Conjunto dominante conexo de un grafo

Javier Díez Corral

Trabajo de fin de grado en Matemáticas
Universidad de Zaragoza

Director del trabajo: Alfredo Martín García Olaverri
24 de junio de 2020

Prólogo

Un conjunto dominante conexo de un grafo $G = (V, E)$ es un subconjunto de vértices D que posee la propiedad de que todo vértice que no pertenece al conjunto D es adyacente a al menos un vértice de D y además el subgrafo inducido por D es conexo. Las primeras referencias a este tipo de conjuntos se encuentran en la segunda mitad del siglo XX, en particular fue O.Ore quien en su libro *Theory of Graphs* introdujo el termino de conjunto dominante. El principal interés de este tipo de conjuntos en teoría de grafos viene del conocido como problema del conjunto dominante conexo que consiste en hallar un conjunto dominante conexo de un grafo G de cardinalidad mínima. Este problema tiene su principal aplicación en la optimización del rendimiento de redes de comunicación, en particular de las redes ad-hoc inalámbricas las cuáles no dependen de una infraestructura previa, como por ejemplo la conexión bluetooth de los móviles.

Este problema es conocido por ser NP-Completo, por lo que el principal ámbito de estudio alrededor del problema es el de intentar encontrar algoritmos que permitan obtener una solución aproximada en tiempo polinómico. Este es el objetivo principal que inspira este trabajo. Nos centraremos primeramente en la demostración de que efectivamente este problema pertenece a la clase de complejidad NP-Completo. Así mismo, describiremos y analizaremos dos algoritmos presentados por S. Guha y S. Khuller que obtienen una solución aproximada al problema en tiempo polinomial. Introduciremos además la formulación equivalente del problema del conjunto dominante conexo y presentaremos un algoritmo que nos permite obtener una solución aproximada a este problema en tiempo lineal.

Incluiremos también en el trabajo los conceptos de triangulación y casi-triangulación desde el punto de vista de la teoría de grafos y estudiaremos el problema del conjunto dominante conexo en este ámbito donde el resultado más importante a tratar será el de la existencia de árboles de expansión sin vértices de grado dos.

Summary

A connected dominating set of a graph $G = (V,E)$ is a subset D of vertices such that every vertex that is not in D is adjacent to at least one vertex of D and also the subgraph induced by D is connected. In graph theory, this definition induces the problem known as the connected dominating set problem which asks for a connected dominating set of minimum size in a graph. This problem is known to be NP-Complete. The main purpose of this work resides in the description and analysis of polynomial time algorithms that provide an approximate solution of this problem.

First of all we introduce some basic definitions and concepts of graphs as well as the connected dominating set problem. The main application of this problem is found in the area of communications. In particular, we focus on wireless sensor networks where the different sensors are the nodes of the graph and we show that finding disjoints connected dominating set of minimum size in that graph form virtual backbones that act as a switch and provide a longer time of life of the network.
The connected dominating set problem can be reformulated as the problem of finding a spanning tree T with maximum number of leaves where the non-leaves vertices of T are the vertices that form the connected dominating set of minimum size of the initial graph.

The main body of this work is focused on the study of NP-Completeness of the connected dominating set problem and the analysis of some aproximation algorithms for this problem. The class of complexity P and NP contains decision problems which can be solved in polynomial time and which can verify an affirmative solution in polynomial time respectively. Besides, a decision problem Q belongs to NP-Complete if belongs to NP and any problem in NP can be reduced in polynomial time to Q.

The tool which will let us to prove that the connected dominating set is NP-Complete is the polynomial time reduction of this problem to another that is known to be NP-Complete. The polynomial time reduction of a problem consists in reformulate an instance of a problem Q in polynomial time to be the instance of a problem Q' which gives a solution to this problem.

We prove that the problem of finding a spanning tree T where all vertices have degree one or three in a planar graph with maximum degree three is NP-Complete by reducing it to the Hamiltonian Path Variant problem which belongs to NP-Complete. Thus, we will conclude that the connected dominating set problem is NP-Complete.

Since this problem is NP-Complete an algorithm that provides an exact solution in polynomial time is unlikely to exist. Therefore, we present three polynomial algorithms that provide an approximate solution. First one consists of, given a graph, growing a tree starting from the vertex of maximum degree and by coloring the chosen vertices black and its neighbors gray in each step. The set of black nodes at the end of the algorithm will form the connected dominating set. We show that this algorithm provides an approximation factor of $2(1+H(\Delta))$.

Second one is an alternative way to grow a tree that first forms separate sets that form a dominating set in the graph and then the algorithm connect them together to form a connected dominating set. We prove that this algorithm gives an aproximation factor of $(\ln\Delta+3)$.

Last one is a linear time algorithm which gives an approximate solution to the equivalent problem of finding a spanning tree with maximum number of leaves in a graph. The algorithm first builds a forest by expanding the vertices that belong to a tree and then connects the trees that are in the forest. The spanning tree grown by the algorithm has at least third of the leaves that has the spanning tree with maximum number of leaves.

Finally, in Chapter 3, we study the connected dominating set in triangulations and near-triangulations. In graph theory a triangulation is a maximal planar graph such that when drawn on a plane all its faces are triangles. A near-triangulation is a planar graph such that all its faces except of the unbounded face are triangles. We focus on the equivalent formulation of the connected dominating set problem. In relation with that, we prove that every near-triangulation with at least four vertices contains a spanning tree with no vertices of degree two.

Índice general

Capítulo 1

Conjunto dominante y conjunto dominante conexo de un grafo. Aplicaciones

Comenzamos este capítulo repasando alguna de las definiciones y conceptos de la teoría de grafos que usaremos a lo largo de este trabajo. También definiremos lo que es un conjunto dominante conexo de un grafo e introduciremos el problema del conjunto dominante conexo comentando también algunas de sus aplicaciones más importantes. Por último, veremos una formulación equivalente a este problema.

1.1. Definiciones

Definición 1.1. Un grafo G se define como un par (V,E) donde:

- V es un conjunto de elementos llamados vértices.
- E es un conjunto de pares de vértices $E \subseteq \{(u,v) : u,v \in V, u \neq v\}$, el conjunto de aristas del grafo. Denotaremos a una arista $e \in E$ como $e = (u,v)$.

El número de elementos de V se llama orden del grafo y se denota como $|V|$. Diremos que dos vertices u y v son adyacentes si $(u,v) \in E$. En este caso, también se dice que u y v son vecinos.

Definición 1.2. Se define el grado de un vértice $v \in V$ como el número de aristas que inciden en v. Se denota como gr(v). Dado un grafo G, llamamos:

- $\Delta(G)$ al máximo grado de todos los vértices de G. Es decir, $\Delta(G) = max\{\text{gr}(v), v \in V\}$.
- $\delta(G)$ al mínimo grado de todos los vértices de G. Es decir, $\delta(G) = min\{\text{gr}(v), v \in V\}$.

Definición 1.3. Dados dos vertices $u,v \in V$ un camino desde u hasta v es una sucesión de vértices $u = u_0, u_1, ..., u_k = v$ tal que $(u_i, u_{i+1}) \in E \ \forall i = 0, ..., k-1$. Diremos que un camino es simple si todos los vértices del camino son distintos. La longitud del camino es el número de aristas que tiene.
Un ciclo es un camino donde el vértice inicial coincide con el final.
Un grafo es conexo si dados dos vértices cualesquiera siempre existe un camino que los une. La distancia entre dos vértices es el número de aristas que tiene un camino de longitud mínima que une dichos vértices.
Un grafo es completo si todas las posibles parejas de vértices son aristas. Denotaremos como K_n al grafo completo de orden n.

Definición 1.4. Sea $G = (V,E)$ un grafo. Un subgrafo de G es otro grafo $G' = (V',E')$ con $V' \subseteq V$ y $E' \subseteq E$.

Definición 1.5. Un árbol es un grafo no dirigido conexo que no tiene ciclos, es decir, para cada vértice v del grafo no hay un camino que empiece y termine en v. A un grafo no dirigido que no tiene ciclos se le llama bosque.
Un árbol de expansión T de un grafo conexo G es un árbol que contiene a todos los vértices de G.

Definición 1.6. Un conjunto dominante de un grafo $G = (V,E)$ es un subconjunto $D \subseteq V$ tal que todo vértice que no está en D es adyacente a al menos un vértice de D. Un conjunto dominante se dice conexo si los vértices que están en D forman un subgrafo conexo.
El número dominante de un grafo G, denotado como $\gamma(G)$, es el número de vértices del conjunto dominante de cardinal mínimo de un grafo G.

