El problema de coloreado óptimo de un grafo

Jorge Ballarín Corvinos

Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Director del trabajo: Alfredo García Olaverri
27 de junio de 2022

Prólogo

Dado $G = (V, E)$ un grafo, un k-coloreado de G es una función que asocia a cada vértice un entero desde 1 hasta k (color) de manera que dos vértices adycentes no tengan el mismo color. En este trabajo estudiaremos cuál es el coloreado óptimo de G, y por tanto, cuál es el $k \in \mathbb{N}$ más pequeño tal que existe un k-coloreado. Este número es el llamado número cromático, denotado como $\chi(G)$, el cual buscaremos a través de cotas y algoritmos.

Empezaremos definiendo conceptos básicos de la teoría de grafos, incluyendo una subsección de conectividad, y nos detendremos con los grafos de intervalos y los cordales. Se expondrá formalmente el problema de coloreado óptimo de un grafo, y veremos varias aplicaciones a problemas reales, como la organización de un congreso o la resolución de sudokus. También detallaremos la relación del coloreado con el teorema de los cuatro colores. Finalmente, explicaremos tres problemas de grafos (coloreado óptimo de ejes, máximo subgrafo completo y máximo conjunto independiente) y sus relaciones con el coloreado óptimo de un grafo.

Después explicaremos varios teoremas intentando acotar el número cromático superiormente. Además introduciremos el algoritmo voraz, y trataremos de reducir la coloración de un grafo a la de ciertos subgrafos de él. Todo ello nos conducirá hasta el teorema de Brooks. Pero no solo nos interesa calcular el número cromático, también es interesante saber cuántos k-coloreados tiene un grafo G dado, por eso definiremos el polinomio cromático. También veremos maneras de calcular los coeficientes de este polinomio. Igualmente explicaremos el algoritmo de contracción para el cálculo del número cromático.

Más adelante, veremos que el problema de coloreado óptimo de un grafo es **NP-hard**. Previamente haremos una introducción de la complejidad computacional de un problema donde explicaremos los principales tipos de problema: **P**, **NP**, **NP-hard** y **NP-completo**. Veremos que los otros problemas de grafos que hemos visto son también problemas **NP-hard**. De hecho, la complejidad de este problema es tal que decidir si un grafo se puede colorear con 3 colores ya es **NP-completo** (y también para valores mayores que 3). Continuaremos con una sección de algoritmos heurísticos, en la que explicaremos el algoritmo DSatur y el algoritmo RLF. Pese a toda esta búsqueda de valores aproximados, veremos que es difícil decir si una cota dada del número cromático es buena.

Terminaremos el trabajo centrándonos en dos tipos específicos de grafos, de intervalos y cordales. Explicaremos qué es un orden de eliminación perfecto de los vértices de un grafo G, ya que en combinación con el algoritmo voraz nos ofrece una coloración de $\chi(G)$ colores. Justificaremos que un grafo es cordal si y solo si tiene un orden de eliminación perfecto, y que por tanto los grafos de intervalos son un tipo de grafo cordal. Finalmente, se incluirán propiedades y algoritmos para reconocer si un grafo dado es de intervalos (cordal) o no.

Abstract

Given $G = (V,E)$ a graph, a k-coloring of G is a function that associates to each vertex an integer from 1 to k (color) such that two adjacent vertices do not have the same color. In this project, we will study what is the optimal vertex coloring of G, and therefore, what is the smallest $k \in \mathbb{N}$ such that there is a k-coloring. This number is called the chromatic number, denoted as $\chi(G)$, which we will look for through bounds and algorithms.

This assignment is divided into 4 chapters, each of them with several sections and subsections. We detail each of them:

The first chapter has two different sections. In the first one, we will start by defining basic concepts of graph theory, from directed and undirected graphs to the main types of graphs, such as complete graphs, trees and regular graphs. We will also include a subsection on connectivity, and justify the existence of a tree called BC-tree for any connected graph. The last concepts that we will define before presenting the problem will be two types of graphs, intervals and chordals, which we will study in the last chapter as they have good coloring properties. In order to acquire a better understanding of the concept of interval graph, it will be explained with an example.

In the second part of the chapter we will have the formal definition of the vertex coloring problem, and we will see several applications to real problems, such as the organization of a congress or the resolution of Sudoku puzzles. We will also detail the connection betwween coloring and the famous four color theorem, where the concept of dual graph is important. Finally, we will explain three graph problems (edge-coloring, maximum clique and maximum independent set) and their connections with the vertex coloring of a graph. Between these relations we will have the first lower bounds of $\chi(G)$.

In the second chapter we will try to limit the chromatic number higher. In order to achieve this, we will explain several theorems, we will also introduce the first coloring algorithm, the greedy algorithm, and we will try to reduce the coloring of a graph to that of certain subgraphs of it. All this will lead us to Brooks' theorem. But we are not only interested in calculating the chromatic number, it is also interesting to know how many k-colors a given graph G has, so we will define the chromatic polynomial. We will also find ways to calculate the coefficients of this polynomial, and related to this, we will explain the contraction algorithm for calculating the chromatic number, since it will be useful for us to see the properties of the polynomial.

In the third part we will see why until then we had not been able to find a simple formula or expression for the chromatic number, and that is because the vertex coloring problem is **NP-hard**. In any case, we will previously introduce the computational complexity of a problem where we will explain the main types of problems: **P**, **NP**, **NP-hard** and **NP-complete** . We will see that the edge-coloring problem, the maximum clique problem and the maximum independent set problem are also **NP-hard** problems. In fact, the complexity of this problem is such that deciding whether a graph can be colored with 3 colors or not is already a **NP-complete** decision problem (and also for values greater than 3).

For all these reasons, we will continue with the search for algorithms that approximate $\chi(G)$ as much as possible, and thus we will have a section on heuristic algorithms, in which we will explain the DSatur algorithm, related to the greedy algorithm, and the RLF algorithm, related to independent sets. Despite all this search for approximate values, we will see that it is difficult to tell whether a given bound on the

chromatic number is a good one.

We will finish the study by focusing on two specific types of graphs, intervals and chordals. We will explain what a perfect elimination order of the vertices of a graph G is, since in combination with the greedy algorithm it offers us a coloring of $\chi(G)$ colors. We will justify that a graph is chordal if and only if it has a perfect elimination order, and therefore interval graphs are a type of chordal graph. Finally, properties and algorithms will be included to recognize if a given graph is interval (chordal) or not, including an example in which we apply the Lex-BFS search algorithm for a perfect elimination order.

Índice general

Capítulo 1

Conceptos básicos

En este capítulo definiremos los conceptos que vamos a utilizar en todo el trabajo. Tras ello, empezaremos a tratar el problema de coloreado óptimo de un grafo, lo compararemos con otros problemas de grafos y estudiaremos la complejidad computacional de los algoritmos.

1.1. Introducción a la teoría de grafos

No podemos empezar a estudiar la teoría de grafos sin la definición de grafo. Tras ella, definiremos sus elementos más importantes.

Definición. Un *grafo no dirigido* es un par $G = (V,E)$ tal que V es un conjunto finito de elementos, a los cuales llamaremos *vértices* o *nodos*, y E es un conjunto de pares no ordenados de vértices, a cuyos elementos llamaremos *ejes*.

Con respecto a la notación, usaremos los símbolos $u, v, ...$ para los vértices y (u, v) para los ejes. Notar que el eje (u, v) es el mismo que el (v, u). El número de vértices se denomina *orden de G* y se denota $n = |V|$, y el número de ejes se denomina *tamaño de G* y se denota $m = |E|$.

Nos podemos preguntar qué pasaría si importa el orden en que tomemos el par de vértices que forma el eje. Surge entonces el siguiente tipo de grafo.

Definición. Un *grafo dirigido* es un par $G = (V,A)$ tal que V es un conjunto finito de elementos, a los cuales llamaremos *vértices*, y A es un subconjunto de $V \times V$ a cuyos elementos llamaremos *arcos*.

En este trabajo hablaremos de grafos refiriéndonos a grafos no dirigidos y, cuando sea el caso, se especificará si el grafo es dirigido. Además, trabajaremos con *grafos simples*, es decir, grafos que no tienen pares repetidos ni bucles (ejes de la forma (u, u)). Nos serán útiles también los siguientes conceptos.

Definición. Dado un grafo G, llamamos *paseo* de longitud l desde el vértice u al vértice v a una sucesión de vértices $(v_0, v_1, ..., v_l)$ tal que $v_0 = u$, $v_l = v$ y $(v_{k-1}, v_k) \in E$ para $k = 1, 2, ...l$. Decimos *camino* a un paseo que no tiene ejes repetidos, y denominamos *ciclo* a un camino de al menos un eje en el que el vértice inicial y final coinciden. Si un camino o un ciclo no contienen vértices repetidos, diremos que son *simples*.

En el estudio que vamos a hacer, a los ciclos simples de longitud l los llamaremos sencillamente ciclos o l-ciclos, ya que aparecerán en varias ocasiones. Su notación será C_l.

