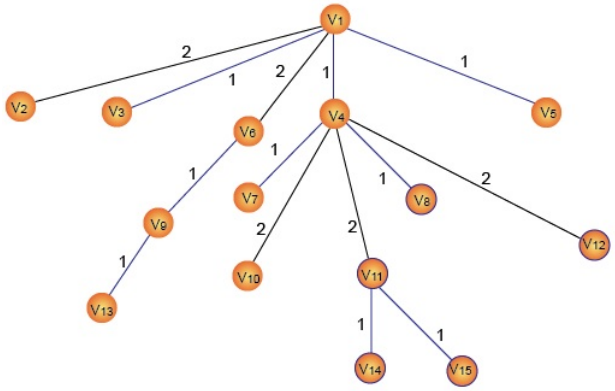


Figura 35: T acualizado, $\varphi(T) = 19$

Después de que se actualiza T nos salimos, para continuar con **Alg** $(T, v_{11}, 5)$.

Construimos $U_{v_{11}}$.

Como $U_{v_{11}} = \emptyset$ y $d^-(v_{11}) + d^+(v_{11}) = 1 + 2 \leq 5$, caemos de nuevo en el caso (1i), entonces se verifica si $p(v_{11})$ existe o no.

El $p(v_{11}) = v_4$ entonces se hace $Adop(T, v_4, v_{11}, v_{14})$ y luego, $Adop(T, v_4, v_{11}, v_{15})$, (Figura 36).
Regresamos a la parte donde por primera vez se hizo la llamada a la recursividad del algoritmo. Hacemos Alg$(T, v_6, 5)$.

Construimos $U_{v_6} = \{v_9\}$. Dado que $U_{v_6} = \{v_9\} \neq \emptyset$ y $d^-(v_6) + d^+(v_6) = 1 + 1 \leq 5$, entonces $x = v_9$ y elegimos y tal que y sea hijo de v_9 por lo que $y = v_{13}$, entonces se aplica $Adop(T, v_6, v_9, v_{13})$, (Figura 37).
Caemos en el caso $d^+(v_{13})$=0 y $d^+(v_9) = 0$, entonces $U_{v_6} = \emptyset$ y $d^+(v_6) = 2$.

Como $U_{v_6} = \emptyset$ y $d^-(v_6) + d^+(v_6) = 1 + 2 < 5$ entonces se verifica si $p(v_6)$ existe o no.

Ya que $p(v_6) = v_1$, entonces se hace $Adop(T, v_1, v_6, v_9)$ y por último $Adop(T, v_1, v_6, v_{13})$, obteniendo el árbol T' de la Figura 38 que es un 5-árbol generador de G con $\varphi(T') = 28$.

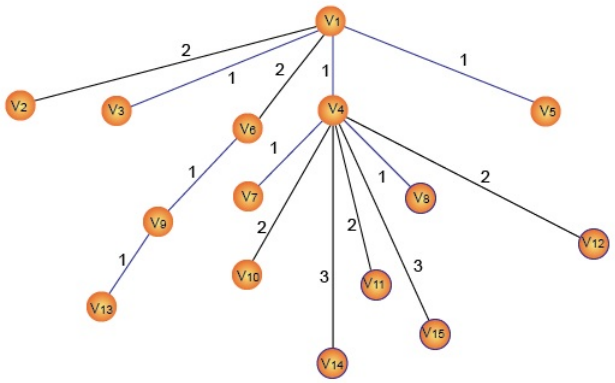


Figura 36: T acualizado, $\varphi(T) = 23$

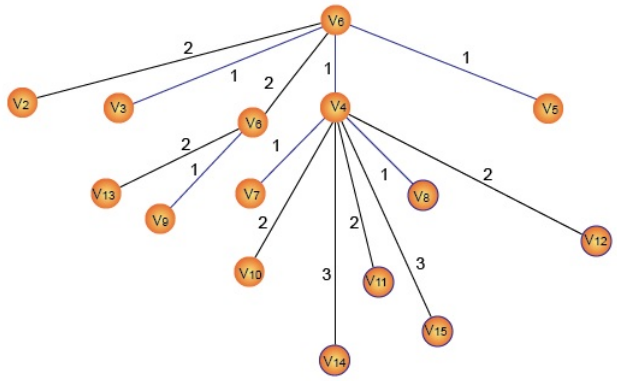


Figura 37: T acualizado, $\varphi(T) = 24$

A continuación demostraremos que para todo árbol T, el árbol T', obtenido al aplicar el algoritmo, satisface $\varphi(T') \leq k(k-1)\varphi(T)$.

Lema 2. *Sean G una gráfica completa con $n \geq 2$ vértices, $1 \leq m \leq n-1$ un entero y φ: $E(G) \to \mathbb{R}^+$ una función de peso en las aristas de G que cumple con la desigualdad del triángulo. Sea A un árbol generador de G con vértice raíz r. Si A^* es una estrella generadora de G con vértice raíz r, entonces $\varphi(A^*) \leq (n-1)\varphi(A)$.*

Demostración.

Sea $x \epsilon V(A) = V(G)$, si el vértice r adopta al vértice x, x es adyacente a u. Sea A' el árbol que resulta después de que el vértice r adopta al vértice u, entonces $\varphi(A') - \varphi(A) \leq \varphi(ru)$, por la operación de adopción y $\varphi(ru)$ es menor o igual a la suma de los pesos de las aristas que se encuentran en la trayectoria $r = u_0, u_1, \ldots, u_j = u$ de r a u en A, es decir $\varphi(ru) \leq \varphi(ru_1) + \varphi(u_1u_2) + \ldots + \varphi(u_{j-1}u)$ por la desigualdad del triángulo. Por otro lado, el peso de cualquier trayectoria en A es menor o igual que $\varphi(A)$, por lo tanto $\varphi(ru) \leq \varphi(A)$.