El problema de optimización del conjunto dominante de un grafo $G = (V,E)$ consiste en hallar un conjunto dominante de cardinal mínimo. El problema de decisión asociado trata de verificar si $\gamma(G) \leq k$ para un grafo G y un entero k dados. Este trabajo se centrará únicamente en el estudio del problema del conjunto dominante conexo. Como veremos en el segundo capítulo, este problema pertenece a la clase de problemas NP, en particular es un problema NP-Completo

1.2. Aplicaciones

1.2.1. Redes ad hoc

La principal aplicación de los conjuntos dominantes conexos la encontramos en el campo de redes inalámbricas. En particular, son útiles para el cálculo de un camino para el tráfico de datos dentro de redes ad hoc móviles. Una red ad hoc se caracteriza por no tener un nodo central sino que cada nodo dentro de la red tiene la misma jerarquía y es libre de asociarse con cualquier otro dispositivo de la red ad hoc en su rango de alcance. El ejemplo más común de hoy en día son las transferencias vía bluetooth a través de dispositivos móviles.
Dentro de las redes ad hoc, nos centraremos en redes de sensores, las cuales están formadas por sensores que trabajan en una tarea común y que tienen un único nodo de salida. El problema es el de encontrar subconjuntos disjuntos de nodos que necesitan estar siempre activos para asegurar la conectividad de la red. Si se logran encontrar varios subconjuntos, entonces se extenderá el tiempo de vida de la red. En efecto, pues como los subconjuntos son disjuntos no es necesario que estén activos todos los sensores de la red para el tráfico de datos por la red, sino que basta con tener activos los sensores de uno de esos subconjuntos durante un periodo de tiempo e ir intercambiando entre el resto de subconjuntos de la red. Como estos subconjuntos son conjuntos dominantes conexos, por definición, todo nodo de la red que no esté en el subconjunto será adyacente a un nodo que si pertenece al subconjunto. Por tanto, al ir intercambiando entre estos conjuntos nos aseguramos que la red seguirá estando conectada.
Así, se producirá un buen balance energético y por tanto se alargará la vida de la red.
Este problema se puede trasladar a uno de encontrar el mínimo conjunto dominante conexo de un grafo de la siguiente forma:
Consideramos el grafo $G = (V,E)$ donde el conjunto de nodos, V, está formado por los diferentes sensores que forman la red y E, el conjunto de aristas del grafo, son las diferentes conexiones entre los sensores de la red. Es decir, dos sensores forman una arista del grafo si se pueden transmitir datos entre dichos sensores. Entonces, el problema trata de encontrar el mínimo conjunto dominante conexo del grafo G.
Puesto que el objetivo final es el de tener varios subconjuntos de nodos, aparte de encontrar el mínimo conjunto dominante conexo, el problema se amplia a encontrar conjuntos dominantes conexos disjuntos de cardinal menor que un número k que permita mantener la eficiencia energética.
A estos conjuntos dominantes se les denomina esqueletos virtuales o columna vertebral virtual de la red. En la Figura 1.1 se ilustran dos conjuntos dominantes conexos disjuntos en una red de sensores y que denotaremos como backbones.
Todos estos conjuntos tienen un nodo común que sería el nodo de salida de la red y que en la imagen

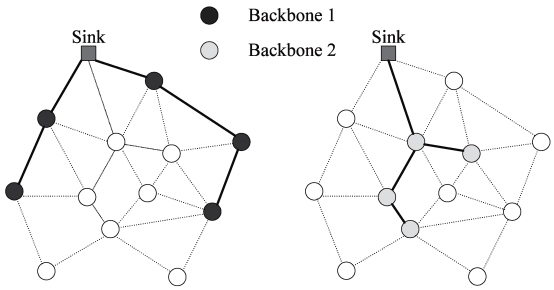


Figura 1.1: Conjuntos dominantes conexos en una red de sensores

se denota como sink. Además, al ser disjuntos aseguramos que los nodos que están activos durante el primer periodo de tiempo, no lo estarán en el siguiente periodo.
Por ejemplo, si al inicio de la red los nodos que forman la backbone1 están activos, será por ellos por los que pasen los datos para transmitirlos al resto de nodos de la red. Al cabo de un cierto periodo de tiempo, los nodos de la backbone2 se activan, cambiandose por los de la backbone1, y por ellos pasará el tráfico de datos de la red. De esta forma, se intenta conseguir la mejor eficencia energética de la red con estos intercambios entre los diferentes conjuntos dominantes conexos obtenidos.

Sin embargo, en la práctica, como el problema del conjunto dominante conexo es NP-completo, no existe un algoritmo óptimo que garantice que la conexión entre los nodos sea la óptima o que todos los nodos estén conectados.

1.2.2. Otras aplicaciones

- Dentro del ámbito tecnológico encontramos más aplicaciones en el caso en el que los nodos que forman el grafo están fijos, al contrario de lo que ocurría con las redes ad-hoc. El ejemplo más característico es el de intentar colocar el mínimo número de cámaras para vigilar el mayor número de objetos posibles dentro de una zona determinada, como por ejemplo en una sala de un museo o dentro de una propiedad privada.

- A pesar de que no existe un algoritmo eficiente que resuelva el problema del conjunto dominante conexo, este se ha usado para el desarrollo de algoritmos tratables de parámetro fijo, más conocidos como algoritmos FPT. Estos algoritmos son usados para resolver problemas NP-completos en los que un parámetro de entrada del problema está fijado. Por ejemplo, varios problemas NP-Duros pueden ser resueltos en tiempo polinómico para grafos con máximo número de hojas acotados.

1.3. Formulación alternativa del problema

En esta sección introduciremos el problema del cálculo de un árbol de expansión con máximo número de hojas.

Definición 1.7. Un vértice de un árbol en un grafo conexo G se llama hoja si tiene grado 1.

En este contexto, definimos el problema del cálculo de un árbol de expansión con número máximo de hojas que trata de encontrar un árbol de expansión en un grafo G cuyo número de hojas sea el máximo posible. Este problema es equivalente al problema de encontrar un conjunto dominante conexo de cardinal mínimo. Recordar que los vértices de un conjunto dominante conexo formaban un subgrafo conexo en el que el resto de vértices del grafo eran adyacentes a los vértices que forman el conjunto dominante conexo. Por tanto, dado un conjunto dominante conexo óptimo D, un árbol de expansión de

D y los nodos restantes del grafo, cada uno de estos conectados únicamente a un nodo de D, forman un árbol de expansión con máximo número de hojas donde las hojas son los nodos restantes ya comentados. Y, recíprocamente, el conjunto de los vértices de grado mayor que uno en un árbol de expansión con número máximo de hojas, forman un conjunto dominante conexo óptimo.
Por ejemplo, a partir del conjunto dominante conexo de la backbone2 visto anteriormente, podemos obtener un árbol con máximo número de hojas:

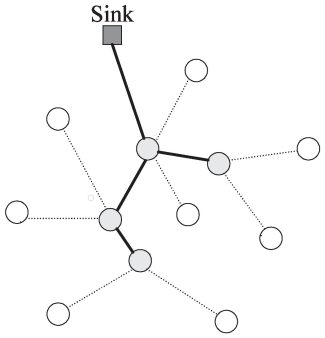


Figura 1.2: Árbol de expansión con máximo número de hojas.

De la misma forma, los nodos de grado mayor que uno de ese árbol forman un conjunto dominante conexo de cardinal mínimo.

Capítulo 2

Complejidad computacional y algoritmos

Este capítulo estará dedicado a demostrar que el problema de hallar el mínimo conjunto dominante conexo de un grafo es NP-completo y a la presentación de distintos algoritmos de aproximación para este problema.

Empezaremos, sin embargo, introduciendo las clases de complejidad P y NP.

2.1. Clases de complejidad P y NP

Las clases de complejidad P y NP son conjuntos donde se clasifican problemas de decisión, es decir, aquellos problemas cuya solución es sí o no. Por ejemplo, en el caso del problema del conjunto dominante conexo, el problema de decisión asociado sería:
Dado un grafo $G = (V,E)$ y un entero no negativo k, ¿Existe un conjunto dominante conexo de tamaño menor o igual que k?
Para definir las clases P y NP más concretamente, antes hay que introducir el término de tiempo polinómico en la resolución de un problema y alguna notación que usaremos.

Definición 2.1. Dada una función $g : \mathbb{N} \longrightarrow \mathbb{N}$, denotamos como $O(g(n))$ al conjunto de funciones:

$$O(g(n)) = \{f(n) | \text{ existe una constante positiva c tal que } 0 \leq f(n) \leq cg(n)\}$$

A este conjunto se le denomina orden de la función g.

Usando esta notación podemos describir el orden de un algoritmo de la siguiente forma:
Dado un algoritmo que resuelve un problema cualquiera, se puede contar el número de operaciones que realiza ese algoritmo hasta llegar a la solución. El total de operaciones del algoritmo es una función que depende del tamaño de los datos de entrada n. Por tanto, diremos que un algoritmo tiene orden $O(g(n))$ si el total de operaciones del algoritmo en su peor caso de ejecución es una función cuyo término de mayor grado es $O(g(n))$. También se dirá que el algoritmo resuelve el problema en tiempo $O(g(n))$.
A continuación definimos el concepto de tiempo polinómico.

Definición 2.2. Diremos que un problema es solucionable en tiempo polinómico si existe un algoritmo que puede resolverlo en tiempo $O(n^k)$ para alguna constante k.

Ahora podemos definir formalmente las clases P y NP.

Definición 2.3. La clase de complejidad P es el conjunto de problemas de decisión para los que se puede obtener una solución en tiempo polinómico.

Definición 2.4. La clase de complejidad NP es la clase de los problemas de decisión en los cuáles las soluciones afirmativas pueden ser verificadas en tiempo polinómico.

Veamos un ejemplo que clarifique esta definición.
Consideremos el problema de decisión asociado al problema de encontrar el conjunto dominante conexo óptimo de un grafo. Tenemos un grafo G y un entero k. Supongamos además que se nos proporciona la información de que existe un conjunto dominante conexo de tamaño k y los vértices que lo forman. Entonces, es claro que se puede verificar en tiempo polinómico que ese conjunto es una solución al problema.
Por tanto, diríamos que este problema pertenece a la clase NP.
Una vez definidas estas dos clases de complejidad es lógico preguntarse si existe alguna relación entre estas las clases P y NP. Veamos que P $\subseteq$ NP.
Esta relación es trivial pues si un problema de decisión está en la clase P, existe un algoritmo que lo soluciona en tiempo polinómico, y por tanto, también se podrá verificar una solución en tiempo polinómico. Por tanto, también pertenece a la clase NP.
Sin embargo, no se ha podido probar que P$\neq$NP. Esta relación sigue siendo una incógnita pero una razón para creer que es así es la existencia de la clase de los problemas NP-completos.