Definición. Decimos que dos vértices u, v son *adyacentes* o *vecinos* si en el conjunto de los ejes se encuentra el elemento (u, v), y en ese caso el eje (u, v) es *incidente* a u y a v. Denotamos por $\Gamma(u)$ al conjunto de vecinos del vértice u. El *grado* de u es el número de ejes incidentes en él, lo denotaremos como $d(u)$.

Dado un grafo G, denotaremos $\Delta(G)$ como el máximo grado de los vértices de G y $\delta(G)$ como el mínimo grado. Notar que siempre existirá un vértice que alcance máximo (o mínimo) grado por tener un número finito de vértices.

Definición. Un *subgrafo* de $G = (V,E)$ es otro grafo $G_1 = (V_1,E_1)$ tal que $V_1 \subseteq V$ y $E_1 \subseteq E$.

El *subgrafo inducido* por el subconjunto de vértices V_1 es el subgrafo $G_1 = (V_1,E_1)$ tal que $E_1 = E \cap (V_1 \times V_1)$. De igual manera, un subgrafo inducido por un subconjunto de ejes E_1 es $G_1 = (V_1,E_1)$ tal que V_1 está formado por los vértices que aparecen en los extremos de los ejes de E_1.

Definición. Un grafo es *completo* de orden n si tiene n vértices y todas las conexiones posibles en él (salvo bucles), es decir, tiene $\binom{n}{2}$ ejes. Lo denotaremos K_n.

Definición. Un grafo $G = (V,E)$ es *bipartito* si V puede dividirse en dos partes no vacías V_1 y V_2 tales que todos los ejes tienen un extremo en V_1 y otro en V_2. Denotando $|V_1| = n_1$, $|V_2| = n_2$, cuando aparecen todos los ejes de V_1 a V_2 se dice que es un grafo *bipartito completo* y se denota $K_{n_1;n_2}$.

Definición. Un grafo es *plano* cuando puede ser dibujado en el plano sin que ningún eje se cruce.

Definición. Un grafo $G = (V,E)$ es un *árbol* si para todo $u,v \in V$ existe un único camino de u hasta v.

También se caracterizan los árboles como grafos conexos y sin ciclos.

Definición. Un grafo es *k-regular* si $d(v) = k$ para todo $v \in V$.

1.1.1. Conectividad

Definición. Un grafo $G = (V,E)$ es *conexo*, cuando para todo $u,v \in V$ existe un camino desde u hasta v. Un subgrafo maximal y conexo de G se denomina *componente* o *componente conexa*.

Definición. Diremos que $v \in V$ es un *vértice de corte* de G si el número de componentes de $G - \{v\}$ es mayor que el número de componentes de G. Un grafo conexo G se dice 2-*conexo* si para cada $v \in V$, $G - \{v\}$ es conexo, es decir, si es un grafo conexo sin vértices de corte. En general, un grafo conexo G se dice *k-conexo* si para cada conjunto de $l < k$ vértices $v_1,...,v_l \in V$, $G - \{v_1,...,v_l\}$ es conexo.

Definición. Un *bloque* de un grafo G es un subgrafo 2-conexo maximal.

Proposición 1.1. *Sea G un grafo conexo. Entonces G puede descomponerse en bloques de manera que:*

(a) Dos bloques cualesquiera de G tendrán como máximo un nodo en común.

(b) Los bloques de G forman una partición de sus ejes.

(c) No hay ningún ciclo entre bloques, es decir, no existen $B_0,B_1,...,B_l$ bloques de G tales que $V(B_i) \cap V(B_{i+1}) \neq \emptyset$, con $0 \leq i \leq l$, donde $G_{l+1} = G_0$.

(d) Todos los bloques tienen al menos un vértice de corte.

Demostración. Todos los vértices y ejes de G están en algún bloque ya que dos vértices unidos entre sí forman trivialmente un grafo 2-conexo. Probamos ahora las propiedades de esta descomposición.

(a) Supongamos que los bloques B_1,B_2 tienen al menos dos nodos comunes y lleguemos a contradicción. Aunque eliminemos un nodo $u \in V(B_1) \cup V(B_2)$, seguimos teniendo un camino desde cualquier nodo de $B_1 - \{u\}$ hasta cualquier nodo de $B_1 \cap B_2 - \{u\}$ por ser B_1 un bloque y haber dos nodos en la intersección. También hay camino desde cualquier nodo de $B_2 - \{u\}$ hasta cualquier nodo de $B_1 \cap B_2 - \{u\}$, y por tanto para todo $v,w \in V(B_1) \cup V(B_2) - \{u\}$ existe un camino desde v hasta w. Así, hemos probado que $B_1 \cup B_2$ es 2-conexo, lo que contradice la maximalidad de B_1 y B_2.

(b) Todos los ejes de G están en algún bloque pero dos bloques tienen como máximo un nodo en común, así que no pueden tener ejes en común. Así, cada eje de G está en un único bloque.

(c) Supongamos que tenemos una secuencia de bloques $B_0, B_1, ..., B_l$ como en el enunciado, definimos $B = \bigcup_{i=0}^{l} B_i$. Es claro que B es conexo y que no puede ser separado por ningún v_i, $0 \leq i \leq l$. Entonces, B es 2-conexo, lo que contradice la maximalidad de B_i.

(d) Para todo bloque B_1, podemos asegurar que existe un bloque B_2 tal que ambos bloques tienen un vértice en común v, ya que todos los ejes y vértices de G están en algún bloque y G es conexo. Además, como no puede haber ciclos entre bloques por (c), $G - \{v\}$ no es conexo. Esto hace que v sea un vértice de corte. Por tanto, el vértice en común entre dos bloques siempre es vértice de corte, por lo que todos los bloques tienen un vértice de corte. □

La estructura de los bloques y vértices de corte de un grafo conexo G puede ser descrita por un árbol llamado BC-tree.

Definición. Denotamos por $\mathscr{B}$ al conjunto de bloques de un grafo conexo G, y $\mathscr{C}$ al conjunto de vértices de corte. Sea $B(G)$ el grafo bipartito cuyo conjunto de vértices tiene la partición $\{\mathscr{C}, \mathscr{B}\}$, y cuyos ejes son los pares $\{c, B\}$, donde c pertenece a B. Entonces $B(G)$ es un árbol, llamado *BC-tree*.

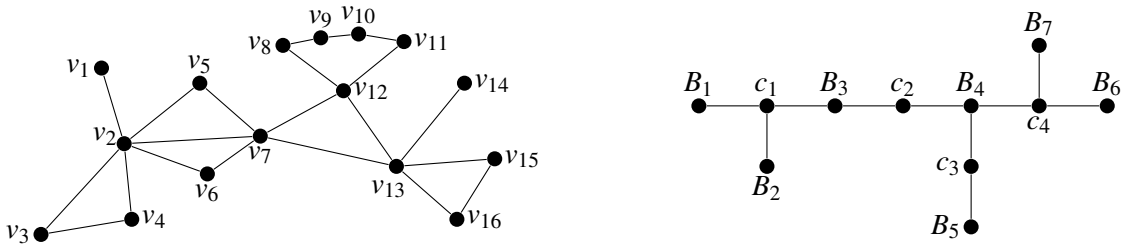

Figura 1.1: Construcción del BC-tree asociado a un grafo conexo G. Los bloques son: $B_1 = \{v_1, v_2\}$, $B_2 = \{v_2, v_3, v_4\}$, $B_3 = \{v_2, v_5, v_6, v_7\}$, $B_4 = \{v_7, v_{12}, v_{13}\}$, $B_5 = \{v_8, v_9, v_{10}, v_{11}, v_{12}\}$, $B_6 = \{v_{13}, v_{14}\}$ y $B_7 = \{v_{13}, v_{15}, v_{16}\}$. Los vértices de corte son: $c_1 = v_2$, $c_2 = v_7$, $c_3 = v_{12}$ y $c_4 = v_{13}$.

1.1.2. Grafos de intervalos y cordales

Acabamos la sección introduciendo dos tipos de grafos que vamos a estudiar en el último capítulo.

Definición. Un *grafo de intervalos* es aquel que puede ser representado como intersección de intervalos en $\mathbb{R}$. Esta representación es tal que por cada intervalo tenemos un vértice (para cada I_u existe un $u \in V$) y tenemos un eje si y solo si los intervalos correspondientes a cada vértice se intersecan ($(u, v) \in E$ si y solo si $I_u \cap I_v \neq \emptyset$).

Definición. Se denomina *grafo cordal* a aquel que no contiene un k-ciclo inducido, para todo $k \geq 4$.

Reciben este nombre ya que dado un k-ciclo con $k \geq 4$, a los ejes entre dos vértices no consecutivos se les denomina cuerdas.

Lema 1.2. *Un k-ciclo, $k \geq 4$, no puede ser un grafo de intervalos. Así se tiene que un grafo de intervalos es siempre cordal.*

Demostración. Supongamos que se puede realizar una representación de un k-ciclo con intervalos, $k \geq 4$, y lleguemos a absurdo. Necesitamos k intervalos, y que cada uno de ellos se interseque con otros dos.