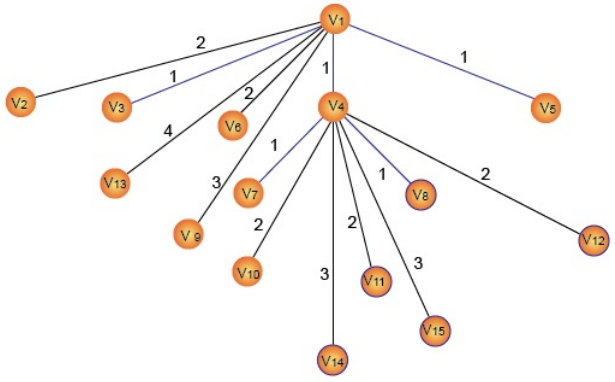


Figura 38: T' es 5-árbol generador de G con $\varphi(T') = 28$

Sea A^* un árbol generador de G que resulta tras las adopciones que realiza r a los vértices que no son adyacentes a él en A. Si $d_A(r) = m$, $m < n-1$, entonces se hace $(n-1)-m$ adopciones, por lo que:

$$\varphi(A^*) \leq \varphi(A) + ((n-1)-m)\varphi(A) = (n-m)\varphi(A) \leq (n-1)\varphi(A). \blacksquare$$

Teorema 2. *Sea G una gráfica completa con $n \geq 3$ vértices y φ: $E(G) \rightarrow \mathbb{R}^+$ una función de peso en las aristas de G que cumple con la desigualdad del triángulo. Sea T un árbol generador de peso mínimo de G con vértice raíz r. Si T' es un k-árbol generador de G obtenido al aplicar el algoritmo de aproximación a T, entonces $\varphi(T') \leq k(k-1)\varphi(T)$.*

Demostración.

La prueba la realizaremos por inducción sobre el número de vértices n de G.

Si $n = 3$, entonces forzosamente $k = 2$, en cuyo caso $T' = T$. Supongamos ahora que $n \geq 4$ y que el teorema es válido para todo árbol con menos de n vértices. Tenemos dos casos:

Caso 1.- Si el $d_T(r) < k$. Sea M el subárbol de T con raíz r y $k+1$ vértices elegidos en los primeros niveles de T a partir del vértice r y cubriendo en su totalidad cada nivel hasta tener los $k+1$ vértices. Observemos que salvo el último nivel podría no ser cubierto en su totalidad si antes se alcanza a tener los $k+1$ vértices.

Sean Q_i y S_j subárboles de T arraigados en los vértices hoja v de M, $1 \leq i \leq m$ y $1 \leq j \leq t$, donde $m + t \leq k$ tal que cada subárbol Q_i tiene al menos k vértices y como vértice raíz al vértice hoja $v = q_i$ de M y S_j tiene 2 o a lo más $k-1$ vértices y como raíz al vértice hoja $v = s_j$ de M, (Figura 39).

Notemos que $\varphi(T) = \varphi(M) + \sum_{i=1}^{m} \varphi(Q_i) + \sum_{j=1}^{t} \varphi(S_j)$.

Sea M^* el árbol que resulta después de que el vértice raíz r de M adopta a todos los vértices x de M que no son adyacentes a él. Si r adopta al vértice x en M y x es adyacente al vértice v, la cota superior del costo de adopción es el peso de la arista rv, por la operación de adopción y $\varphi(rv)$ es menor o igual a la suma de los pesos de las aristas que se encuentran en la trayectoria $r = v_o, v_1, ..v_k = v$, es decir $\varphi(rv) \leq \varphi(rv_1)+\varphi(v_1v_2)+...+\varphi(v_{k-1}v)$, por la desigualdad del triángulo, y el peso de cualquier trayectoria en M es menor o igual a $\varphi(M)$, por lo tanto el peso de cualquier arista en M^* es menor o igual al peso $\varphi(M)$ y $\varphi(M^*) \leq k\varphi(M)$, por el lema 2, notemos que $d_{M^*}(r) = k$, (Figura 40).

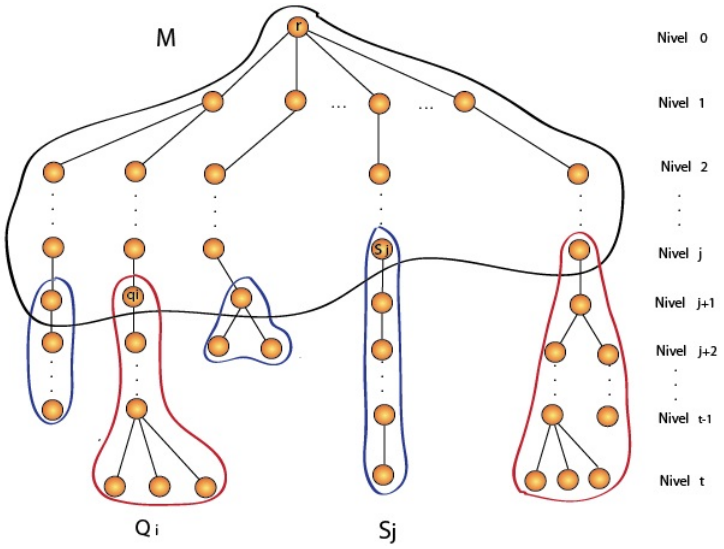


Figura 39: Árbol T.

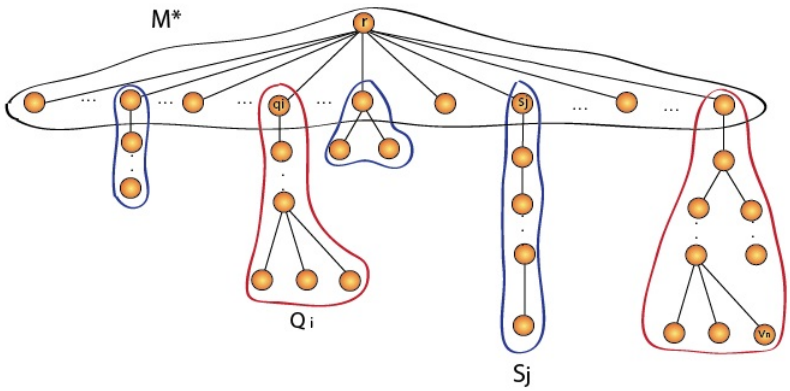


Figura 40

Sea Q_i' el árbol que resulta después de aplicar el algoritmo de aproximación a Q_i, como Q_i tiene al menos k vértices pero no más de $n-1$ vértices, entonces por hipótesis de

inducción se cumple $\varphi(Q'_i) \leq k(k-1)\varphi(Q_i)$.