Definición 2.5. Definimos la clase de complejidad NP-completo, denotada como NPC, como el conjunto de problemas de decisión tal que un problema Q pertenece a NPC si:

- $Q \in$ NP
- Todo problema de NP es reducible a Q en tiempo polinómico.

El concepto de reducción de un problema lo definimos a continuación.

Definición 2.6. Dados dos problemas Q y Q' y una serie de datos de entrada x para Q, decimos que el problema Q es reducible al problema Q' en tiempo polinómico, si los datos iniciales de Q se pueden reformular, mediante una transformación en tiempo polinómico, para que sean los datos de entrada x' del problema Q' y además la solución de Q' con estos datos x' proporciona una solución al problema Q con sus datos de entrada iniciales.

Esta será la herramienta que utilizaremos para probar que el problema de encontrar el mínimo conjunto dominante conexo de un grafo es NP-completo. A partir de un problema que sepamos que es NP-completo, podemos usar la reducción en tiempo polinómico para probar que nuestro problema es NP-completo.
Por último, para finalizar la sección definimos la clase de complejidad NP-Duro.

Definición 2.7. La clase de complejidad NP-Duro es el conjunto de problemas de decisión que satisfacen la propiedad 2 de la definición de NP-completo.

2.2. Complejidad computacional del problema del mínimo conjunto dominante conexo

En esta sección incluiremos la demostración de que el problema de hallar el mínimo conjunto dominante conexo de un grafo es NP-Completo. Para ello, usaremos la técnica de reducir el problema a uno conocido que es NP-Completo. Más concretamente, al problema del HPV (Hamiltonian Path Variant) que definimos a continuación.

Definición 2.8. Dado un grafo conexo y plano $G = (V,E)$ y dos vertices $u,v \in V$, definimos un camino hamiltoniano como un camino simple de u hasta v que contiene a todos los vértices de V.
En este contexto definimos el problema HPV:
Dado un grafo $G = (V,E)$ conexo con dos vértices $u,v \in V$ de grado uno y el resto de vértices de grado 3, el problema de decisión asociado al HPV trata de verificar si existe un camino hamiltoniano entre u y v en G.

A continuación, se incluyen los resultados en los que nos apoyaremos para probar que el problema de hallar el conjunto dominante conexo de un grafo es NP-Completo. Previamente, se introduce la notación que se usará en dichos resultados.

Definición 2.9. Si $S \subseteq \mathbb{Z}^+$, un $S-$árbol T es un árbol en el cuál $\text{gr}(x) \in S$ para todo nodo $x \in T$.

Teorema 2.1. *Dado un grafo plano cualquiera $G = (V,E)$ tal que $gr(v) \leq 3\ \forall v \in V$, el problema de decisión sobre la existencia de un árbol de expansión T para G cumpliendo que $gr(v,T) = 1$ ó $gr(v,T) = 3\ \forall v \in V$ es NP-Completo.*

Demostración. Denotemos como $P^{1,3}$ el problema anterior. Es claro que $P^{1,3} \in$ NP pues, dado un conjunto de vértices de G se puede verificar que forman un $\{1,3\}$-árbol de expansión en tiempo polinómico. Ahora, reducimos este problema al problema HPV.
Sea $G = (V,E)$ un grafo plano y conexo cualquiera con dos nodos $u, v \in V$ de grado 1 y tal que $\text{gr}(w) = 3$ $\forall w \in V, w \neq u, v$. Construimos un grafo $G' = (V', E')$, reemplazando cada nodo de grado 3 de G de la siguiente forma:

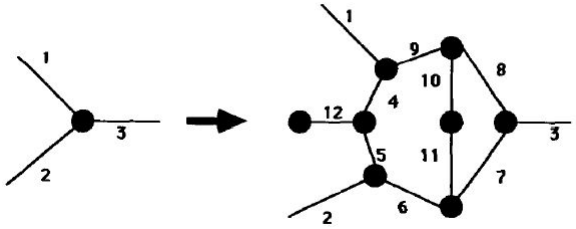


Figura 2.1: Transformación de los nodos de grado 3

Así, obtenemos un grafo G' cuyos nodos tienen como máximo grado 3. El grafo que reemplaza a un nodo de grado 3, lo llamaremos expansión del nodo.
Esta construcción se puede llevar a cabo en tiempo polinómico, puesto que para cada nodo de grado 3, se obtiene un subgrafo con 8 vértices y 12 aristas. Luego, como la construcción es la misma para cada nodo de grado 3, el grafo G' tendrá $O(8k)$ vértices y $O(12k)$ aristas donde k es el número de nodos de grado 3 del grafo original. Por tanto, el grafo G' se puede construir en tiempo polinómico.
Veamos entonces que existe un camino hamiltoniano P entre u y v en G si y solo si existe un $\{1,3\}$-árbol de expansión T para G'.
$\Rightarrow$) : Supongamos que existe un camino hamiltoniano P entre u y v. Si P usa las aristas 1 y 2 de la Figura 2.1 (o las aristas 1 y 3 respectivamente) entonces supongamos que T usa las aristas en negrita de la Figura 2.2.

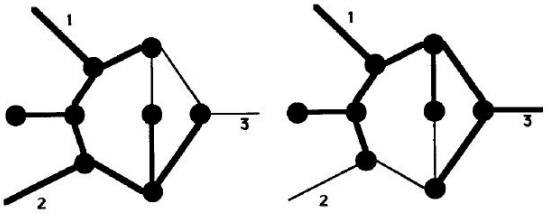


Figura 2.2: Las aristas en negrita de la figura de la izquierda representan las aristas que usa T si el camino P usa las aristas 1 y 2. La figura de la derecha las aristas que usa T si P usa 1 y 3.

Por tanto, si P usa las aristas 1 y 2 (respectivamente la 1 y la 3) de G entonces las aristas en negrita de la expansión del nodo pertenecen a un $\{1,3\}$-árbol de expansión T para G'.
El caso donde P usa las aristas 2 y 3 es simétrico al caso en el que use las aristas 1 y 3. Entonces, obtenemos un $\{1,3\}$-árbol de expansión T para G' a partir del camino P de G.

($\Leftarrow$: Supongamos ahora que existe un $\{1,3\}$-árbol de expansión T para G'. Veamos cuales son las aristas que puede usar T en la expansión del nodo de la Figura 2.1.
T tiene que usar la arista 12 y, para cumplir la restricción de grados, también tiene que usar las aristas 4 y 5. Además, T no puede usar las aristas 1, 2 y 3 a la vez pues en ese caso, T debería usar todas las aristas 4, 5, ... , 9, lo cuál formaría un ciclo que contradice que T sea un árbol de expansión.
Tampoco puede usar solo una de estas tres aristas, pues en ese caso no se cumpliría la restricción de los grados de los nodos que forman T. En efecto, si por ejemplo T usa la arista 1 y no usa la 2 ni la 3, entonces T usaría las aristas 4, 5, 6, 7, 8, 9, 10 y 12, pero en este caso habría un nodo que tendría grado 2.
Por tanto, T tiene que usar exactamente dos aristas del conjunto $\{1,2,3\}$. Además T usa una de las aristas 10 y 11 pero no puede usar ambas a la vez por la restricción en los grados. Si T usa la arista 10 entonces para satisfacer la restricción de los grados, también usará la arista 1 y, o bien la 2 o bien la 3. En el caso en el que use la arista 11 entonces T usará la arista 2 y una de las aristas 1 ó 3.
Notar que en cualquiera de estas posibilidades el árbol T forma un grafo conexo de 8 nodos a partir de la expansión del nodo de la Figura 2.1.
Veámoslo para el caso en el que T usa las aristas 1,2 y 10 ya que para el resto de posibilidades se cumplirá lo mismo. Por las restricciones en los grados, y por las que hemos visto anteriormente, a partir de la Figura 2.1 obtenemos el siguiente grafo conexo de 8 nodos:

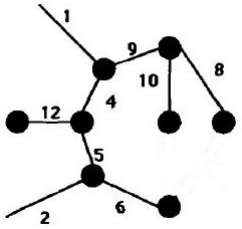


Figura 2.3: Grafo conexo de 8 nodos cumpliendo las restricciones de grados en sus nodos.

Para el resto de casos, el procedimiento es análogo y por tanto, también se obtiene un grafo conexo de 8 nodos.
Por tanto, podemos realizar la transformación inversa a la hecha en la Figura 2.1 y colapsar la figura 2.3 a un nodo de grado 3, de forma que T se convierte en un subgrafo conexo P de G en el cúal todo nodo de grado 3 tiene exactamente dos aristas en P (dependiendo de las aristas del conjunto $\{1,2,3\}$ que use T en cada expansión del nodo) y en el cuál los nodos u y v tienen grado 1 en P. Luego, por el problema HPV, P es un camino hamiltoniano entre u y v en G.
Y, como el problema HPV es NP-Completo, queda demostrado el Teorema. □

Corolario 2.2. *Sea G un grafo plano con n nodos y cuyos nodos tienen un grado máximo de 3. Entonces, es NP-Completo decidir si existe un árbol de expansión para G con al menos $\frac{n}{2}+1$ hojas. Además, cualquiera de esos árboles de expansión debe tener exactamente $\frac{n}{2}+1$ hojas, de hecho, tiene que ser un $\{1,3\}$-árbol de expansión.*

Demostración. Denotaremos $K = \frac{n}{2}+1$. Sea T un árbol de expansión de G, veamos que las siguientes afirmaciones son equivalentes:

i) T es un $\{1,3\}$-árbol.

ii) T tiene exactamente K hojas.

iii) T tiene al menos K hojas.