Sea un intervalo I_1, y sean I_2, I_k los dos intervalos con los que se corta. Notar que I_2 e I_k no tienen que intersecar entre sí (no hay un 3-ciclo), y además I_2 e I_k no están contenidos en I_1, ya que si lo estuvieran,

entonces el otro intervalo con el que se cortan también cortaría a I_1, pero no puede hacerlo pues el k-ciclo no tiene cuerdas. Así, un intervalo empezará a la izquierda de I_1, supongamos I_2, y acabará antes de que acabe I_1; mientras que I_k comenzará después de que acabe I_2 pero antes de que acabe I_1, y se alargará hasta más allá del punto final de I_1.

I_2, además de I_1, interseca con I_3. Igual que se ha razonado antes, I_3 no está contenido en I_2 y no conecta con I_1. Construimos con esta idea este intervalo y los intervalos $I_4,...,I_{k-1}$, donde I_j corta solamente con I_{j-1} y con I_{j+1}, para $j = 4,...,k-2$, y obtenemos un esquema similar a 1.2.

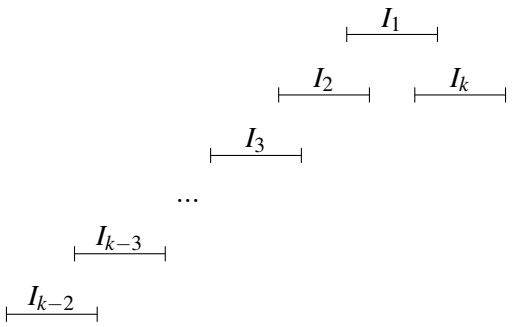

Figura 1.2: Construcción de los primeros $k-1$ intervalos. No es posible trazar el último intervalo I_{k-1}.

Finalmente, queda construir un último intervalo I_{k-1} que interseque con I_{k-2} y con I_k, pero que no lo haga con ningún intervalo más, cosa que es imposible, ya que tendría que tener el punto final antes de que empiece el intervalo I_{k-3} y tendría que tener punto inicial más a la derecha del final de I_1. Con todo ello, llegamos a contradicción, es decir, los ciclos de longitud $k \geq 4$ no son grafos de intervalos, y por tanto, los grafos de intervalos son cordales. □

Para comprender los grafos de intervalos veremos un ejemplo, obtenido del libro de Gondram y Minoux (1984).

Ejemplo 1. Hace más de 10 años, el duque de Densmore murió por la explosión de una bomba que destruyó su castillo. Antes de su muerte, el duque invitó a sus 7 exmujeres (Ann, Betty, Charlotte, Edith, Felicia, Georgina y Helen) a pasar unos días en su castillo. Ellas eran las principales sospechosas y fueron interrogadas. No se acordaban de las fechas exactas ya que fue hace tiempo, pero todas dijeron que habían estado solo durante un periodo de tiempo y que se acordaban de las otras exmujeres que habían conocido durante su estancia. Esto fueron los datos que se obtuvieron.

Ann conoció a Betty, Charlotte, Felicia y Georgina.
Betty conoció a Ann, Charlotte, Edith, Felicia, Helen.
Charlotte conoció a Ann, Betty y Edith.
Edith conoció a Betty, Charlotte y Felicia.
Felicia conoció a Ann, Betty, Edith y Helen.
Georgina conoció a Ann y Helen.
Helen conoció a Betty, Felicia y Georgina.

Los datos concuerdan, ya que si la exmujer X dice que coincidió con la Y, Y también afirma que coincidió con X. Aparentemente no se puede obtener ninguna conclusión de esta información, pero todavía no se ha comprobado si es verosímil que las exmujeres se hayan relacionado así habiendo estado cada una un único intervalo de tiempo.

Seguimos la investigación construyendo el grafo tal que cada exmujer se corresponde con un nodo, y dos nodos están conectados si y solo si se conocieron las respectivas exmujeres durante su estancia. Dicho grafo está representado en la figura 1.3.

Tenemos que ver si este grafo es de intervalos, ya que en caso de que lo fuera, cada mujer estaría representada con un único periodo de tiempo, su intervalo. Además, los ejes corresponderían a las intersecciones de dichos intervalos, cuando coinciden en el tiempo. En caso de no ser de intervalos, sería porque alguna de las exmujeres mintió y estuvo en varias ocasiones.

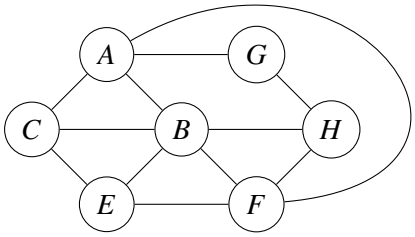

Figura 1.3: Grafo con la conexión de las mujeres que se conocen.

Se ve claramente que el grafo contiene tres ciclos de longitud 4 sin cuerdas, $ACEF$, $AGHF$ y $AGHB$, y que por tanto no es un grafo de intervalos. El único nodo que está en los tres ciclos es A, así que al quitar cualquier nodo distinto de A vamos a seguir teniendo algún ciclo de longitud 4, y por tanto seguirá habiendo una posible mentirosa. En cambio, como se observa en la figura 1.4, el grafo resultante de quitar el nodo A y sus conexiones no tiene el problema de los ciclos de longitud 4, y asociando adecuadamente un intervalo a cada nodo, observamos que sí es un grafo de intervalos.

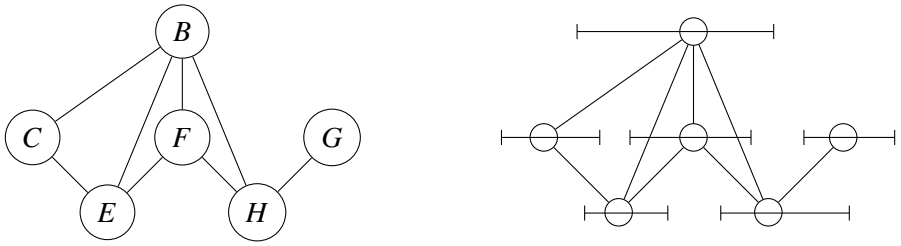

Figura 1.4: Grafo del problema sin el nodo A y representación con intervalos.

No sabemos con certeza quién ha matado al duque, pero de nuestro estudio se obtiene un dato interesante, y es que es seguro que alguna de las exmujeres miente. Además, se ha observado que si solo ha mentido una persona, entonces es Ann. Lo más probable es que ella sea la que ha mentido, por lo tanto es la principal sospechosa del asesinato.

1.2. Coloreado óptimo de un grafo y problemas relacionados

Definición. Un *k-coloreado* o *coloreado* de k colores de un grafo G es una función $c : V \to \{1,2,...,k\}$ que a cada vértice le asocia un número desde 1 hasta k de forma que si $(u,v) \in E$, entonces $c(u) \neq c(v)$.

Observamos que siempre existe un k suficientemente grande tal que un grafo G es k-coloreable (trivialmente tomar $k = n$), pero el problema de *coloración óptima* no es encontrar una coloración, sino encontrar aquella que utiliza el mínimo número de enteros, el k más pequeño posible.

El nombre de coloreado se entiende mejor si representamos gráficamente el grafo. Entonces el problema consistirá en pintar los vértices de manera que dos vértices adyacentes no tengan el mismo color, asegurándonos además de que no podemos usar menos colores. Parece un ejercicio sencillo, y lo es para grafos pequeños, pero a medida que el grafo aumenta en orden y tamaño se convierte en un problema difícil de tratar, sin una fórmula explícita o un método rápido de cálculo. Todo ello motiva la definición siguiente.

Definición. Dado un grafo G, llamamos *número cromático* y lo denotamos como $\chi(G)$, al mínimo número k tal que existe una k-coloración de G.

Es claro que $\chi(K_n) = n$, $\chi(C_{2k}) = 2$ y $\chi(C_{2k+1}) = 3$. Además, G es bipartito si y solo si $\chi(G) \geq 2$ (incluimos el caso $\chi(G) = 1$ si no hay ejes). En particular, los árboles son bipartitos.

Veamos unas aplicaciones a la vida real de este problema.

Ejemplo 2. Tenemos que organizar las charlas de un congreso (todas ellas de igual duración) de manera que ningún asistente se pierda una a la cual quiera asistir. Se suponen en este ejemplo que tenemos tantas salas como nos sean necesarias para albergar las charlas. El objetivo es organizar el congreso de manera que este dure lo menos posible.

En este caso, creamos el grafo $G = (V,E)$ de tal manera que los vértices serán cada una de las charlas, y dos vértices estarán conectados si y solo si un asistente quiere ir a las charlas correspondientes a dichos vértices. De esta manera, el coloreado óptimo del grafo G nos daría una solución al problema de organización, ya que los colores corresponderían a los distintos turnos en los que tienen que empezar las charlas, y hemos encontrado que el número de colores es el mínimo.

Ejemplo 3. Los cuadrados latinos y los sudokus se pueden resolver con ayuda del coloreado de grafos.

Un cuadrado latino es una rejilla de tamaño $l \times l$ con l símbolos diferentes (normalmente números desde 1 hasta l). Cada casilla debe de tener un símbolo de manera que aparezca solo una vez por fila y una por columna. El sudoku es un cuadrado latino de tamaño 9×9 que se divide a su vez en 9 cuadrados de dimensión 3×3 disjuntos tales que en cada cuadrado solo puede aparecer cada símbolo una vez.