Sea S^*_j el árbol que resulta después de aplicar el algoritmo de aproximación a S_j, como S_j tiene a lo más $k-1$ vértices la raíz s_j termina por adoptar a todos los vértices de S_j que no son adyacentes a él, entonces por el lema 2, $\varphi(S^*_j) \leq (k-2)\varphi(S_j)$, notemos $d_{S^*{}_j}(s_j) \leq k-2$, (Figura 41).

Sea T^{**} el árbol que resulta después de haber aplicado el algoritmo de aproximación a los subárboles de T y obtener M por M^*, Q_i por Q'_i y S_j por S^*_j.

Notemos que $\varphi(T^{**}) = \varphi(M^*) + \sum_{i=1}^{m} \varphi(Q'_i) + \sum_{j=1}^{t} \varphi(S^*_j)$.

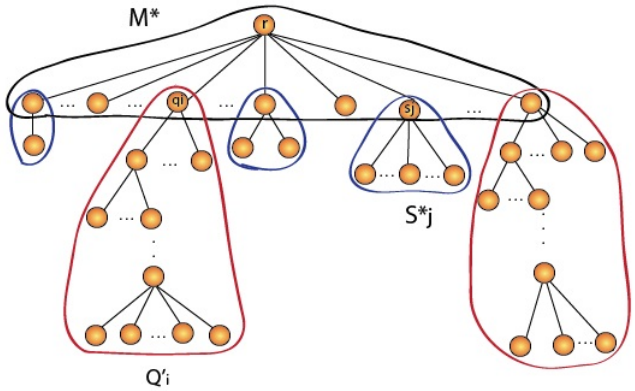


Figura 41: $\varphi(T^{**})$, M^*, R'_i y S^*_j

Finalmente los únicos vértices no terminales de T^{**} cuyos grados no son igual a uno o mayor o igual que k son los vértices s_j, donde $d_{T^{**}}(s_j) < k$, entonces el algoritmo de aproximación hace que el vértice raíz r adopte a cada uno de los hijos de cada s_j.

El costo de adopción por cada vértice hijo de s_j que adopte r es menor o igual a $\varphi(rs_j)$, por la desigualdad del triángulo, pero el $\varphi(rs_j) \leq \varphi(M)$.

Sea T' el árbol que resulta después de que el vértice raíz r adopta a todos los hijos de cada vértice s_j, $1 \leq j \leq t$. T' es el k-árbol de G, (Figura 44).

Notemos que $\varphi(T') \leq \varphi(T^{**}) + \sum_{j=1}^{t} \sum_{i=1}^{k-2} \varphi(rs_j)$

$$\varphi(T') \leq \varphi(T^{**}) + \sum_{j=1}^{t} (k-2)\varphi(rs_j)$$

$$\varphi(T') \leq \varphi(T^{**}) + \sum_{j=1}^{t} (k-2)\varphi(M)$$

$$\varphi(T') \leq \varphi(T^{**}) + t(k-2)\varphi(M) \tag{1}$$

Sustituyendo $\varphi(T^{**})$ por $\varphi(M^*) + \sum_{i=1}^{m} \varphi(Q_i') + \sum_{j=1}^{t} \varphi(S_j^*)$ en (1), obtenemos

$$\varphi(T') \leq \varphi(M^*) + \sum_{i=1}^{m} \varphi(Q_i') + \sum_{j=1}^{t} \varphi(S_j^*) + t(k-2)\varphi(M) \tag{2}$$

Sustituyendo $\varphi(M^*)$ por $k\varphi(M)$, $\varphi(Q_i')$ por $k(k-1)\varphi(Q_i)$ y $\varphi(S_j^*)$ por $(k-2)\varphi(S_j)$ en (2), obtenemos

$$\varphi(T') \leq k\varphi(M) + \sum_{i=1}^{m} k(k-1)\varphi(Q_i) + \sum_{j=1}^{t}(k-2)\varphi(S_j) + t(k-2)\varphi(M)$$

Como $m + t \leq k$, entonces $t \leq k$, por tanto

$$\varphi(T') \leq k\varphi(M) + \sum_{i=1}^{m} k(k-1)\varphi(Q_i) + \sum_{j=1}^{t}(k-2)\varphi(S_j) + k(k-2)\varphi(M)$$

$$\varphi(T') \leq (k^2-k)\varphi(M) + \sum_{i=1}^{m} k(k-1)\varphi(Q_i) + \sum_{j=1}^{t}(k-2)\varphi(S_j)$$

$$\varphi(T') \leq (k^2-k)\varphi(M) + k(k-1)\sum_{i=1}^{m} \varphi(Q_i) + (k-2)\sum_{j=1}^{t} \varphi(S_j)$$

$$\varphi(T') \leq k(k-1)\varphi(M) + k(k-1)\sum_{i=1}^{m} \varphi(Q_i) + k(k-1)\sum_{j=1}^{t} \varphi(S_j)$$

$$\varphi(T') \leq k(k-1)(\varphi(M) + \sum_{i=1}^{m} \varphi(Q_i) + \sum_{j=1}^{t} \varphi(S_j))$$

$$\varphi(T') \leq k(k-1)\varphi(T).$$

Caso 2.- Si el $d_T(r) \geq k$. Sea M el subárbol de T con vértice raíz r y por los l vértices adyacentes a r, $l \geq k$.

Notemos que $\varphi(M) = \sum_{i=1}^{l} \varphi(rv_i)$.

Sean Q_i y S_j subárboles de T, arraigados en los vértices hoja v de M, $1 \leq i \leq m$ y $1 \leq j \leq t$, donde $m + t \leq l$ tal que cada subárbol Q_i tiene al menos k vértices y como vértice raíz al vértice hoja $v = q_i$ de M y S_j tiene 2 vértices o a lo más $k-1$ vértices y como vértice raíz al vértice hoja $v = s_j$, (Figura 42).

Notemos que $\varphi(T) = \varphi(M) + \sum_{i=1}^{m} \varphi(Q_i) + \sum_{j=1}^{t} \varphi(S_j)$, donde

$\varphi(M) = \sum_{i=1}^{l} \varphi(rv_i) \geq \sum_{i=1}^{m} \varphi(rq_i) + \sum_{j=1}^{t} \varphi(rs_j)$.