Obviamente, ii) $\Rightarrow$ iii).
Para demostrar el resto, sea n_i el número de nodos x de T tal que $\text{gr}(x,T) = i$, para $i = 1,2,3$.
i) $\Rightarrow$ ii). Para T se cumple que $n_1 + n_3 = n$ y $3n_3 + n_1 = 2(n-1)$. Entonces, resolviendo este sistema de dos ecuaciones, se llega a que $n_1 = \frac{n}{2} + 1$ o equivalentemente, T tiene $\frac{n}{2} + 1 = K$ hojas.
iii) $\Rightarrow$ i). En este caso, se cumple que $n_1 + n_2 + n_3 = n$, $3n_3 + 2n_2 + n_1 = 2(n-1)$ y $n_1 \geq k$. Entonces, resolviendo este sistema de ecuaciones imponiendo que $n_1 \geq k$ se llega a que $n_2 = 0$ o, lo que es lo mismo, que no hay nodos de grado 2 y por tanto, T es un $\{1,3\}$-árbol.

Por tanto, se sigue del Teorema anterior que el problema de decisión del enunciado es NP-Completo. □

Corolario 2.3. *Sea G un grafo plano con n nodos y cuyos nodos tienen un grado máximo de 3. Es NP-Completo decidir si existe un conjunto dominante conexo D en G con cardinalidad no mayor que $\frac{n}{2} - 1$. Además, cualquier conjunto D cumpliendo que $|D| \leq \frac{n}{2} - 1$ debe tener cardinalidad igual a $\frac{n}{2} - 1$.*

Demostración. Denotaremos $K = \frac{n}{2} - 1$. Probaremos el corolario reduciendo este problema al problema del corolario anterior que sabemos que es NP-Completo. Por tanto, decimos que existe un árbol de expansión T para G con al menos $n - K$ hojas si y solo si existe un conjunto dominante conexo D en G tal que $|D| \leq K$. Además, cualquier conjunto D de esa forma debe cumplir que $|D| = K$.
$\Rightarrow$) Si T es un árbol con al menos $n - K$ hojas, entonces los nodos de T que no son hojas forman un conjunto dominante conexo D que tendrá cardinal menor o igual que $n - (n - K) = K$.
($\Leftarrow$ Sea T' un árbol de expansión para el subgrafo inducido por D. Entonces los nodos de G que no están en D cada uno de ellos conectados únicamente a un nodo de D serán las hojas que se añaden al árbol T' para formar un árbol de expansión T para G.

Falta ver que $|D| = K$. Pero esto se sigue del corolario anterior, puesto que si un árbol tiene al menos $n - K = \frac{n}{2} + 1$ hojas entonces el árbol tendrá exactamente $\frac{n}{2} + 1$ hojas. Por tanto, D tiene que tener cardinalidad $n - \frac{n}{2} - 1 = \frac{n}{2} - 1 = K$.
Luego la doble equivalencia queda probada y que el problema es NP-Completo se sigue ahora del corolario anterior. □

Una vez probados estos tres resultados, ya podemos razonar que el problema de decisión asociado al problema de hallar el mínimo conjunto dominante conexo de un grafo G es NP-Completo.

Corolario 2.4. *Dado un grafo plano $G = (V,E)$ es NP-Completo decidir si existe un conjunto dominante conexo de cardinal mínimo en G.*

Demostración. Es consecuencia directa del corolario anterior ya que si el problema es NP-Completo para un grafo con restricciones en sus nodos entonces también será NP-Completo decidir si existe un conjunto dominante conexo para un grafo en el que los grados de los nodos pueden ser cualesquiera. Además, si es NP-Completo cuando la cardinalidad del conjunto no puede superar $\frac{n}{2} - 1$ también lo será si el cardinal es el mínimo posible. □

Por tanto, el problema de hallar el mínimo conjunto dominante conexo de un grafo G es NP-Completo.

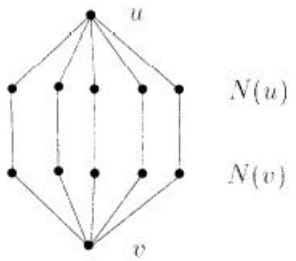
2.3. Algoritmos de aproximación

En esta sección presentaremos y analizaremos diferentes algoritmos para intentar hallar un conjunto dominante conexo óptimo de un grafo. Como hemos visto que el problema de encontrar el conjunto dominante conexo óptimo de un grafo es NP-Completo, la existencia de un algoritmo óptimo para este problema es poco probable que exista. Por tanto, los algoritmos que vamos a analizar producirán una solución lo más próxima posible a la óptima.

2.3.1. Algoritmo para la construcción de un conjunto dominante conexo

Descripción y análisis del algoritmo

Primero, introduciremos la notación que vamos a usar:
Dado un grafo $G = (V,E)$ denotaremos el grado máximo de un vértice del grafo como Δ. Usaremos n y m para denotar el número de vértices y de aristas del grafo, respectivamente. Por último, denotamos $N(u)$ al conjunto de vecinos del vértice u.
Este algoritmo está basado en encontrar un conjunto dominante conexo de un grafo G mediante la construcción de un árbol T en el grafo G. Se empezará por el vértice de G de grado máximo el cuál se añade a T. A continuación , en cada paso cogemos un vértice u de T y lo "escaneamos". Escanear un vértice trata de añadir las aristas que unen el vértice u con sus vecinos a T. Así se procede hasta que se obtiene un árbol de expansión T en G del cuál los vértices que no son hojas formarán el conjunto dominante conexo.
Para entender mejor el funcionamiento del algoritmo, usaremos un proceso de marcado con colores para cada uno de los vértices. Partimos con todos los vértices sin marcar, es decir, todos de color blanco. Entonces, cada vez que escaneamos un vértice, lo coloreamos de negro y marcamos a sus vecinos de color gris. Estos vértices de color gris se añaden a T y son los diferentes candidatos a ser escaneados en el siguiente paso del algoritmo. De esta forma el algoritmo continua escaneando vértices hasta que todos los vértices estén marcados, o bien en negro o bien en gris. Entonces, los nodos coloreados en negro forman el conjunto dominante conexo.
Queda por aclarar cuál es el criterio de elección a la hora de elegir el vértice gris que se va a escanear en el paso siguiente. A priori, se podría pensar que el mejor vértice a elegir es aquel que tiene mayor número de vecinos sin marcar. Sin embargo, veremos posteriormente con un ejemplo que hay una opción mejor. Esta opción se basa en escanear por pares de vértices adyacentes. Esto quiere decir que para un par de vértices u,v marcamos u de negro y por tanto, v se marcará de gris. A continuación coloreamos v de negro, lo cuál generará más nodos marcados en gris. El número total de vértices coloreados en gris con este paso, se le denominará rendimiento del paso.
A continuación veamos por qué esta opción es mejor. Consideramos el siguiente grafo:



Si usamos la primera regla, empezamos marcando y escaneando u. Por tanto, marcamos y añadimos todos sus vecinos al árbol T. Ahora cogemos uno de los vértices de $N(u)$ y lo escaneamos añadiendo su vecino a T. Reiteramos este proceso hasta que hayamos escaneado todos los vértices de $N(u)$. Por últi-

mo, escaneamos uno de los vértices de $N(v)$ y marcamos v. De esta forma, hemos necesitado escanear un total de 7 vértices hasta llegar a la solución.
Ahora, veamos como mejora si escaneamos los vértices por pares. Marcamos u de negro y elegimos un vértice w de $N(u)$ que estará marcado de gris por ser vecino de u. A continuación, marcamos de negro al vértice w y marcamos su único vecino de $N(v)$. Repetimos este paso con este vértice y llegamos a v. De esta forma, se obtiene un conjunto dominante conexo óptimo de 4 vértices.
Con este último proceso de escaneo podemos demostrar el siguiente teorema.

Teorema 2.5. *Usando el proceso de escaneo de vértices por pares, se obtiene un conjunto dominante conexo de tamaño como máximo* $2(1+H(\Delta))|OPT_{CD}|$*, donde* $H(n)$ *es el n-ésimo número armónico y* OPT_{CD} *es un conjunto dominante óptimo del grafo.*

Demostración. Sea OPT_{CD} un conjunto de vértices que forman un conjunto dominante óptimo en el grafo. Llamamos S_i al conjunto de vértices dominados por el vértice $i \in OPT_{CD}$. En el caso en el que un vértice este dominado por más de un vértice, se añadirá aleatoriamente a uno de esos conjuntos.
Esta demostración va a estar basada en la asignación de costes a cada uno de los vértices que marcamos en cada paso. Notar que cada vértice se marca una única vez y por tanto, se le asociará únicamente un coste. Probaremos ahora que la suma total de los costes de los vértices en S_i es como máximo $2(1+H(\Delta))$. Y como el número de conjuntos S_i en la solución es $|OPT_{CD}|$ se sigue el teorema.
La asignación de los costes se realiza de la siguiente forma. Cada vez que coloreamos un vértice de color negro, marcamos un número nuevo x de vértices de color gris. A cada nuevo vértice marcado se le asocia el coste $\frac{1}{x}$, y en el caso en el que escaneemos un par de vértices, a cada nuevo vértice se le asocia el coste $\frac{2}{x}$. Así, nos aseguramos que al sumar todos los costes obtendremos el tamaño del conjunto dominante conexo formado.
Probemos que efectivamente la suma de los costes es como máximo $2(1+H(\Delta))$ en cada S_i.
Sea u_0 el número de vértices que no están marcados inicialmente en S_i y sea u_j el número de vértices que no están marcados en S_i después del paso j.
Por simplicidad, consideramos que en cada paso marcamos nuevos vértices de S_i de forma que en cada paso el número de vértices no marcados decrece.
El número de vértices marcados en el primer paso es $u_0 - u_1$ y a cada vértice se le asocia un coste de $\frac{2}{u_0-u_1}$. Notar además que una vez que un vértice de S_i es marcado, entonces el vértice i puede ser elegido para ser escaneado como parte de un par por ser adyacente a todos los vértices de S_i.
En el paso j-ésimo, el número de vértices marcados es $u_j - u_{j+1}$ y el coste de cada vértice de S_i será como máximo $\frac{2}{u_j}$, pues si el vértice i es elegido en el paso j-ésimo, se tendrá que $u_{j+1} = 0$.
Ahora, sea u_k el número de vértices sin marcar en el último paso. Se tiene entonces que $u_k = 0$. Por tanto, sumando todos los costes se llega a que:

$$\frac{2}{u_0-u_1}(u_0-u_1) + \sum_{j=1}^{k-1} \frac{2}{u_j}(u_j - u_{j+1}) \leq 2 + 2\sum_{j=1}^{k-1} \frac{u_j - u_{j+1}}{u_j}$$