Planteamos la resolución del cuadrado latino haciendo corresponder cada casilla con un nodo, teniendo así un grafo de $n = l^2$ nodos. Cada nodo estará conectado a los nodos que corresponden a casillas que están en su misma columna y a los de su misma fila. Así, el grafo tendrá $m = l^2(l-1)$ ejes, con todos los nodos de grado $2(l-1)$. Cada l-coloración del grafo da un cuadrado latino, donde nodos de color k corresponden al número k.

Para resolver un sudoku con el coloreado de grafos, procederemos de la misma manera que con los cuadrados latinos pero añadiendo la restricción de los cuadrados 3×3, de manera que tendremos $n = 81$ nodos; y cada uno de ellos, además de las conexiones con los 8 nodos de su fila y los otros 8 de su columna, se conectará con los 8 nodos de su cuadrado. Los nodos tendrán grado 24, y el grafo total tendrá $m = 972$ ejes. Esto dará una manera de generar todas las 9-coloraciones del grafo, y por tanto, todos los sudokus. Resolver un sudoku es ver si el grafo que construimos admite una 9-coloración. Cuando se plantea un sudoku como rompecabezas, se dan unos números repartidos por la cuadrícula, y el objetivo es completarlo en estas condiciones. Normalmente son de solución única para que puedan resolverse con razonamientos lógicos, por tanto solo tiene que haber una única coloración compatible con los datos iniciales.

Ejemplo 4. El teorema de los cuatro colores afirma que no hacen falta más de cuatro colores para colorear las regiones conexas de un mapa de manera que dos regiones adyacentes no tengan el mismo color.

La demostración de este teorema se dio a finales de siglo XX tras más de 120 años desde el primer enunciado del problema. Esta prueba fue hecha con ordenador.

Para ver la relación con el coloreado de grafos consideramos el concepto de *grafo dual* G^* de un grafo plano G, que se construye de la siguiente manera. En primer lugar, dibujamos un vértice v_i^* por cada región F_i de G; y en segundo lugar, por cada eje e en G dibujamos una línea e^* tal que cruza e pero no cruza ningún otro eje en G, y unimos los dos vértices en G^* correspondiendo a las dos regiones de G que el eje e separa. Notar que de este proceso se observa que el grafo dual es también plano. Vemos la construcción del grafo plano en la figura 1.5.

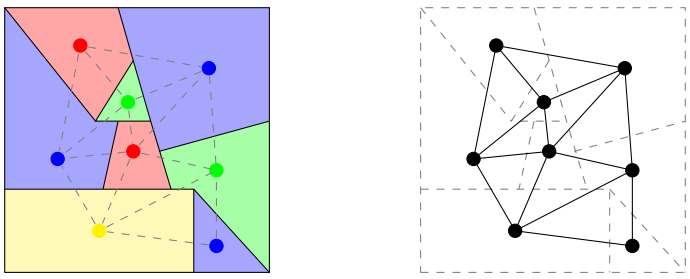

Figura 1.5: Construcción del grafo dual G^* de un grafo plano G, con el coloreado.

En el lenguaje de grafos, el teorema de los cuatro colores dice que todo grafo plano se puede colorear con 4 colores o menos.

1.2.1. Problemas relacionados

Problema del coloreado óptimo de los ejes de un grafo

También hay una variante de la coloración que consiste en asociar un número entero a cada eje (en vez de a los vértices). Surge así una nueva definición, el coloreado de ejes, que consistirá en una función $ce : E \to \{1,2,...,k\}$ que a cada eje le asocia un número desde 1 hasta k de forma que dos ejes que incidan en el mismo vértice no tengan el mismo color. El mínimo número k tal que existe un coloreado de ejes de k colores lo llamamos *número cromático de ejes* y se denota como $\chi'(G)$.

Este problema está directamente relacionado con el coloreado óptimo de los nodos. Dado un grafo G, su *grafo línea* $L(G)$ es aquel que tiene un vértice para cada arista de G, y dos vértices son adyacentes si y solo si sus aristas correspondientes de G tienen un vértice común en G. Claramente, $\chi'(G) = \chi(L(G))$.

Ejemplo 5. Cada uno de los n empresarios desea reunirse con alguno de los otros empresarios. Asumiendo que cada reunión dura un día y que en cada reunión hay 2 empresarios, se nos plantea el problema de organizar estas reuniones de manera que duren lo menos posible.

Para la resolución del problema, creamos un grafo G que posea n nodos (cada uno de ellos corresponde a un empresario), de manera que dos vértices estén unidos, si y solo si, los empresarios correspondientes a dichos vértices quieren reunirse. Entonces, el problema nos pregunta por el mínimo número de colores que necesita un coloreado de ejes del grafo G.

Problema del máximo subgrafo completo de un grafo

Definición. Un *clique* es un subconjunto de vértices de un grafo G tal que el subgrafo de G inducido por ese conjunto de vértices es un grafo completo.

Otro problema relacionado con el de coloreado de un grafo G es el de encontrar el máximo orden de un subgrafo completo contenido en G, esto es, encontrar el clique de G de máximo orden. A este número se le llama *clique number* y se denota $\omega(G)$.

Claramente si $K_k \subseteq G$, entonces $\chi(G) \geq \chi(K_k) = k$. En particular, tomando el máximo subgrafo completo se tiene que $\chi(G) \geq \omega(G)$.

Problema del máximo conjunto independiente de un grafo

Definición. Dado un grafo $G = (V,E)$, un subconjunto S de V se dice que es un *conjunto independiente* si no existe ningún eje que tenga ambos extremos en S. Si este conjunto no es además subconjunto de otro conjunto independiente, se dice *conjunto independiente maximal*.

El problema del máximo conjunto independiente consiste en encontrar el conjunto independiente de G que tenga el mayor número posible de vértices. El número de vértices que contiene el máximo conjunto independiente se denomina *número de independencia* de G y se denota por $\alpha(G)$.

Este problema y el anterior están relacionados con lo que se denomina grafo complementario.

Definición. El *grafo complementario* de un grafo G es un grafo H con los mismos vértices tal que dos vértices distintos están conectados en H si y solo si no son adyacentes en G. Lo denotamos por $\bar{G}$.

Si S es un conjunto de vértices independiente en $G = (V,E)$, entonces S es un clique en el grafo complementario $\bar{G}$. Así, $\alpha(G) = \omega(\bar{G})$.

Lema 1.3. *Sea G un grafo, se tiene que* $\chi(G) \geq \text{máx}\left\{\omega(G), \dfrac{|G|}{\alpha(G)}\right\}$.

Demostración. Que $\chi(G) \geq \omega(G)$ ya lo hemos visto. Sea una coloración de G de k colores. Todos los nodos de un mismo color son independientes, es decir, existen $S_1, ..., S_k$ subconjuntos independientes de V tales que $|S_1| + \cdots + |S_k| = n$. Sea S_i el mayor de los subconjuntos anteriores de tamaño $\alpha(G)$, que cumple $\alpha(G) \geq \frac{n}{k}$. Esto es, si y solo si $k \geq \frac{n}{\alpha(G)}$, y de esta forma, se tiene $\chi(G) \geq \frac{|G|}{\alpha(G)}$. □

Capítulo 2

Teorema de Brooks y polinomio cromático

Se han visto fórmulas que acotaban al número cromático inferiormente, como la relación con el clique number o la del número de independencia. Buscamos ahora expresiones que acoten este valor superiormente, de manera que la solución la podamos encontrar entre unos pocos valores.

2.1. Teorema de Brooks

Sin ninguna fórmula ni método para calcular el número cromático, surge inicialmente la idea de ir probando coloraciones pintando los vértices de manera ordenada. Describimos de esta forma uno de los algoritmos más simples para el coloreado de grafos.

Algoritmo voraz

Sea $G = (V, E)$ un grafo y sea $\{v_1, v_2, ..., v_n\}$ una ordenación de sus vértices. Se colorean los vértices en dicho orden, asignando a v_i el color disponible más pequeño no usado por los vecinos de v_i, añadiendo un color nuevo si fuera necesario.

Este algoritmo no siempre va a producir una buena coloración, y es fácil que use más colores de los necesarios. No obstante, en muchas ocasiones es interesante ver varias ordenaciones del grafo y aplicar en cada una este algoritmo para ver si mejora. Vemos esto en la figura 2.1, donde observamos un grafo bipartito (que por ser bipartito admite un 2-coloreado, uno para cada conjunto de nodos). El primer coloreado con la ordenación de vértices $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ obtiene 4 colores, mientras que la segunda ordenación $\{v_1, v_3, v_5, v_2, v_4, v_6\}$ necesita 2 solamente y por tanto es más eficaz.

Figura 2.1: Ejemplo de la importancia de una buena ordenación en el algoritmo voraz.