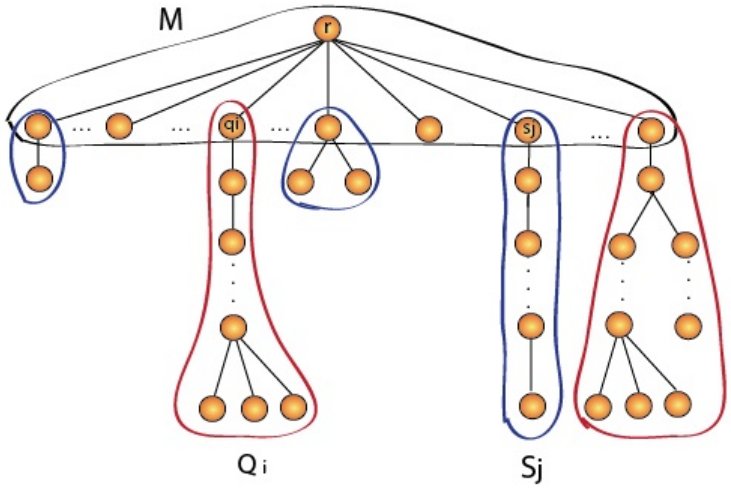


Figura 42: Árbol T

Sea Q_i' el árbol que resulta después de aplicar el algoritmo de aproximación a Q_i, como Q_i tiene al menos k vértices pero no más de $n-1$ vértices, entonces por hipótesis de inducción se cumple $\varphi(Q_i') \leq k(k-1)\varphi(Q_i)$.

Sea S_j^* el árbol que resulta después de aplicar el algoritmo de aproximación a S_j, como S_j tiene a lo más $k-1$ vértices, la raíz s_j termina por adoptar a los vértices de S_j que no son adyacentes a s_j, entonces por el lema 2, $\varphi(S_j^*) \leq (k-2)\varphi(S_j)$, donde $d_{S_j^*}(s_j) \leq k-2$.

Sea T^{**} el árbol que resulta después de aplicar el algoritmo a los subárboles Q_i y obtener Q_i' y S_j por S_j^*,(Figura 43).

Notemos que $\varphi(T^{**}) = \varphi(M) + \sum_{i=1}^{m} \varphi(Q_i') + \sum_{j=1}^{t} \varphi(S_j^*)$.

Los únicos vértices no terminales de T^{**} cuyos grados no son igual a uno o mayores o iguales que k son los vértices s_j, es decir $d_{T^{**}}(s_j) < k$, por lo que el vértice raíz r adopta a cada una de los hijos de cada s_j.

El costo de adopción por cada vértice hijo de s_j que adopte r es menor o igual $\varphi(rs_j)$, por la desigualdad del triángulo.

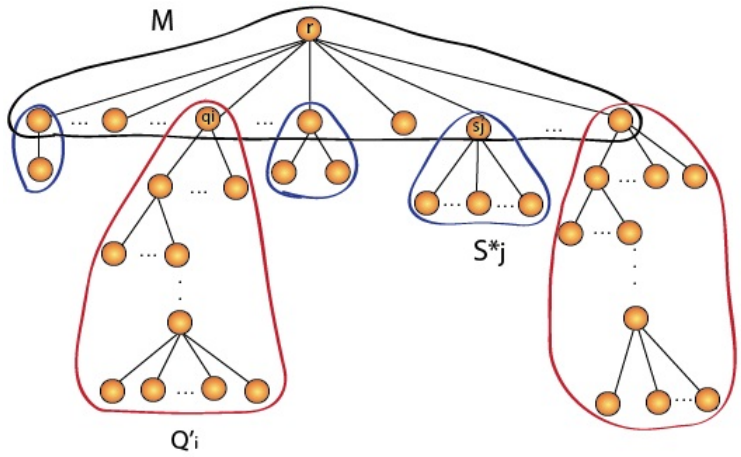


Figura 43: $\varphi(T^{**})$, M, R'_i y S^*_j

Sea T' el árbol que resulta después de que el vértice r adopta a todos lo hijos de cada s_j, $1 \leq j \leq t$. T' es el k-árbol de G, (Figura 44).

Notemos que $\varphi(T') \leq \varphi(T^{**}) + \sum_{j=1}^{t} \sum_{i=1}^{k-2} \varphi(rs_j)$

$$\varphi(T') \leq \varphi(T^{**}) + \sum_{j=1}^{t} (k-2)\varphi(rs_j) \tag{1}$$

Sustituyendo $\varphi(T^{**})$ por $\varphi(M) + \sum_{i=1}^{m} \varphi(Q'_i) + \sum_{j=1}^{t} \varphi(S^*_j)$ en (1), obtenemos

$$\varphi(T') \leq \varphi(M) + \sum_{i=1}^{m} \varphi(Q'_i) + \sum_{j=1}^{t} \varphi(S^*_j) + \sum_{j=1}^{t} (k-2)\varphi(rs_j) \tag{2}$$

Sustituyendo $\varphi(Q'_i)$ por $k(k-1)\varphi(Q_i)$ y $\varphi(S^*_j)$ por $(k-2)\varphi(S_j)$ en (2), obtenemos

$$\varphi(T') \leq \varphi(M) + \sum_{i=1}^{m} k(k-1)\varphi(Q_i) + \sum_{j=1}^{t} (k-2)\varphi(S_j) + \sum_{j=1}^{t} (k-2)\varphi(rs_j)$$

$$\varphi(T') \leq \varphi(M) + k(k-1) \sum_{i=1}^{m} \varphi(Q_i) + (k-2) \sum_{j=1}^{t} \varphi(S_j) + (k-2) \sum_{j=1}^{t} \varphi(rs_j) \tag{3}$$

Retomando que $\varphi(M)$ es igual a $\sum_{i=1}^{l} \varphi(rv_i) \geq \sum_{i=1}^{m} \varphi(rq_i) + \sum_{j=1}^{t} \varphi(rs_j)$, tenemos que