Ahora, por definición del número armónico:

$$H(n) = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}$$

Por tanto, para dos enteros a y b tales que $a \leq b$ se tiene que:

$$H(b) - H(a) = \sum_{i=a+1}^{b} \frac{1}{i} \geq (b-a)\frac{1}{b}$$

Luego, aplicando esta desigualdad a la suma de costes, obtenemos una suma telescópica:

$$2 + 2\sum_{j=1}^{k-1} \frac{u_j - u_{j+1}}{u_j} \leq 2 + 2\sum_{j=1}^{k-1} (H(u_j) - H(u_{j+1})) = 2 + 2(H(u_1) - H(u_k))$$

Y usando que $u_0 \leq \Delta + 1$ y que $u_k = 0$, se tiene que:

$$2 + 2(H(u_0 - (u_0 - u_1))) \leq 2 + 2(H(\Delta + 1 - (u_0 - u_1))$$

Y como hemos supuesto que en cada paso marcamos al menos un vertice se tiene que $\Delta + 1 - (u_0 - u_1) \leq \Delta$ y por tanto:

$$2 + 2(H(\Delta + 1 - (u_0 - u_1)) \leq 2 + 2H(\Delta) = 2(1 + H(\Delta))$$

Y de esta desigualdad se sigue el teorema. □

Coste computacional

Analizemos ahora el coste computacional del algoritmo.
En cada iteración del algoritmo, el objetivo es escanear un par de vértices que nos den el máximo rendimiento, es decir, que marquen el mayor número de vértices posibles. Por ello, en cada paso, consideramos un vértice u gris (marcado) y todos sus vértices adyacentes, y para cada vecino v de u blanco (sin marcar) examinamos el número de vértices que se marcarían si se escanease el par (u, v).
Hay que tener en cuenta que podría ocurrir que un vértice fuese a la vez vecino de u y de v. Por tanto, tenemos que contar todos los vértices blancos vecinos de v pero que no lo son de u.
Entonces, para cada vértice tenemos que hacer dos listas de adyacencia, una de los vecinos grises del vértice y otra de los vecinos blancos. Denotemos por GR al conjunto de vértices grises de T, B al conjunto de los vértices blancos que son adyacentes a los grises y $d_B(u)$, $d_{GR}(u)$ al número de vecinos blancos y al número de vecinos grises de un vértice u respectivamente.
Entonces, en cada iteración hay que checkear el número de vecinos blancos de cada vértice v tal que $v \in B$ y v es vecino de u. Y este checkeo hay que repetirlo para cada vértice gris u que hubiese. Por tanto, el número de comprobaciones que hay que hacer se puede poner como:

$$\begin{aligned} \text{N}^{\text{o}} \text{ comprobaciones} &= \sum_{u \in GR} \sum_{v \in B \wedge v \in N(u)} d_B(v) \\ &= \sum_{v \in B} d_B(v) \cdot d_{GR}(v) \end{aligned}$$

Tras este proceso, se marca un cierto número de vértices blancos. Es fácil ver que se cumple el siguiente lema:

Lema 2.6. *El número de vértices blancos que se marcan en cada iteración es al menos:*

$$\max_{v \in B} d_B(v)$$

Demostración. Basta con tomar el par (u, v) de vértices que tienen el mayor rendimiento. Entonces, al escanear este par, el número de vértices marcados serán todos los vecinos blancos de v, y ninguno tiene más vecinos blancos que este por haber tomado el par de vértices con mayor rendimiento. □

Ahora, podemos asociar un coste a cada uno de los vértices que se han marcado de color gris. El coste de cada uno de esos vértices es como mucho:

$$\frac{\sum_{v \in B} d_B(v) \cdot d_{GR}(v)}{\max_{v \in B} d_B(v)}$$

Notar que este esquema de costes es igual al del teorema probado anteriormente. Es el cociente entre el número de vértices que se escanean y el nuevo número de vértices marcados. Notar que el peor caso del algoritmo es en el que cada vértice v solo tiene únicamente un vecino blanco y por tanto, podemos acotar el coste anterior por la suma del número de vértices grises que son vecinos de cada vértice v de esta forma:

$$\frac{\sum_{v \in B} d_B(v) \cdot d_{GR}(v)}{\max_{v \in B} d_B(v)} \leq \sum_{v \in B} d_{GR}(v) \leq 2m$$

Además, cada vértice es marcado solo una vez en todo el algoritmo, y como se tienen un total de n vértices, el número total de pasos del algoritmo en su peor tiempo de ejecución es $O(mn)$.

2.3.2. Mejora del algoritmo anterior

Descripción y análisis del algoritmo

En este segundo algoritmo se presenta una forma alternativa de construir un árbol T en G. La notación y terminología será la misma que la usada en el algoritmo anterior. La idea principal en la que se basa el algoritmo es la construcción de componentes separadas que forman un conjunto dominante en G y a continuación conectarlas. Por tanto, el algoritmo tendrá dos fases.
En la primera fase, partimos de un grafo G en la que todos sus vértices son de color blanco. Cada vez que se añada un vértice al conjunto dominante, este se coloreará de negro y los vértices dominados por el vértice negro se marcarán en gris. En cada paso de esta primera fase, el algoritmo escoge un único vértice, lo colorea de negro y marca sus correspondientes vértices adyacentes de gris.
Denotaremos como "pieza" a un vértice blanco o a una componente conexa formada por vértices negros. Por tanto, el vértice que se coloree de negro en cada paso del algoritmo será aquel que reduzca el máximo número de piezas.
Procediendo de esta manera, en el último paso de esta fase, ningún vértice podrá reducir el número de piezas del grafo, y probaremos que en ese caso, no quedará ningún vértice blanco.
En la segunda fase tendremos una colección de componentes conexas negras que forman un conjunto dominante en G. Para obtener el conjunto dominante conexo deseado únicamente queda conectarlas. Para ello, se partirá de un vértice gris adyacente a una componente negra y recursivamente, se conectará un vértice gris con otro del mismo color que sea adyacente hasta llegar a uno que tenga como vecino a un vértice negro de otra componente conexa negra. De esta forma se habrá generado una cadena de vértices que une dos componentes negras. Se probará que esta cadena en realidad está formada por únicamente dos vértices.
Al final del algoritmo llegamos a un árbol T cuyas hojas serán los vértices grises sobrantes. Por tanto, los vértices negros de ese árbol forman un conjunto dominante conexo que es la solución que se buscaba.

Lema 2.7. *Al final de la primera fase no queda ningún vértice blanco.*

Demostración. Supongamos que hay un vértice blanco u al final de la primera fase. Entonces, probemos que en ese caso existe un vértice que permite reducir el número de piezas en ese paso.
Notar que todos los vecinos de u tienen que ser blancos o grises, pues sino u sería gris. Veamos que u no puede tener ni vecinos blancos ni grises.
Si u tuviese un vecino blanco, entonces, coloreando a u de negro reduce en dos el número de vértices blancos e incrementa en uno el número de componentes negras. Luego, en este caso se reduciría el número de piezas. Contradicción con que haya terminado la fase uno. En consecuencia, u tiene que tener un vecino gris v.
Pero, en este caso, si se colorea v de negro, se marcaría u de gris y, por tanto, se reduciría el número de vértices blancos. Sin embargo, no aumentaría el número de componentes negras pues v, al ser gris, es adyacente a una componente negra. Por tanto, añadiendo v a la componente, se reduce también el número de piezas. Igual que antes se llega a contradicción pues en ese caso la fase uno no habría terminado.
Luego no puede haber ningún vértice blanco al final de la primera fase. □

Lema 2.8. *Si al final de la primera fase hay más de una componente conexa negra, entonces siempre existe una pareja de componentes conexas negras que pueden ser conectadas por una cadena de dos vértices.*

Demostración. Consideremos el camino más corto que conecta esas dos componentes y supongamos que esta formado por la sucesión de vértices $u_0, u_1, u_2, u_3, ..., u_k$ donde u_0 y u_k pertenecen a las componentes negras i y j respectivamente. Es claro que el vértice u_1 está dominado por u_0. Supongamos ahora que u_2 fuese negro, entonces haciendo u_1 negro reduciríamos en uno el número de piezas, lo cuál es una contradicción pues se supone terminada la fase uno. u_2 tampoco puede ser blanco por el lema anterior,

luego necesariamente, u_2 tiene que ser gris.
Por tanto, será adyacente a una componente negra l distinta de i. Entonces, las componentes i y l pueden conectarse mediante la cadena una cadena de dos vértices grises formada por u_1 y u_2. □

Por tanto, hemos probado que el algoritmo funciona correctamente y al finalizar obtenemos un conjunto dominante conexo. El siguiente teorema indica una cota del tamaño del conjunto dominante conexo obtenido tras aplicar el algoritmo.