Veamos cuantos colores usa el algoritmo voraz. Cuando coloreamos el vértice v_i, con $d(v_i)$ vecinos, tenemos seguro al menos un color disponible de entre los $d(v_i)+1$ primeros. Esto nos ocurre en todos los pasos del algoritmo, por tanto $\chi(G) \leq d(v_i)+1$ para todo $i = 1, ..., n$. Tomando en particular el vértice con mayor grado, se tiene que $\chi(G) \leq \Delta(G)+1$.

Además, en cada paso lo que realmente nos importa no es el grado del vértice en sí, sino que lo importante es ver el número de vecinos que hemos coloreado antes de colorear dicho vértice. Siguiendo esta idea enunciamos el siguiente teorema.

Teorema 2.1. *Sea* $k := \max_H \delta(H)$*, para H subgrafo inducido de un grafo G. Entonces* $\chi(G) \leq k+1$.

Demostración. En primer lugar, tomamos un vértice de G tal que tenga grado como máximo k. Dicho vértice va a existir ya que $\delta(G) \leq \max_H \delta(H) = k$. Llamémosle v_n, y sea ahora $H_{n-1} = G - \{v_n\}$. Por el mismo razonamiento que antes, existe un vértice de grado como máximo k en H_{n-1}, supongamos v_{n-1}, y así denotamos $H_{n-2} = G - \{v_n, v_{n-1}\}$. Repetimos este proceso hasta el último vértice, obteniendo una ordenación $v_1, v_2, ..., v_n$ donde cada v_j está unido a como mucho k vértices antes que él. En el algoritmo voraz nos fijamos únicamente al colorear en los vértices ya pintados, por tanto, nunca se van a necesitar $k+2$ colores. □

En alguna ocasión sabremos como colorear un subgrafo H_0 de G. Podríamos empezar pues coloreando ese subgrafo y después seguir con el resto del grafo. Esto nos da el siguiente resultado que se puede ver como una extensión del teorema anterior.

Teorema 2.2. *Sea* H_0 *subgrafo de G. Supongamos que todo subgrafo H satisfaciendo* $H_0 \subset H \subset G$*,* $V(H_0) \neq V(H)$*, tiene un vértice* $v \in V(H) - V(H_0)$ *con* $d(v) \leq k$*. Entonces* $\chi(G) \leq \max\{k+1, \chi(H_0)\}$. □

Notar que si tenemos un grafo que no es conexo y por tanto se tienen que $G_1, ..., G_l$ son sus componentes conexas, entonces $\chi(G) = \max_{i=1,...,l} \chi(G_i)$. En el caso de grafos conexos, puede suceder que tengamos un vértice (o un conjunto de vértices) que si los quitamos desconectan al grafo, y que entonces el problema de coloreado del grafo pueda reducirse a un problema de coloreado de ciertos subgrafos de él.

Lema 2.3. *Sea G un grafo, se tiene que* $\chi(G) = \max(\chi(B_i))$*, con* B_i *bloque de G.*

Demostración. Sea G un grafo con su descomposición BC-tree. Para su coloración, empezamos primero con los bloques de G y después todo el grafo. Puede haber problemas porque al colorear dos bloques B_1, B_2 no concuerden en el coloreado de su único vértice en común, pero se puede resolver, al rotar los colores en uno de los dos bloques. Por la descomposición en árbol, podemos elegir un bloque como principal (también llamado raíz) para resolver todos los conflictos de color, empezando a resolver estos problemas desde el principal hasta el resto del árbol, fijando los colores de los bloques secundarios en cada paso. Con todo ello, el número cromático del grafo será el mayor de entre los números cromáticos de los bloques. □

Por eso, si tenemos un grafo conexo, el estudio del número cromático se reduce a estudiar el número cromático de sus componentes 2-conexas.

Ya habíamos visto la cota $\chi(G) \leq \Delta(G) + 1$ como consecuencia del algoritmo voraz. También se puede ver como consecuencia del teorema 2.1, ya que $\max_H \delta(H) \leq \Delta(G)$. De todas formas, todavía podemos mejorar algo más esta cota.

Teorema 2.4. *(Teorema de Brooks). Sea G un grafo conexo y* Δ *el máximo grado de sus vértices. Si G no es un grafo completo ni tampoco un ciclo impar (de longitud impar), entonces* $\chi(G) \leq \Delta$.

Demostración. Por el lema anterior, podemos suponer sin pérdida de generalidad que G es 2-conexo. En primer lugar, supongamos que no es Δ-regular. Sea $v \in V(G)$ tal que $d(v) < \Delta$, entonces situamos el nodo al final del algoritmo voraz ($v := v_n$). Al colorear G, cuando se llega a v_n, se tiene libre por lo menos uno de los Δ primeros colores, ya que tiene $Pred(v_n) = d(v) < \Delta$, por lo tanto G admite Δ-coloración si y solo si $G - \{v_n\}$ admite Δ-coloración. Puede ser que el grafo $G - \{v_n\}$ sea $\Delta(G - \{v_n\})$-regular. En caso de que $G - \{v_n\}$ no lo fuera, procederíamos a hacer lo mismo que antes sobre este grafo, buscando un vértice $u \in V(G - \{v\})$ tal que $d(u) < \Delta(G - \{v\})$. Repetimos eso hasta que se obtenga un grafo regular. Sean $v_k, ..., v_n$ los vértices que se han eliminado hasta obtener un grafo regular. Se tiene que G admite Δ-coloración si y solo si $G - \{v_k, ..., v_n\}$ admite Δ-coloración. Por tanto, reducimos el problema a grafos regulares.

Así, supongamos que G es Δ-regular. Además, podemos suponer que $\Delta \geq 3$ ya que un grafo 2-regular conexo es un ciclo, y solo tendría número cromático 3 si fuera impar.

Si G es 3-conexo, sea $v_n \in V$ y sean $v_1, v_2 \in \Gamma(v_n)$ no adyacentes. Estos vértices existen por ser G regular y no completo.

Si G no es 3-conexo, sea $v_n \in V$ tal que $G - \{v_n\}$ sea conexo pero no 2-conexo, entonces tiene como poco dos bloques. Como G es 2-conexo, cada bloque tendrá un vértice adyacente a v_n. Sean pues $v_1, v_2 \in V$ dos de esos vértices que pertenezcan a distintos bloques de $G - \{v_n\}$. Notar que por ser v_1, v_2 de distintos bloques, $G - \{v_1, v_2\}$ es conexo.

En cualquier caso, tenemos $v_1, v_2, v_n \in V$ tales que $G - \{v_1, v_2\}$ es conexo, $(v_1, v_2) \notin E$, $(v_1, v_n) \in E$ y $(v_2, v_n) \in E$. Sea ahora $v_{n-1} \in V - \{v_1, v_2, v_n\}$ un vecino de v_n, sea $v_{n-2} \in V - \{v_1, v_2, v_{n-1}, v_n\}$ un vecino de v_n o v_{n-1}, y así sucesivamente vamos construyendo un orden de vértices $v_1, v_2, ..., v_n$ tal que cada vértice es vecino de al menos un vértice posterior. Entonces, coloreando los nodos con el algoritmo voraz, usaremos como mucho Δ colores ya que todos los vértices menos v_n tienen a lo sumo $\Delta - 1$ vértices que le preceden; y v_n tiene en sus Δ vecinos como máximo $\Delta - 1$ colores distintos ya que v_1 y v_2 tienen el mismo color. □

2.2. Polinomio cromático

Además de la búsqueda de la coloración óptima, hay otros asuntos importantes de coloración, como por ejemplo saber cuántos coloreados distintos hay. Esto es el concepto de polinomio cromático.

Definición. El *polinomio cromático* de un grafo G es una función que cuenta el número de k-coloreados posibles de G en función del número de colores usados k. Se denota $p_G(k)$.

Es evidente que esta función tomará valores nulos para $k < \chi(G)$. Asimismo, el número cromático $\chi(G)$ se puede ver como el mínimo número natural k tal que $p_G(x) \geq 1$. Además, $p_G(n) = n!$

Nos será útil para justificar alguna propiedad del polinomio cromático la siguiente construcción de grafos. También tendrá aplicación directa en el cálculo del número cromático.

Sea $G = (V, E)$ un grafo con dos vértices $v_1, v_2 \in V$ que no son adyacentes. Entonces, construimos desde él dos grafos. El primero, $G' = G + (v_1, v_2)$, como un grafo que tiene los mismos vértices y ejes que G, pero añadiendo el eje (v_1, v_2). El segundo, G'', se obtiene de G identificando v_1 y v_2 como un mismo vértice fusionado, $v_{1,2}$, el cual estará unido a otro vértice v_j si y solo si $(v_1, v_j) \in E$ o $(v_2, v_j) \in E$. Vemos cómo viene dada esta construcción en la figura 2.2.

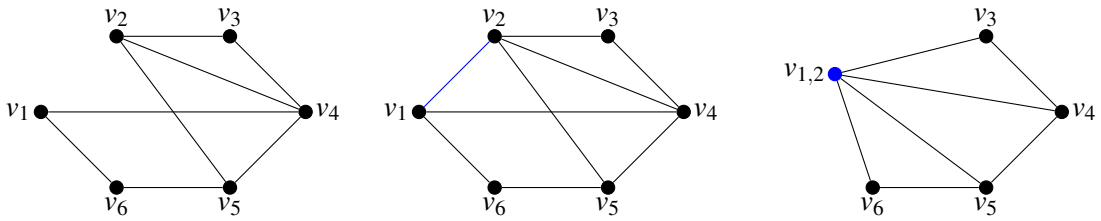

Figura 2.2: De izquierda a derecha, G, G' y G''.