$\sum_{j=1}^{t} \varphi(rs_j) \leq \sum_{i=1}^{l} \varphi(rv_i) - \sum_{i=1}^{m} \varphi(rq_i)$, entonces

$$\sum_{j=1}^{t}\varphi(rs_j) \leq \sum_{i=1}^{l}\varphi(rv_i) = \varphi(M), \text{ por tanto } \sum_{j=1}^{t}\varphi(rs_j) \leq \varphi(M) \quad (4)$$

Sustituyendo (4) en (3), obtenemos

$$\varphi(T') \leq \varphi(M) + k(k-1)\sum_{i=1}^{m}\varphi(Q_i) + (k-2)\sum_{j=1}^{t}\varphi(S_j) + (k-2)\varphi(M)$$

$$\varphi(T') \leq (k-1)\varphi(M) + k(k-1)\sum_{i=1}^{m}\varphi(Q_i) + (k-2)\sum_{j=1}^{t}\varphi(S_j)$$

$$\varphi(T') \leq k(k-1)\varphi(M) + k(k-1)\sum_{i=1}^{m}\varphi(Q_i) + k(k-1)\sum_{j=1}^{t}\varphi(S_j)$$

$$\varphi(T') \leq k(k-1)(\varphi(M) + \sum_{i=1}^{m}\varphi(Q_i) + \sum_{j=1}^{t}\varphi(S_j))$$

$\varphi(T') \leq k(k-1)\varphi(T)$.■

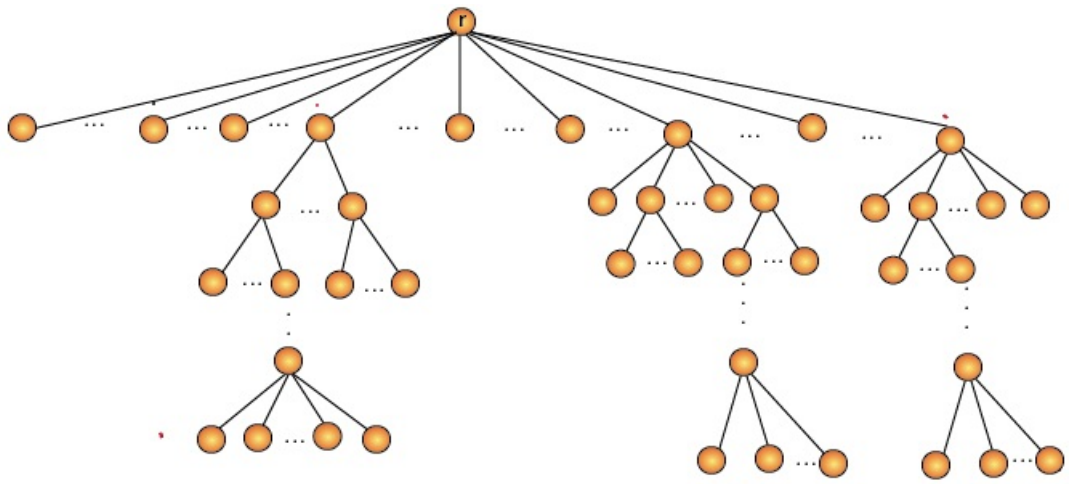


Figura 44: T el k-árbol de G, $\varphi(T') \leq k(k-1)\varphi(T)$

El peor caso para encontrar el k-árbol de G es el siguiente.

Sea $k \geq 2$ un entero, consideremos el árbol T con $n = 1 + k(k-1)$ vértices, (Figura 45).

Sea G la gráfica con $V(G) = V(T)$ con la siguiente función de peso φ.

1.- $\varphi(rv_0) = 1$.
2.-Si $uv \in E(T)$ con $u \neq r \neq v$, entonces $\varphi(ur) = 0$.
3.-Si $uv \notin E(T)$, entonces $\varphi(uv)$ es el peso de la única trayectoria de u a v en T.

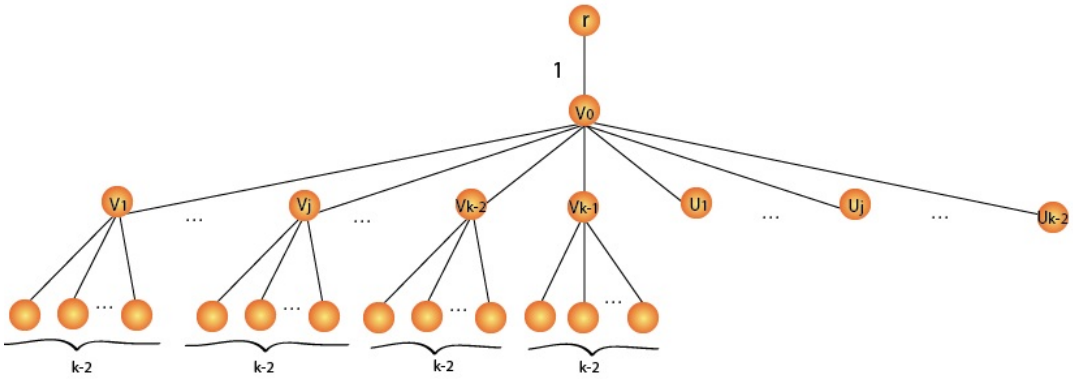


Figura 45: Árbol T, $\varphi(T) = 1$.

Notemos que φ satisface la desigualdad del triángulo.
Al aplicar el algoritmo de aproximación al árbol T con raíz r, el vértice r adopta primero a los vértices $v_1, ..., v_{k-1}$, obteniendo el árbol T^*, (Figura 46).

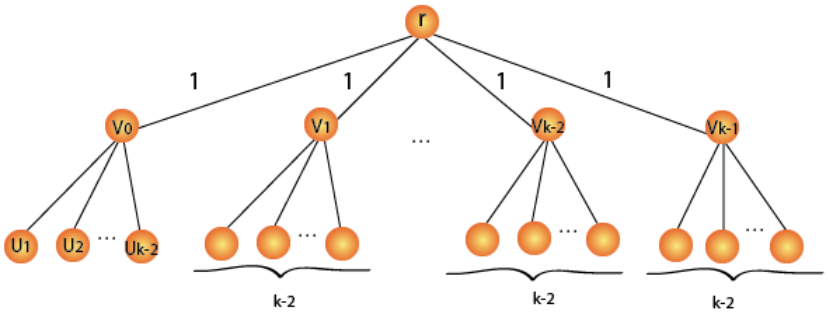


Figura 46: Árbol T^*.