Teorema 2.9. *El conjunto dominante conexo obtenido por el algoritmo tiene un tamaño máximo de* $(ln\Delta+3)\cdot|OPT_{CDS}|$ *donde* OPT_{CDS} *es el conjunto dominante conexo óptimo del grafo.*

Demostración. Denotemos por a_i el número de piezas restantes tras el paso i del algoritmo y $a_0 = n$. Cada vértice del grafo puede dominar como mucho Δ vértices por lo que podrá conectar como mucho Δ piezas, entonces se cumple que $|OPT_{CDS}| \geq \frac{a_0}{\Delta}$. Consideremos ahora el paso $i+1$. Como la solución óptima en este paso podría conectar las a_i piezas restantes, la forma de conseguir conectar el mayor número de piezas en este paso sería elegir un vértice que conecte al menos $\lceil \frac{a_i}{|OPT_{CDS}|} \rceil$. Este proceso, reduciría el número de piezas en al menos $\lceil \frac{a_i}{|OPT_{CDS}|} \rceil - 1$ pues sumamos una componente negra. Así, obtenemos la siguiente relación de recurrencia entre las piezas restantes de dos pasos consecutivos del algoritmo:

$$a_{i+1} \leq a_i - \lceil \frac{a_i}{|OPT_{CDS}|} \rceil + 1 \leq a_i \left(1 - \frac{1}{|OPT_{CDS}|}\right) + 1$$

De donde obtenemos:

$$a_i \leq a_0 \left(1 - \frac{1}{|OPT_{CDS}|}\right)^i + \sum_{j=0}^{i-1} \left(1 - \frac{1}{|OPT_{CDS}|}\right)^j =$$

$$= a_0 \left(1 - \frac{1}{|OPT_{CDS}|}\right)^i + |OPT_{CDS}| \left(1 - \left(1 - \frac{1}{|OPT_{CDS}|}\right)^i\right) =$$

$$= (a_0 - |OPT_{CDS}|) \left(1 - \frac{1}{|OPT_{CDS}|}\right)^i + |OPT_{CDS}|$$

Estamos usando un procedimiento en el que se elige un vértice que reduce el número de piezas en al menos $\lceil \frac{a_i}{|OPT_{CDS}|} \rceil - 1$ unidades. Mientras el número de piezas restantes en un paso sea mayor que $2|OPT_{CDS}|$, el número de piezas en el paso siguiente se podrá reducir en al menos dos unidades. Por tanto, el algoritmo reduce el número de piezas en más de dos unidades hasta que se cumple que:

$$2|OPT_{CDS}| \leq (a_0 - |OPT_{CDS}|) \left(1 - \frac{1}{|OPT_{CDS}|}\right)^i + |OPT_{CDS}| \leq a_0 e^{\left(\frac{-i}{|OPT_{CDS}|}\right)} + |OPT_{CDS}|$$

Y esta relación se satisface si:

$$i \leq |OPT_{CDS}| \cdot \ln\left(\frac{a_0}{|OPT_{CDS}|}\right)$$

Por tanto, tras $|OPT_{CDS}| \cdot \ln\left(\frac{a_0}{|OPT_{CDS}|}\right)$ pasos, el número de piezas restantes es menor que $2|OPT_{CDS}|$. Tras esta iteración, por cada vértice que tomemos, el número de piezas decrece en al menos una hasta que el número de componentes negras sea como máximo $|OPT_{CDS}|$. Por tanto, como mucho en este proceso tomaremos $|OPT_{CDS}|$ vértices.
Así, tras $|OPT_{CDS}| \cdot \ln\left(\frac{a_0}{|OPT_{CDS}|}\right) + |OPT_{CDS}|$ iteraciones, quedarán como mucho $|OPT_{CDS}|$ piezas por conectar.
Supongamos que a partir de aquí, paramos después de elegir a_f vértices más. Entonces, las piezas restantes que queden por conectar serán como mucho $|OPT_{CDS}| - a_f$. A continuación, con los vértices

restantes que no pertenecen a las componentes negras, formamos cadenas de dos vértices que conecten las piezas restantes.
Por tanto, el total de vértices escogidos en todo este proceso, y por tanto, los que formarán el conjunto dominante conexo final son como mucho

$$|OPT_{CDS}| \cdot \ln\left(\frac{a_0}{|OPT_{CDS}|}\right) + |OPT_{CDS}| + a_f + 2(|OPT_{CDS}| - a_f)$$

Y como teníamos que $\Delta \geq \frac{a_0}{|OPT_{CDS}|}$ y sacando factor común, llegamos a que el conjunto dominante conexo final tiene una cardinalidad máxima de $|OPT_{CDS}| \cdot (\ln\Delta + 3)$. □

Existe otra demostración alternativa que mejora levemente el factor de aproximación al cardinal del conjunto dominante conexo óptimo, obteniendo una solución que tiene tamaño máximo $|OPT_{CDS}| \cdot (H(\Delta) + 2)$.

Coste computacional

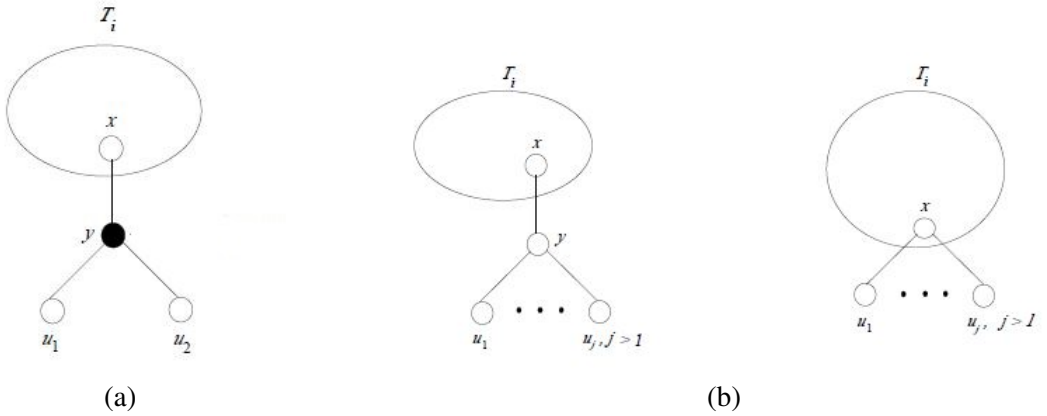
El coste computacional del algoritmo es del orden $O(n^2)$ puesto que en cada iteración del algoritmo este examina los vecinos de cada vértice para ver cuál es el que más reducción de piezas proporciona. Y como esto lo realiza para cada uno de los vértices del grafo nos queda que el tiempo de ejecución del algoritmo en su peor caso es $O(n^2)$.

2.3.3. Algoritmo para la construcción de un árbol de expansión con máximo número de hojas

Descripción del algoritmo

Hemos visto en el capítulo anterior la equivalencia entre el problema de hallar el conjunto dominante conexo óptimo de un grafo y el de encontrar un árbol de expansión con máximo número de hojas. El algoritmo que vamos a presentar obtiene una solución aproximada a este segundo problema en un grafo conexo G. En particular, probaremos que el algoritmo nos permite obtener un árbol de expansión T con al menos un tercio de las hojas que tendría el árbol de expansión con máximo número de hojas y que denotaremos por T^*.
Consideremos $G = (V, E)$ un grafo conexo y denotemos por m y n al número de aristas y vértices respectivamente del grafo G.
El algoritmo comienza construyendo un bosque F siguiendo una serie de restricciones que comentaremos más adelante. A continuación, los árboles que forman el bosque se conectan para forman un árbol de expansión T. Las restricciones que se imponen a la hora de construir F son con el objetivo de conseguir que el mayor número de vértices de F sean hojas, de forma que al construir el árbol T se usen el menor número posible de hojas para conectar los árboles de F. Así, obtendremos un árbol de expansión con el mayor número de hojas posibles.
Veamos ahora como es la construcción de F. Todo árbol $T_i \in F$ se construye tomando como raíz un vértice de grado mínimo 3 y añadiendo sus vecinos al árbol. A continuación, expandiremos sus hojas siguiendo tres esquemas. Estos esquemas tienen una prioridad diferente y siempre que sea posible se usará el que tenga más prioridad que el resto. Los presentaremos en un orden de mayor a menor prioridad.
El primer esquema se usa si x es una hoja que tiene un único vecino $y \notin F$ y a la vez y tiene exactamente dos vecinos que no están en F. De esta forma, el vértice y se añade como hijo de x y nos referiremos a él como vértice negro. A continuación los vecinos de y se añaden al árbol como sus hijos. Este esquema tendrá orden de prioridad 1 y se puede observar en la Figura 2.4.
Los otros dos esquemas tienen ambos orden de prioridad 2 y se usarán indistintamente cuando se pueda.
El segundo esquema se usa si x es una hoja que tiene un solo vecino $y \notin T_i$ e y tiene al menos dos vecinos que no estén en T_i. La forma de añadir estos vértices es análoga a la del esquema uno. Por último, el

tercer esquema se utilizará si x tiene al menos dos vecinos que no están en T_i que se añadirán a T_i como hijos de x. Estos dos esquemas se pueden observar en la Figura 2.4.
Estos esquemas se siguen usando hasta que no quede ninguna hoja de T_i que se pueda expandir mediante ellos en cuyo caso el árbol quedará construido.



(a) (b)

Figura 2.4: (a) Esquema 1. (b) Esquemas 2 y 3.

Análisis del algoritmo

El uso del algoritmo no nos asegura que tras la aplicación de los esquemas anteriores todos los vértices del grafo inicial pertenezcan a F. Por tanto, cabe la posibilidad que al finalizar la expansión de las hojas queden vértices fuera de F. A estos vértices los llamaremos vértices exteriores. Serán estos vértices los que nos permitirán conectar los árboles de F para formar el árbol de expansión T. Veamos que el número de hojas necesitadas para esta conexión es $2k$, con k el número de árboles de F, y que cada hoja de F se conecta únicamente con un vértice exterior.