Notar que se tiene un coloreado c en G tal que $c(v_1) \neq c(v_2)$ si y solo si c es un coloreado de G'; y se tiene un coloreado c en G con $c(v_1) = c(v_2)$ si y solo si c es un coloreado de G''. De esta forma, se obtiene $p_G(x) = p_{G'}(x) + p_{G''}(x)$. Igualmente, se observa que $\chi(G) = \min\{\chi(G'), \chi(G'')\}$.

Repitiendo esta simplificación encontramos un algoritmo para conseguir el número cromático.

Algoritmo de contracción

Sea G el grafo que queremos colorear, construimos una secuencia de grafos $G_0, G_1, ...$ de la siguiente manera. Definimos $G_0 = G$; entonces sea G_i, si es completo terminamos la secuencia, pero si no lo es definimos G_{i+1} como G_i' o G_i''. De esta forma, la secuencia terminará con grafo completo G_l tal que

$|G_l| = k$. Es fácil ver que en ese caso $\chi(G) \leq k$, y por la relación entre el polinomio cromático de G, G' y G'', entonces $\chi(G)$ es el máximo orden del último grafo.

Si encontrar el coloreado óptimo podía ser muy difícil, sin ninguna fórmula directa, para el polinomio cromático ocurre lo mismo (notar que si la fórmula fuera sencilla, con ella podríamos calcular $\chi(G)$ fácilmente). Continuamos con el estudio del polinomio cromático con el siguiente teorema.

Teorema 2.5. *Sea un grafo H con $n \geq 1$ vértices, m ejes y k componentes. Entonces*

$$p_H(x) = \sum_{i=0}^{n-k} (-1)^i a_i x^{n-i},$$

donde $a_0 = 1$, $a_1 = m$ y a_i son enteros positivos para todo $i = 0, 1, ..., n-k$.

Demostración. La haremos por inducción sobre $n+m$. Para $n+m = 1$ es trivial, así que hacemos el paso por inducción. Si $m = 0$, no hay ejes, por tanto $n = k$. Así, toda función $f : V(H) \to \{1, 2, ..., x\}$ es una coloración de H, por lo tanto $p_H(x) = x^n$.

Si $m > 0$, tomando dos vértices adyacentes $u, v \in V(H)$, denotamos $G = H - \{(u,v)\}$, $G' = H$ y G'' como se ha construido anteriormente, obteniéndose de H con un vértice fusión uv. Como $|E(G)| = m-1$ y $|V(G'')| + |E(G'')| \leq n - 1 + m$, por la hipótesis de inducción se cumple el resultado en $p_G(x)$ y en $p_{G''}(x)$. También notar que G'' tiene k componentes y G tiene como mínimo k, así que

$$p_G(x) = x^n - (m-1)x^{n-1} + \sum_{i=2}^{n-k} (-1)^i b_i x^{n-i},$$

donde b_i son enteros no negativos, para $0 \leq i \leq n-k$ y

$$p_{G''}(x) = x^{n-1} - \sum_{i=2}^{n-k} (-1)^i c_i x^{n-i},$$

donde c_i son enteros positivos para $0 \leq i \leq n-k$. Por lo tanto, se tiene

$$p_H(x) = p_{G'}(x) = p_G(x) - p_{G''}(x) = x^n - mx^{n-1} + \sum_{i=2}^{n-k} (-1)^i (b_i + c_i) x^{n-i},$$

donde $a_i = b_i + c_i$ son enteros positivos para $0 \leq i \leq n-k$ como en el enunciado. □

Que el polinomio cromático tiene coeficientes enteros lo podemos ver de otra manera. Sea $\pi_r(G)$ el número de particiones de $V(G)$ en r conjuntos no vacíos de vértices independientes, y sea $(x)_r = x(x-1)\cdots(x-r+1)$. Entonces para cada natural x se tiene que

$$p_G(x) = \sum_{r-1}^{n} \pi_r(G)(x)_r$$

También podemos interpretar los coeficientes del número cromático de la siguiente manera.

Teorema 2.6. *Sea un grafo H con n vértices y ejes $E = \{e_1, ..., e_m\}$. Un subconjunto de E se llama ciclo roto si se obtiene de un conjunto de ejes que forman un ciclo, eliminando el de índice mayor. Entonces el polinomio cromático de H es*

$$p_H(x) = \sum_{i=0}^{n-1} (-1)^i a_i x^{n-i}$$

donde a_i es el número de subconjuntos de E de tamaño i que no contienen ningún ciclo roto.

Demostración. También la haremos por inducción sobre m. Para $m = 0$ es trivial, así que asumimos $m \geq 1$, y hacemos el paso de inducción.

Sean $e_1 = (u,v)$ y $G = H - \{e_1\}$, así que $G' = G + \{e_1\} = H$ y G'' como se ha construido anteriormente con un vértice fusión uv. Identificamos $E(G) = \{e_2, e_3, ..., e_m\}$, y a $E(G'')$ como un subconjunto de $E(H)$ tal que los ejes de $E(G'')$ que estén en $E(H)$ conservan su notación, y si un eje viene de dos vértices distintos de $E(H)$, siendo de la forma (uv,w) para un vértice $w \in V(G'')$ con $e_h = (u,w)$ y $e_i = (v,w)$, lo denotamos como $e_k = (uv,w)$ con $k = \text{máx}\{i,h\}$.

Tenemos ahora que probar que el número de subconjuntos de $E(G')$ de tamaño i que no contienen un ciclo roto de G' es la suma del número de subconjuntos de $E(G)$ de tamaño i que no contienen un ciclo roto de G y del número de subconjuntos de $E(G'')$ de tamaño i que no contienen un ciclo roto de G'', pero esto es una consecuencia de las siguientes afirmaciones:

1. Suponiendo que $e_1 \notin F \subseteq E(G')$, entonces F no contiene ningún ciclo roto de G' si y solo si F no contiene ciclos rotos de G.
2. Suponiendo que $e_1 \in F \subseteq E(G')$, entonces F no contiene ningún ciclo roto de G' si $F - \{e_1\} \subseteq E(G'')$ y $F - \{e_1\}$ no contiene ningún ciclo roto de G''. □

Como consecuencia vemos que el número de subconjuntos de E de tamaño i que no contienen ningún ciclo roto es independiente del orden impuesto en E, resultado que no es evidente.

Este teorema no es muy práctico. Sin embargo, si el grafo no tiene ciclos cortos, entonces podemos conocer los primeros coeficientes. Terminamos la sección con este corolario, aunque antes introducimos la noción de *cintura* de un grafo, que es la longitud del ciclo más corto contenido en dicho grafo.

Corolario 2.7. *Sea G un grafo con n vértices, m ejes, cintura g y polinomio cromático*

$$p_G(x) = \sum_{i=0}^{n} (-1)^i a_i x^{n-i}$$

Entonces $a_i = \binom{m}{i}$ *para* $i \leq g-2$. *Además, si g es finita y G tiene* c_g *ciclos de longitud g, entonces* $a_{g-1} = \binom{m}{g-1} - c_g$. □

Capítulo 3

Complejidad computacional y algoritmos heurísticos

Hasta ahora se han dado cotas o aproximaciones del número cromático. Veremos ahora que nos vamos a tener que conformar con eso, ya que nos encontramos ante un problema **NP-completo**. Explicamos la complejidad de los problemas en este primer punto.

3.1. Complejidad computacional. La clase de problemas NP-Hard

Podremos dar solución a un problema con distintos métodos de resolución, pero tendremos alguno que será más rápido que otro o que requiera de menos pasos. En esta sección formalizamos estas ideas.

Definición. Un *algoritmo* es un conjunto finito ordenado de instrucciones usadas para resolver un problema. Los algoritmos requieren de un *input* (entrada), que será la información o estado inicial alrededor del problema, y, tras ejecutar las instrucciones, ofrecen un *output* (salida) con el resultado obtenido.

La complejidad computacional trata de responder la pregunta: "Dado un problema X, ¿cuánto cuesta resolver X usando un algoritmo A?". En nuestro trabajo, mediremos el tiempo de resolución con pasos, considerando un paso cada una de las sumas, restas, multiplicaciones, divisiones o comparaciones entre dos números que hagamos (las operaciones elementales). El input normalmente será la información de un grafo $G = (V,E)$, con $n = |V|$ y $m = |E|$, y la expresión de la complejidad del algoritmo vendrá dada por una función que depende de n y m.

En muchas ocasiones expresaremos las complejidades de manera asintótica, así que las constantes y términos dominados por otros términos no serán tan importantes. Introducimos así la notación: $f(x) = O(g(x))$ si existen unas constantes reales $c > 0$ y K tales que $|f(x)| \leq cg(x)$ para todo $x \geq K$.