Para cada $i = 0, ..., k-1$ de $d_{T^*}(v_i) = k-1$, por lo cual el vértice r adopta a todos los $k-2$ hijos de v_i. De esta forma el árbol final T' resulta ser el k-árbol de G, (Figura 47). Por la definición de φ, $\varphi(rx) = 1$, para todo $x \in V(T)$. Por lo tanto $\varphi(T') = (n-1) = k(k-1)\varphi(T)$.

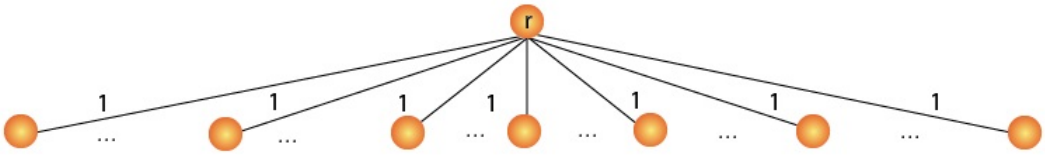


Figura 47: T' es el k-árbol de G, $\varphi(T') = k(k-1)\varphi(T)$.

Apéndice A

Ejemplos con el programa Algoritmo de Aproximación

Aquí daremos algunos ejemplos donde utilizamos el programa del algoritmo de Aproximación para encontrar el k-árbol de una gráfica completa.

Ejemplo 1.- Vamos a considerar el mismo ejemplo de la Figura 30, pero teniendo en cuenta esta vez al vértice v_3 como la raíz de T de G, (Figura 48).

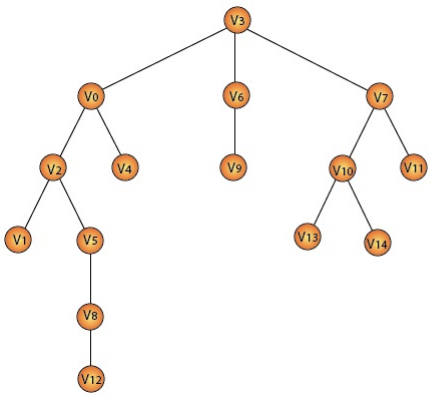
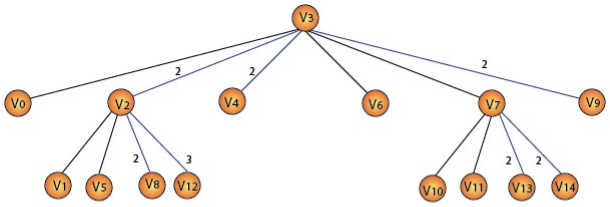


Figura 48: T, $\varphi(T) = 14$



T' es el 5-árbol de G, $\varphi(T') = 22$.

Ejemplo 2.- Sea F una gráfica completa con $n = 22$, donde los pesos de las aristas de F están dados de la siguiente manera: sea H el árbol generador de peso mínimo de F, (Figura 49); el peso de cada una de las aristas de F es igual a uno y el peso de cualquier otra arista $\varphi(uv)$ que se encuentra $E(F) \setminus E(H)$ es igual al número de aristas que se encuentran en la única trayectoria de u a v en H. Encontrar el 4-árbol de F.

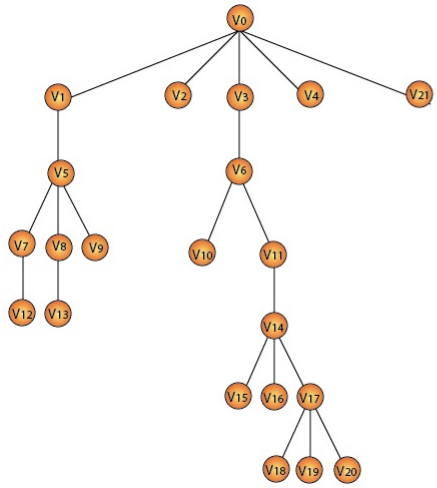
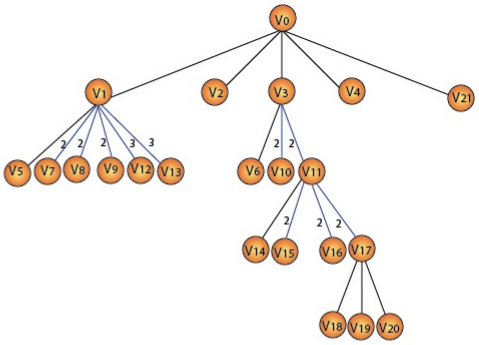


Figura 49: H, $\varphi(H) = 22$



H' es el 4-árbol de F, $\varphi(H') = 33$.

Ejemplo 3.- Sea D una gráfica completa con función de peso φ dada por la tabla 5, (Figura 50). Sea J el árbol de peso mínimo de D, primero vamos a elegir al vértice v_5 como vértice raíz de J y luego al vértice v_{16}. Encontrar el 7-árbol de D, (Figura 52 y Figura 53).

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17
V0	∞	9	6	12	23	1	5	1	25	8	2	4	2	1	9	21	35	1
V1	9	∞	13	2	8	36	7	5	23	29	4	19	7	33	21	17	34	10
V2	6	13	∞	37	21	8	40	17	23	6	2	9	23	31	12	11	2	10
V3	12	2	37	∞	1	37	1	7	6	8	14	6	21	26	4	5	4	5
V4	23	8	21	1	∞	16	13	11	2	8	5	16	20	11	18	2	1	13
V5	1	36	8	37	16	∞	29	19	1	10	35	36	1	4	1	17	12	3
V6	5	7	40	1	13	29	∞	4	23	10	40	15	21	2	27	40	5	22
V7	1	5	17	7	11	19	4	∞	27	13	19	30	33	8	9	21	10	2
V8	25	23	23	6	2	1	23	27	∞	12	1	20	7	4	11	15	22	15
V9	8	29	6	8	8	10	10	13	12	∞	21	14	17	5	12	9	13	10
V10	2	4	2	14	5	35	40	19	1	21	∞	11	31	7	6	8	6	4
V11	4	19	9	6	16	36	15	30	20	14	11	∞	18	3	15	2	1	19
V12	2	7	23	21	20	1	21	33	7	17	31	18	∞	4	8	7	20	9
V13	1	33	31	26	11	4	2	8	4	5	7	3	4	∞	5	5	7	1
V14	9	21	12	4	18	1	27	9	11	12	6	15	8	5	∞	9	2	21
V15	21	17	11	5	2	17	40	21	15	9	8	2	7	5	9	∞	20	4
V16	35	34	2	4	1	12	5	10	22	13	6	1	20	7	2	20	∞	5
V17	1	10	10	5	13	3	22	2	15	10	4	19	9	1	21	4	5	∞

Figura 50: Tabla 5.