Lema 2.10. *Sea $G' = (V', E')$ el grafo que se genera al colapsar cada árbol T_i a un único vértice y eliminar las aristas que unen parejas de estos vértices. Entonces todo vértice exterior tiene como mucho grado 2 en G'.*

Demostración. Supongamos que existe un vértice exterior v de grado al menos 3 y consideremos tres de sus vecinos. Notar que los tres vecinos no pueden ser vértices exteriores, pues en ese caso el algoritmo habría elegido v como raíz de un nuevo árbol. Por tanto, al menos un vértice u tiene que estar en F. Supongamos que $u \in T_i$ y que T_i es el primer árbol construido por el algoritmo. Entonces, aplicando el esquema 1 al vértice u podemos añadir v y sus dos vecinos restantes a T_i lo cuál contradice que v sea exterior. Luego v tiene como mucho grado 2. □

Lema 2.11. *Sea u una hoja de un árbol T_i de F. Si u es adyacente a dos vértices $v, w \notin T_i$, entonces v y w son hojas del mismo árbol.*

Demostración. Primero notar que v y w no pueden ser vértices exteriores pues en ese caso podríamos usar el esquema 3 para añadir estos vértices a T_i. Tampoco pueden ser vértices internos de T_j. Consideremos el caso en el que v lo fuese (para w es análogo).
Si el algoritmo construye T_j antes que T_i entonces usando el esquema 3 el algoritmo añadiría u a T_j lo que entra en contradicción con que u y v pertenecen a distintos árboles.
Ahora, si T_i se construye antes que T_j entonces como por la propia construcción del algoritmo todo vértice interno tiene al menos 3 vecinos, podríamos añadir v a T_i expandiendo u mediante el esquema 2. Contradicción.
Por tanto, al menos uno de v y w debe ser una hoja. Supongamos que v es una hoja de T_j y que w no

lo es y veamos que llegamos a contradicción. Podemos suponer sin pérdida de generalidad que cuando el algoritmo construye T_j, w no pertenece todavía a F. Por tanto, T_j se construye también antes que T_i pues sino el algoritmo habría incluido a v y w en T_i. Notar además que por hipótesis, u será una hoja de T_i luego tendrá un vecino $z \in T_i$ distinto de v y w. Por tanto, podríamos aplicar el esquema 2 a u,w y z y añadir u a T_j lo cual contradice que u y v pertenezcan al mismo árbol.
Luego v y w son hojas de T_j. □

Estos resultados nos permiten establecer una cota sobre el número de hojas de un árbol de expansión de G.

Corolario 2.12. *Cualquier árbol de expansión de G tiene como mucho $|V(F)|-2k$ hojas, donde $|V(F)|$ denota el número de vértices de F y k el número de árboles de F.*

Demostración. Es una consecuencia inmediata de los dos lemas anteriores pues por el Lema 2.10 un vértice exterior puede conectar como mucho dos árboles y por el Lema 2.11 cada hoja de un árbol solo puede ser adyacente a un único vértice exterior. Por tanto, en cada conexión de este tipo cada árbol pierde dos hojas. Por tanto, F pierde al menos $2k$ hojas al conectar sus árboles.
Por último, en caso de que un vértice exterior no se usase para la conexión, para formar un árbol de expansión en G habría que conectarlo con una hoja de F. □

Para terminar el análisis de este algoritmo, veamos que el número de hojas obtenidas por este algoritmo son como mínimo un tercio de las que tiene T^*. Previamente, probaremos una acotación del número de hojas de F.
A partir de ahora denotaremos como $B(T)$ al conjunto de vértices negros de un árbol y $l(T)$ el conjunto de hojas de un árbol.

Lema 2.13. *Para cualquier árbol $T_i \in F$ se tiene que* $|l(T_i)| \geq 3 + |B(T_i)| + \dfrac{|V(T_i)| - 3|B(T_i)| - 4}{2}$.

Demostración. Cada vez que se construye un árbol se toma como raíz un vértice de grado mínimo 3, luego cada árbol tiene como mínimo tres hojas. Ahora, notar que cada vez que se aplica el esquema 1 se añaden dos hojas al árbol y se quita una que sería la del vértice expandido. Por tanto al usar el esquema 1 se añaden exactamente un número de hojas igual al número de vértices negros. Si se usa cualquiera de las reglas con prioridad 2 entonces es claro que el número de vértices añadidos es al menos $|V(T_i)| - 3|B(T_i)| - 4$ pues el esquema uno añade 3 vértices y los 4 vértices que se restan son los que tiene como mínimo el árbol al principio. Y ahora, al aplicar el esquema 2, que es el que menos hojas añade de los de prioridad 2, el número de hojas añadido es al menos la mitad de los vértices que se añaden al árbol al considerar ese esquema.
Juntando estos razonamientos se tiene la acotación. □

Teorema 2.14. *El algoritmo obtiene un árbol de expansión T con un número de hojas que cumple*

$$\frac{l(T^*)}{l(T)} \leq 3.$$

Demostración. Para probar el teorema haremos uso del Corolario 2.12 y del Lema 2.13. Usando estos dos resultados tenemos que:

$$\frac{l(T^*)}{l(T)} \leq \frac{|V(F)| - 2k}{\sum_{i=0}^{k} \left(3 + |B(T_i)| + \dfrac{|V(T_i)| - 3|B(T_i)| - 4}{2}\right) - 2k} \leq$$

$$\leq \frac{2(|V(F)| - 2k)}{|V(F)| - |B(F)| - 2k + 2}$$

Esta segunda desigualdad viene simplemente de multiplicar en el numerador y denominador por 2 y aplicar que $\sum_{i=0}^{k}|V(T_i)| = |V(F)|$ y lo mismo para $|B(T_i)|$. Ahora, es claro que:

$$\frac{2(|V(F)|-2k)}{|V(F)|-|B(F)|-2k+2} \leq 2+\frac{2|B(F)|}{|V(F)|-|B(F)|-2k}$$

Notar que en el mejor caso de ejecución del algoritmo siempre se usará el esquema de prioridad 1, por lo que cada árbol T_i tendrá al menos 4 vértices, que son los añadidos al formar el árbol con un nodo de grado al menos 3, más los vértices que se añaden cada vez que se usa el esquema 1 que son $3|B(T_i)|$. Por tanto, este razonamiento nos lleva a la siguiente acotación:
$|V(F)| = \sum_{i=0}^{k}|V(T_i)| \geq \sum_{i=0}^{k}(4+3|B(T_i)|)$ por lo que $|V(F)| > 4k+3|B(F)|$. Por tanto,

$$\frac{l(T^*)}{l(T)} < 2+\frac{2|B(F)|}{2|B(F)|+2k}$$

Y si al aplicar el algoritmo obtenemos muchos vértices negros y a la vez $k << |B(F)|$, el término de la última desigualdad es muy cercano a 3. Por tanto,

$$\frac{l(T^*)}{l(T)} < 3$$

□

Por tanto, el algoritmo permite obtener un árbol de expansión T con al menos un tercio de las hojas del árbol de expansión con máximo número de hojas T^*. En [7] podemos encontrar el razonamiento y la demostración de que en realidad este algoritmo obtiene un árbol de expansión T con al menos la mitad de hojas que T^*. Este razonamiento no se incluye aquí debido a su extensión.

Coste computacional

Por último, comentar que el algoritmo tiene un tiempo de ejecución lineal. En particular es del orden $O(n+m)$ puesto que primero tenemos que examinar los vértices del grafo hasta encontrar uno de grado al menos 3 que pondremos como raíz de un árbol. Y para este vértice hay que examinar su lista de adyacencia para ver si alguno de sus vecinos se puede expandir por alguno de los esquemas.
De donde se puede observar que el tiempo de ejecución en su peor caso es $O(n+m)$.

Capítulo 3

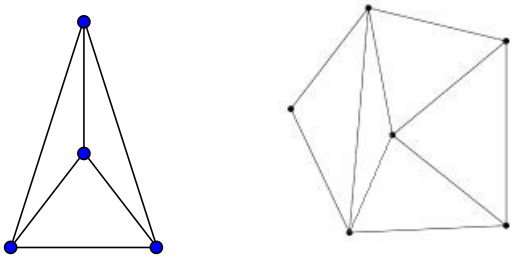
Triangulaciones planas

En este capítulo trataremos el problema del conjunto dominante conexo para triangulaciones planas. Hemos visto la equivalencia de este problema con el de encontrar un árbol de expansión con máximo número de hojas. Probaremos un resultado que nos permitirá asegurar que toda casi-triangulación con al menos cuatro vértices contiene un árbol de expansión sin vértices de grado 2.
Primero, introduzcamos los conceptos de triangulación plana y casi-triangulación en un grafo.

Definición 3.1. Se dice que un grafo plano G es maximal si al añadir cualquier arista al grafo este deja de ser plano.

Definición 3.2. Como grafo, una triangulación es un grafo plano maximal que además posee la propiedad de que al dibujarlo en un plano todas sus caras son triángulos.
Una casi-triangulación es un grafo plano en el cuál todas sus caras internas son triángulos.
El grafo dual de una triangulación es un grafo plano cúbico, es decir, un grafo cuyos vértices tienen todos grado 3.

Es claro que toda triangulación es una casi-triangulación pues todas sus caras son triángulos. Veamos un ejemplo de una triangulación y una casi-triangulación.



(a) (b)

Figura 3.1: (a)Triangulación (b) Casi-triangulación

La noción de triangulación varía según el área de conocimiento. Por ejemplo, en geometría se conoce como triangulación de un conjunto de puntos a un trazado cuyas caras son triángulos salvo la exterior que es de cualquier tamaño. En este capítulo, solamente consideraremos la noción definida en grafos.