Definición. Diremos que un problema se puede resolver en *tiempo polinomial* si el número de pasos que necesita para ser resuelto con un algoritmo está acotado por un polinomio que depende del tamaño de la entrada, es decir, siendo x el tamaño de la entrada, si el problema se puede resolver en tiempo $O(x^k)$ para alguna constante k.

Definición. Dados P y P' dos problemas y x una entrada para P, decimos que P *es reducible a* P' *en tiempo polinomial*, si x se puede reformular mediante una transformación en tiempo polinomial, de manera que tengamos los datos de entrada x' del problema P', tal que la solución de P' con la entrada x' nos dé una solución de P para su entrada x.

Hay distintos tipos de problemas, y así los clasificamos basándonos en su dificultad. Tendremos en cuenta el tiempo que tarda en resolver el problema el mejor algoritmo que haya en el peor caso posible.

Definición. La clase **P** consta de los problemas que pueden ser resueltos en un tiempo polinomial.

Definición. Los problemas de clase **NP** contiene a los problemas que una máquina no determinista puede resolver en un tiempo polinomial.

También se nos puede dar una solución y que la tengamos que comprobar, o que se nos pregunte si el problema cumple una propiedad. Por ejemplo, si dado un grafo con una coloración, saber si es óptima. Esto es lo que se denomina problemas de decisión.

Definición. Los *problemas de decisión* son aquellos que dada una entrada tenemos que responder "SÍ" o "NO", en función de si la entrada satisface algún propiedad o no.

Con esta definición redefiniremos los problemas de clase **NP**, y además introduciremos nuevas clases de problemas.

Definición. La clase **NP** de problemas de decisión consta de los problemas para los que si tenemos una respuesta "SÍ", se puede comprobar en tiempo polinomial.

Definición. La clase **NP-completa** de problemas de decisión contiene a los problemas L tales que cualquier problema $L' \in$ **NP** es reducible a L en tiempo polinomial.

Esto es, un problema de decisión P es **NP-completo** si pertenece a la clase **NP** y además cualquier otro problema de decisión $P' \in$ **NP** se puede reducir a él en tiempo polinomial, es decir, si P fuese polinomial entonces P' sería polinomial para todo $P' \in$ **NP**.

Definición. La clase **NP-hard** contiene a los problemas de optimización tales que su versión de decisión es **NP-completa**.

Ejemplo 6. El problema del máximo subgrafo completo de un grafo es **NP-hard**, y por la relación de antes, el problema del máximo conjunto independiente de un grafo también. De hecho, los problemas del primer capítulo son todos **NP-hard**.

El estudio de estos problemas, por tanto, no nos da ventajas con respecto al problema de coloreado de grafos.

Teorema 3.1. *Decidir si un grafo G se puede colorear con k colores es **NP-completo** para cualquier $k \geq 3$. Por lo tanto, el problema de hallar una coloración óptima de un grafo es un problema **NP-hard**.*

Demostración. La prueba se puede ver en Garey y Johnson (1979).

3.2. Algoritmos heurísticos

En esta sección buscaremos algoritmos que permitan dar el valor más cercano posible para el número cromático. Por ser un problema **NP-hard** lo más seguro es que no haya un algoritmo óptimo que nos ofrezca siempre la mejor solución en un tiempo polinomial. No nos queda otra que probar *algoritmos heurísticos*, es decir, algoritmos que debido a la dificultad del problema descartan encontrar soluciones óptimas, encontrándose con valores lo más cercanos posibles a la solución, a pesar de que el valor encontrado pudiera no ser muy bueno en algunos casos.

En el anterior capítulo se han introducido ya dos algoritmos para la búsqueda del número cromático, el algoritmo voraz y el algoritmo de contracción.

El primer algoritmo que introducimos ahora es parecido al algoritmo voraz. El segundo, relacionado con la búsqueda de conjuntos independientes maximales.

Algoritmo DSatur

Sea $G = (V,E)$ un grafo. Vamos a colorear los vértices de G de uno en uno. Tomamos en cada paso el vértice v sin colorear tal que tiene el mayor número de colores distintos en sus vecinos (este número se denomina *grado de saturación* de v), y lo coloreamos con el color disponible más pequeño no usado por sus vecinos, añadiendo un nuevo color si fuera necesario. En caso de empate (que varios vértices tengan en algún paso el mismo grado de saturación y este sea el máximo de entre los nodos sin pintar), se elige entre estos el vértice de mayor grado en el subgrafo inducido por los vértices sin colorear.

En el peor caso, la complejidad del algoritmo es de $O(n^2)$, ya que el proceso de selección del nuevo vértice lleva un tiempo $O(n)$ y hemos de hacer este proceso n veces.

Algoritmo RLF

Dado $G = (V,E)$ un grafo, con el algoritmo recursivo más grande primero (en inglés, recursive largest first algorithm, con siglas RLF) se obtiene una partición $\mathscr{S}$ de V donde a cada conjunto se le asocia un color distinto.

Sea inicialmente $\mathscr{S} = \emptyset$. Primero, en cada paso identificamos un conjunto independiente maximal $S \subseteq V$. Lo hacemos en primer lugar tomando en S el vértice de G que tenga el mayor grado. Entonces, agregamos a S los vértices tales que no son adyacentes a ningún vértice en S, y tienen grado máximo de entre los vértices adyacentes a los vértices en S. Si hay empate en la última condición se puede romper seleccionando el vértice con el menor número de vecinos que no están en S.

Después añadimos S a $\mathscr{S}$ y finalmente eliminamos los vértices de S de G. Repetimos los dos últimos párrafos hasta que G esté vacío.

En Lewis (2021) se comparan empíricamente estos dos algoritmos, además del voraz, obteniendo las mejores coloraciones con el algoritmo RLF, aunque los tiempos de ejecución eran mayores en este método por ser más complejo ($O(n+m)$ para el algoritmo voraz y $O((n+m)\log n)$ para el algoritmo DSatur).

3.3. Resultados de complejidad

Sabemos ya varias cotas, y los algoritmos nos pueden dar algún valor aproximado para el número cromático, pero, ¿serán suficientemente buenos estos valores?

Existen diversos estudios relacionados con este tema. Garey y Johnson (1976) muestran que obtener coloraciones usando $s\chi(G)$ colores, con $s < 2$, es **NP-hard**. Por otro lado, Lund y Yannakakis (1994) ven que $\chi(G)$ es difícil de aproximar, fijado un $\varepsilon > 0$, para valores más pequeños de n^{ε}. También dicen que dado un grafo G, para cualquier constante h existe un entero k_h tal que si G que es k_h-coloreable, entonces es **NP-hard** encontrar una coloración de $h\,k_h$ colores.

Capítulo 4

Grafos de intervalos y grafos cordales

Ya hemos incluido en la introducción los conceptos de grafo de intervalos y grafo cordal. Es interesante detenernos a estudiar estos tipos de grafos debido a que tienen métodos de resolución de coloreado propios. Estudiaremos en esta sección algoritmos y propiedades directamente relacionadas con ellos.

4.1. Orden de eliminación perfecto

Definición. Sea G un grafo no dirigido y $\{v_1, ..., v_n\}$ una ordenación de los vértices.

- Decimos que *dirigimos los ejes de izquierda a derecha* si para todo $(v_i, v_j) \in V$ dirigimos el eje tal que $v_i \to v_j$ si $i < j$, y $v_j \to v_i$ si $i > j$.
- Si $v_i \to v_j$, entonces v_i es un *predecesor* de v_j, y v_j es un *sucesor* de v_i.
- Denotamos al conjunto de predecesores de $v \in V$ por $Pred(v)$.
- Denotamos por $indeg(v)$ al número de predecesores de $v \in V$.

Un cierto orden de los vértices que buscaremos en muchas ocasiones será el siguiente.

Definición. Un *orden de eliminación perfecto* (o p.e.o.) es una ordenación de los vértices $\{v_1, ..., v_n\}$ tal que $Pred(v_i)$ es un clique para todo $i = 1, ..., n$.

Esta ordenación de los vértices es especial para el coloreado, como vemos en el siguiente resultado.

Teorema 4.1. *Si tenemos un p.e.o. para un grafo G, al aplicar el algoritmo voraz obtenemos una coloración con $\chi(G)$ colores.*

Demostración. Sabemos que el algoritmo voraz, aplicado a un p.e.o., encuentra el coloreado con máximo $\max_{i \in \{1,...,n\}} \{indeg(v_i) + 1\}$ colores. Sea v_i^* ese vértice, así que $\chi(G) \leq indeg(v_i^*) + 1$.

Por ser un p.e.o., se tiene que $Pred(v_i^*)$ es un clique, por tanto $\{v_i^*\} \cup Pred(v_i^*)$ también es un clique, y de esta manera $\omega(G) \geq indeg(v_i^*) + 1$.

Pero sabemos que $\chi(G) \geq \omega(G)$, así que $\chi(G) = \omega(G) = indeg(v_i^*) + 1$, y por lo tanto se obtiene un coloreado óptimo. □

Con esta demostración también se encuentra el máximo subgrafo completo $\omega(G) = \chi(G)$.