Ejemplo 4.- Sea Z una gráfica completa con función de peso φ dada por la tabla 6, (Figura 51). Sea Y el árbol de peso mínimo de Z, consideremos primero a v_{11} como el vértice raíz de Y y luego a v_7. Encontrar el 11-árbol de Z,(Figura 54 y Figura 55).

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
V0	∞	3	3	7	9	4	7	8	10	3	4	7	2
V1	3	∞	2	2	5	9	10	6	1	5	8	12	9
V2	3	2	∞	8	1	7	5	12	6	2	12	1	3
V3	7	2	8	∞	11	4	15	1	12	15	6	11	14
V4	9	5	1	11	∞	11	8	2	8	7	6	3	4
V5	4	9	7	4	11	∞	7	10	1	10	2	12	9
V6	7	10	5	15	8	7	∞	2	4	9	9	9	7
V7	8	6	12	1	2	10	2	∞	5	3	8	1	1
V8	10	1	6	12	8	1	4	5	∞	4	6	1	1
V9	3	5	2	15	7	10	9	3	4	∞	3	1	1
V10	4	8	12	6	6	2	9	8	6	3	∞	1	2
V11	7	12	1	11	3	12	9	1	1	1	1	∞	2
V12	2	9	3	14	4	9	7	1	1	1	2	2	∞

Figura 51: Tabla 6.

A continuación se muestra las soluciones del ejemplo 3 y 4.

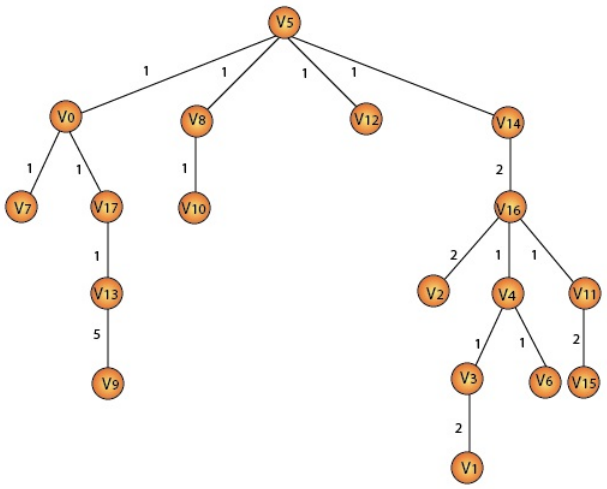
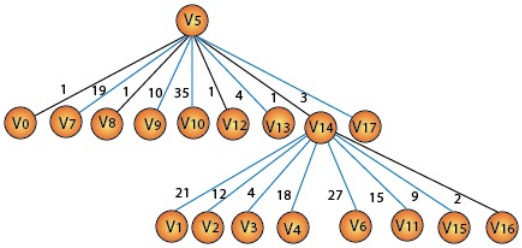


Figura 52: J, $\varphi(J) = 25$



J' es el 7-árbol de D, $\varphi(J') = 183$.

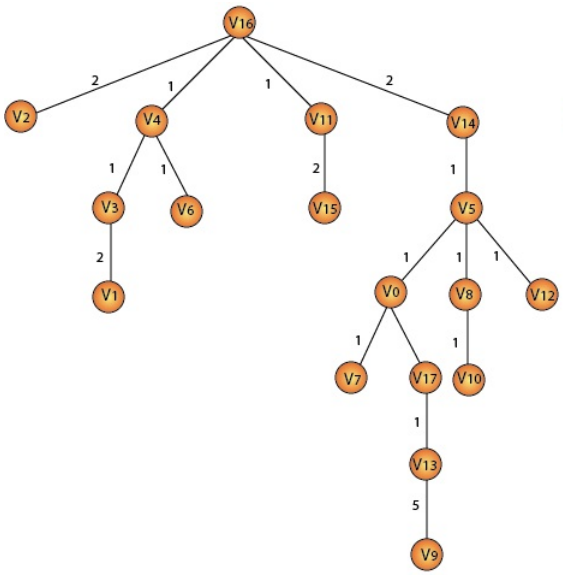
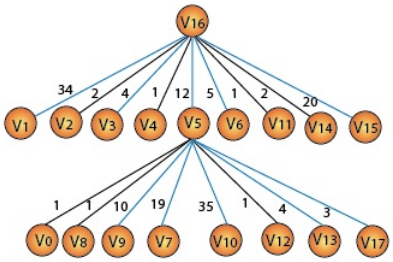


Figura 53: J, $\varphi(J) = 25$



J' es el 7-árbol de D, $\varphi(J') = 155$.

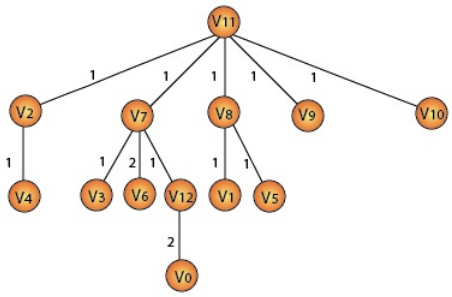
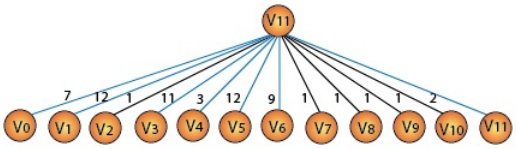


Figura 54: Y, $\varphi(Y) = 14$



Y' es el 11-árbol de Z, $\varphi(Y') = 61$.