Respecto al problema del conjunto dominante conexo en triangulaciones planas, existen artículos que acotan el número de dominación del grafo en triangulaciones, es decir, dada una triangulación G se puede saber que el conjunto dominante conexo óptimo en G tiene cardinalidad menor que un cierto número k. Nosotros nos centraremos en el problema de hallar un árbol de expansión sin vértices de grado 2 en casi-triangulaciones. En el Teorema 2.1 vimos que el problema de hallar un $\{1,3\}$-árbol de expansión en un grafo con vértices de grado menor o igual que 3 era NP-Completo. Por tanto, el

problema de hallar un árbol de expansión sin vértices de grado 2 también lo será.
A continuación presentamos un resultado que nos permite asegurar la existencia de este tipo de árbol de expansión en casi-triangulaciones.

Teorema 3.1. *Toda casi-triangulación G con al menos cuatro vértices contiene un árbol de expansión sin vértices de grado 2.*

Demostración. Procederemos por inducción sobre el número de aristas $|E(G)|$ de G. Denotaremos por n al número de vértices de G y llamaremos ST-2 a un árbol de expansión sin vértices de grado 2. Si $n = 4$ entonces es claro que G contiene a un ST-2.
Supongamos entonces que $n > 4$. Sea G una casi-triangulación tal que la cara externa está formada por el ciclo $C = x_1, x_2, ..., x_k, x_1$ con $k \geq 3$. Sea z un vértice que no está en el ciclo C y supongamos que los vértices x_i, x_{i+1} y z forman una cara interna. Entonces, si eliminamos la arista (x_i, x_{i+1}) de G obtenemos también una casi-triangulación G' en la cuál el ciclo de la cara exterior incluye al vértice z entre x_i y x_{i+1} y el resto de caras es claro que seguirán siendo triángulos. Además, G' tiene una arista menos que G luego por hipótesis de inducción, tendrá un ST-2.
Por tanto, podemos suponer sin pérdida de generalidad que toda cara formada por 3 vértices y una única arista de C tiene los tres vértices en C ya que en caso contrario, la eliminación de una arista con el procedimiento anterior no afectará a la construcción de un ST-2 en G y se tendrá el mismo $ST-2$ para G que para la casi-triangulación $G-(x_i, x_{i+1})$. A partir de ahora llamaremos 3-ciclo esencial a una cara cíclica formada por tres vértices y una única arista de C.

Supongamos primero que G no tiene 3-ciclos esenciales. Entonces podemos considerar dos casos:

- Si la cara C está formada por un número par de vértices mayor que 4. Entonces, como G no tiene 3-ciclos esenciales, los vértices x_1, x_2, x_3 y x_3, x_4, x_5 formarán dos caras triangulares internas. En este caso, si eliminamos de G los vértices x_2 y x_4 seguiremos teniendo una casi-triangulación pues las aristas (x_1, x_3) y (x_3, x_5) forman ahora parte de la cara externa. Por tanto, por hipótesis de inducción, la casi-triangulación $G - \{x_2, x_4\}$ tendrá un ST-2. Y si añadimos las aristas (x_2, x_3) y (x_3, x_4) entonces G tiene también un ST-2.

- Si la cara C está formada por un número impar de vértices. Entonces en este caso, la casi-triangulación no tendrá caras internas y solo tendrá la cara externa C pues, en caso contrario, al formar las caras triangulares internas siempre se obtiene al menos una que es un 3-ciclo esencial. Sin embargo, podemos unir los vértices x_1, x_2, x_3 y x_3, x_4, x_5 y formar dos caras triangulares internas que no son 3-ciclos esenciales. Y, ahora procedemos igual que en el punto anterior. Si eliminamos los vértices x_2 y x_4 obtenemos una casi-triangulación en el caso en el que $k > 5$, o el grafo completo de tres vértices K_3 si $k = 5$. Pero en ambos casos, es claro que si añadimos las aristas (x_2, x_3) y (x_3, x_4) a $G - \{x_2, x_4\}$, G tiene también un ST-2 que incluirá a esas dos aristas.

Supongamos ahora que G tiene solo un 3-ciclo esencial que denotaremos por C'. Notar que entonces la cara externa de G tiene que tener un número impar de vértices pues, si tuviese un número par, no podría tener un único 3-ciclo esencial. Supongamos también que C' está formado por el ciclo x_k, x_1, x_j, x_k con $j \neq 2$ y $j \neq k-1$ pues en estos dos casos no sería un 3-ciclo esencial. Y supongamos además que C' es tal que la casi-triangulación G' contenida en el ciclo $x_1, x_2, ..., x_j, x_k$ tiene las menos aristas posibles. Esto implica que G' no tiene 3-ciclos esenciales.
En efecto, si $j = 3$ entonces G' tiene 4 aristas y, por la construcción de C', G' no puede tener 3-ciclos esenciales. Si $j \neq 3$ entonces como G' tiene que tener el menor número de aristas posibles, G' tendrá como mucho las mismas aristas que en el caso $j = 3$ y, por tanto no tendrá 3-ciclos esenciales. Por tanto, si $j \neq 3$ entonces G tendrá que tener las caras formadas por los ciclos x_1, x_2, x_3, x_1 y x_3, x_4, x_5, x_3 pues en caso contrario existiría otro j que haría que G' tuviese menos aristas dentro del ciclo $x_1, x_2, ..., x_j, x_k$. Pero notar que este caso es el que hemos demostrado ya en el párrafo de arriba y por tanto G tiene un ST-2. Si $j = 3$, si eliminamos los vértices x_1 y x_2 es claro que seguimos teniendo una casi triangulación.

Por tanto, por hipótesis de inducción $G-\{x_1,x_2\}$ tiene un ST-2. Y si añadimos a esta casi-triangulación las aristas (x_2,x_3) y (x_1,x_3), que existen por la construcción de C', se tiene que G tiene también un ST-2. □

Este teorema induce el siguiente corolario que nos permite acotar el tamaño de un conjunto dominante conexo en casi-triangulaciones.

Corolario 3.2. *Toda casi-triangulación G con al menos cuatro vértices tiene un conjunto dominante conexo D tal que* $|D| \leq \frac{n-2}{2}$.

Demostración. Por el Teorema 3.1 sabemos que G tendrá un árbol de expansión T sin vértices de grado 2. Sabemos además que los vértices que no son hojas en T forman un conjunto dominante conexo D. Veamos cuál es el tamaño de D.
Denotemos por n_i al número de vértices de grado i en T y supongamos que $\Delta(T)=k$, $k\in\mathbb{N}$. Entonces, se cumplen las dos siguientes relaciones: $n_1+3n_3+4n_4+...+kn_k=2(n-1)$ y $n_1+n_3+n_4+...+n_k=n$. Juntando ambas ecuaciones llegamos a que:

$$2n_3+3n_4+4n_5+...+(k-1)n_k=2\left(n_3+\frac{3}{2}n_4+...+\frac{k-1}{2}n_k\right)=n-2$$

Y esto implica que:

$$n_3+n_4+...+n_k\leq n_3+\frac{3}{2}n_4+...+\frac{k-1}{2}n_k=\frac{n-2}{2}$$

Pero la suma de esos vértices es precisamente el número de vértices de T que no son hojas, es decir, el número de vértices que forman un conjunto dominante conexo en G. Por lo que se tiene que $|D|\leq \frac{n-2}{2}$. □

Para terminar, veamos el caso particular del octaedro para el cuál sabemos que existe un conjunto dominante conexo formado por dos vértices por el corolario anterior y que además contiene un árbol de expansión sin vértices de grado 2 por el teorema 3.1.

(a) (b)

Figura 3.2: (a) Octaedro (b) Conjunto dominante conexo formado por los vértices en rojo

Bibliografía

[1] THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVES, *Introduction to Algorithms*, MIT Press, Massachusetts, 1989.

[2] YAXIONG ZHAO, JIE WU, FENG LI, SANGLU LU, *On Maximizing the Lifetime of Wireless Sensor Networks Using Virtual Backbone Scheduling*, https://www.semanticscholar.org/paper/On-Maximizing-the-Lifetime-of-Wireless-Sensor-Using-Zhao-Wu/0a70589e283f52247cf10b8858910376b658828a.

[3] M. YUVARAJ, DR. DEVARAJ AN, SHEELASHOBANARANI GOVERNMENT COLLEGE OF TECHNOLOGY, *Routing in adhoc networks using minimum connected dominating sets*, https://www.researchgate.net/publication/308916087_Routing_in_Adhoc_Networks_Using_Minimum_Connected_Dominating_Sets.

[4] TETSUYA FUJIE, *An exact algorithm for the maximum leaf spanning tree problem*, http://cedric.cnam.fr/~bentzc/INITREC/Files/AP1.pdf.

[5] ROBERT JAMES DOUGLAS, *NP-completeness and degree restricted spanning trees*, Computer Science Department, San Francisco State University, San Francisco, CA 94132, USA 1990.

[6] S.GUHA, S.KHULLER, *Approximation Algorithms for Connected Dominating Sets*, Algorithmica 1998 Springer-Verlag New York Inc.

[7] ROBERTO SOLIS-OBA, *Approximation Algorithm for Finding a Spanning Tree with Maximum Number of Leaves*, Max Planck Institut für Informatik Im Stadtwald, 66123 Saarbrücken, Germany.

[8] CARSTEN THOMASSEN, MICHAEL O. ALBERTSON, DAVID M. BERMAN, JOAN P. HUTCHINSON, *Graph with homeomorphically irreducible spanning trees*, Journal of Graph Theory, June 1990.

[9] PROSENJIT BOSE, FERRAN HURTADO, *Flips in planar graphs*, Computatioal Geometry, (p.60-80), January 2009