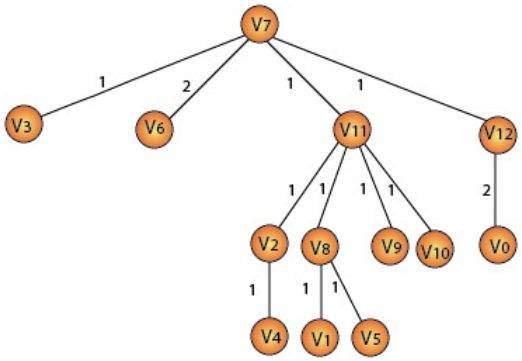
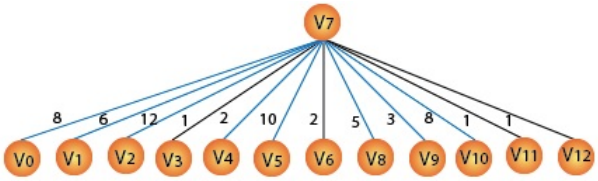


Figura 55: Y, $\varphi(Y) = 14$



Y' es el 11-árbol de Z, $\varphi(Y') = 59$.

Apéndice B

Programa del Algoritmo de Aproximación

A continuación presentamos de manera general las estructuras y funciones utilizadas en el programa del algoritmo de Aproximación para encontrar el k-árbol de una gráfica G. El algoritmo fue programado en lenguaje C.

La gráfica G se lee de un archivo de texto ASCII (nombre.txt), del cual se obtiene el número de vértices y el peso de cada arista de G.

El algoritmo de aproximación, necesita como entrada al árbol generador de peso mínimo T de G.

T de G es obtenido a través de la función **arbolExpansionKruskal** en la implementación del algoritmo de Kruskal dado por Aguilar et al [6]. T es almacenado en un arreglo de estructuras de tipo Arco, en las que se agrupan los vértices y los pesos asociados a las aristas que pertenecen a T de G.

```
typedef struct{
    int u;                          //v\'ertice  u.
    int v;                          //v\'ertice  v.
    float peso;          //peso de la arista.
} Arco;
```

Cada vértice de T es una estructura de datos de tipo Nodo, donde se almacena el in-grado, ex-grado, el peso de la arista entre el padre del vértice y una lista de los hijos del vértice.

```
typedef struct nodo Nodo;
typedef struct hijo Hijo;
typedef Hijo * LHIJOS;

struct nodo{
        int ingrado;
```

```
        int exgrado;
        float w;        //peso de la arista con el padre
        int nod;        //id vértice
        LHIJOS hijos;//apuntador a lista de hijos
};
```

Los hijos de cada vértice se encuentran en una lista ligada de apuntadores, cada elemento de la lista tiene la siguiente estructura

```
typedef Nodo * ARBOL;
struct hijo{
        ARBOL nod;      //apuntador a nodo hijo (subárbol)
        Hijo *sig;      //apuntador al siguiente elemento de la lista
};
```

A continuación se muestra la función **algAprox** de la codificación del Algoritmo de Aproximación.

```
1  int algAprox(ARBOL T, int r, int k, int orden, ARBOL pr, GRAF mtz)
2  {
3          int Ur[orden];
4          int cardU,i;
5          ARBOL x,y;//x hijo, y nieto
6          LHIJOS aux;
7          int continua;
8
9          continua=1;
10         while(continua){
11
12                 cardU=obtenUr(Ur,T,orden);
13
14                 if(cardU == 0){
15                         if((T->ingrado + T->exgrado)<k){
16                                 if(T->ingrado == 1){
17
18                                         adopTodos(T,pr,mtz);
19
20                                         return 1;
21                                 }
22                                 else{//Es el nodo raíz, su ingrado es
                                       cero.
23                                         printf("No exite el k-árbol
                                            generador de peso mínimo de G
                                             para k >= n\n");
24                                         exit(0);
25                                 }
26                         }
27                         else{//ingrado+exgrado >= k,
28                                 return 1;
29                         }
30                 }
```

```
31                      else{//Ur != 0
32                              if( T->ingrado + T->exgrado < k)
33                                      x=seleccionaX(Ur,mtz,T->nod,orden,T);
34
35                                      y=seleccionaY(x);
36
37                                      adop(T,x,y,mtz);
38                                      actualizaUr(Ur,x,y,&cardU);
39                              }
40                              else{//ingrado+exgrado >=k
41                                      aux = T->hijos;
42                                      do{
43                                              if(Ur[aux->nod->nod]==1){//
                                                    Llamado recursivo
44                                                      Ur[aux->nod->nod]=0;
45                                                      cardU = cardU -1;
46                                                      alg(aux->nod,aux->nod->
                                                            nod,k,orden,T,mtz);
47                                              }
48                                              aux = aux->sig;
49                                      }while(aux != NULL);
50
51                                      return 1;
52                              }
53                      }
54
55                      printf("\n");
56
57              }
58  }
```

obtenUr: Es una función cuya entrada nos da la cardinalidad del conjunto U_r del vértice r.

adopTodos : Es una función donde el $p(r)$ termina por adoptar a todos lo hijos de r.

seleccionaX: Es una función que selecciona a x hijo de r.

seleccionaY: Es una función que selecciona a y hijo de x.

Bibliografía

[1] J. A. Bondy, U. S. R. Murty, Graph Theory with Applications, North Holland, 1982.

[2] N. Christofides, Graph Theory An Algorithmic Approach, Academic Press, 1975.

[3] Robin J. Wilson, Introduction to Graph Theory, Longman Group Ltd. 1972.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, 2ed. MIT Press, 2001.

[5] Sándor P. Fekete, Samir Khuller, Monika Klemmstein, Balaji Raghavachari, Neal Young, A Network-Flow Technique for Finding Low-Weight Bounded-Degree Spanning Trees, Journal of Algorithms, 24 No 2 (1997), 310–324.

[6] Luis J. Aguilar, Ignacio Z. Martínez, Algoritmos, estructuras de datos. Una perspectiva en C, McGraw-Hill, 1ed. 2004.