Ni todas las ordenaciones de nodos son perfectas ni todos los grafos tienen una. Serán de nuestro interés los grafos que sí la tengan.

Teorema 4.2. *Todo grafo de intervalos G tiene un p.e.o.*

Demostración. Lo podemos originar ordenando los intervalos según su punto inicial en orden de izquierda a derecha. De esta forma, para un vértice v_i cuyo intervalo correspondiente empieza en el punto s_i, se tiene que $Pred(v_i)$ es el conjunto de los vértices correspondientes a los intervalos que empiezan antes que s_i y terminan después de s_i. Como todos los vértices contienen al punto s_i, todos los predecesores de v_i están conectados y $Pred(v_i)$ forma un clique. □

Los grafos cordales también tienen un orden de eliminación perfecto, y es más, todos los grafos que tienen un p.e.o. son cordales. Veremos esto más adelante. De momento, vemos la primera implicación.

Teorema 4.3. *Si un grafo G tiene p.e.o entonces G es cordal.*

Demostración. Supongamos que tenemos un grafo que no es cordal. Sea v el vértice en un ciclo que aparece el último en el p.e.o. Tiene como poco 2 predecesores por estar en el ciclo, y esos vértices formarán un clique por ser p.e.o. Sin embargo, estos vértices no pueden estar unidos porque no es cordal y tiene k-ciclos para $k \geq 4$, lo cual es una contradicción. □

Las siguientes definiciones serán fundamentales para ver la otra implicación.

Definición. Sea un grafo $G = (V, E)$:

- Un *separador* es una partición de los vértices $V = S \cup A \cup B$ tal que no hay ejes entre A y B.
- Dados $a, b \in V$ no adyacentes, se dice (a,b)-*separador* a un separador $V = S \cup A \cup B$ tal que $a \in A$, $b \in B$.
- Un (a,b)-*separador mínimo* es un (a,b)-separador tal que ningún subconjunto de S es (a,b)-separador.
- Un vértice v es *simplicial* si los vecinos de v forman un clique en V.

Claramente siempre habrá un separador. No hace falta especificar A y B en un (a,b)-separador. Notar que cualquier conjunto S puede expandirse a (a,b)-separador. Con estos conceptos ya estamos en condiciones de ver el siguiente resultado.

Teorema 4.4. *Todo grafo cordal tiene p.e.o.*

Demostración. En primer lugar, notar que si le quitamos un vértice a un grafo cordal sigue siendo cordal (sigue sin tener k-ciclos, $k \geq 4$). Además, en un p.e.o. el último vértice es simplicial, ya que sus vecinos (sus predecesores) forman un clique. Así, bastará probar que todo grafo cordal tiene un vértice simplicial.

Suponemos sin pérdida de generalidad que el grafo es conexo. Procederemos por inducción sobre n. Para $n = 1$, el vértice de G es simplicial, así que asumimos $n \geq 2$. Si G es completo entonces todos los vértices son simplicial y está hecho, así que pensamos en G no completo, y de esta forma tenemos $a, b \in V$ tal que $(a,b) \notin E$.

Sea S el (a,b)-separador mínimo, y sea G_T el grafo inducido por un conjunto de vértices $T \subseteq V$. Entonces G_{V-S} no es conexo y tiene varias componentes conexas, una que contiene a a (llamémosle A), otra a b (la llamaremos B) y puede tener otras, como vemos en figura 4.1.

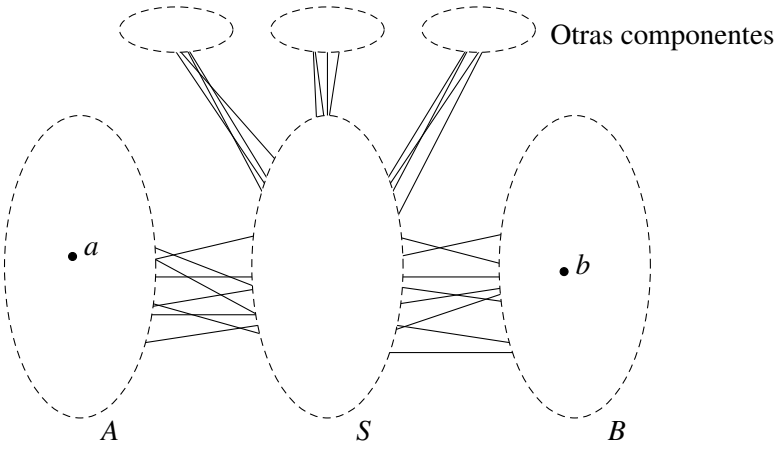

Figura 4.1: Componentes conexas con el (a,b)-separador minimal.

El problema por tanto trata de encontrar un vértice simplicial en A y otro en B, claramente no adyacentes. Además, con probar que hay un simplicial en A es suficiente, el otro resultado es similar.

Si $G_{A\cup S}$ es completo, entonces a simplicial en $G_{A\cup S}$, lo cual implica que a es simplicial en G, ya que todos los vecinos de a en G pertenecen a $A\cup S$, y por tanto tenemos el resultado. Por tanto, sea $G_{A\cup S}$ no completo. Aplicamos ahora la inducción.

$G_{A\cup S}\subset G$ ya que $b\notin A\cup S$, por tanto, al ser un grafo más pequeño, por hipótesis de inducción se tiene que hay dos vértices simplicial $x,y\in G_{A\cup S}$. Si $x\in A$, entonces x simplicial en G también ya que todos los vecinos de x en G están en $A\cup S$. De igual manera si $y\in B$, así que nos falta ver el caso de que $x,y,\in S$. De hecho, vamos a demostrar que ese caso no puede ocurrir.

x tiene un vecino a_x en A, porque si no $S-\{x\}$ sería un (a,b)-separador y S no sería minimal. De igual manera y tiene un vecino a_y en A. Como G_A es conexo, existe un camino de a_x a a_y con vértices en A. De esta forma, existe un camino de x a y con vértices intermedios en A, sea el camino más corto $x-a_1-a_2-\cdots-a_k-y$, con longitud mayor que 2 por ser x e y no adyacentes. De manera similar encontramos un camino en B que une x con y. Vemos esto en la figura 4.2.

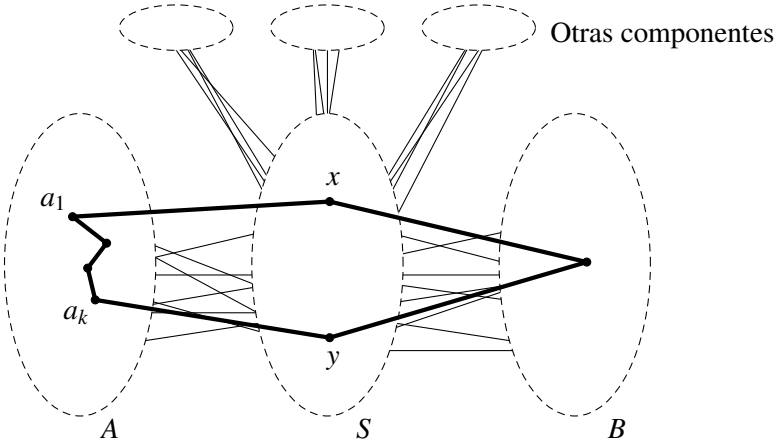

Figura 4.2: Un camino que une x e y pasando por A, y otro por B.

Combinando estos caminos obtenemos un ciclo de longitud como mínimo 4, y por ser G un grafo cordal tendrá que tener una cuerda. Por haber tomado los caminos más cortos, no hay ninguna cuerda en el camino de x a y con nodos de A o de B. Tampoco hay conexiones entre nodos de A y B; por lo tanto, la cuerda que necesitamos sería (x,y), pero no son adyacentes, lo cual significa que es una contradicción que $x,y\in S$ a la vez. □

Como consecuencia del teorema vemos nuevamente que todos los grafos de intervalos son grafos cordales. Notar también que el final de la demostración muestra que cualquier (a,b)-separador minimal en un grafo cordal es un clique.

4.2. Reconocimiento de los grafos de intervalos

Empecemos viendo una caracterización de los grafos de intervalos relacionada con cliques maximales. Estos son cliques que no son subconjuntos de otro clique de G.

Teorema 4.5. *Un grafo G es de intervalos si y solo si los cliques maximales de G pueden ser ordenados consecutivamente, es decir, ordenados como $C_1,...,C_k$ de forma que si $v\in C_i$ y $v\in C_j$, entonces $v\in C_h$, para todo $i<h<j$.*

Demostración. Supongamos que G es un grafo de intervalos, y elegimos una representación tal que todos los intervalos tengan puntos iniciales disjuntos. Como ya sabemos, si ordenamos los intervalos de izquierda a derecha por el punto en el que empiezan tenemos un p.e.o., $v_1,...,v_n$, y que todos los cliques maximales son de la forma $Pred(v)\cup\{v\}$, para algún $v\in V$. Si para $v_i\in V$, $Pred(v_i)\cup\{v_i\}$ es clique maximal, entonces le asociamos i al clique maximal. Después ordenamos los cliques maximales por su entero, y los denotamos $C_1,...,C_k$. Vemos esto en la figura 4.3.

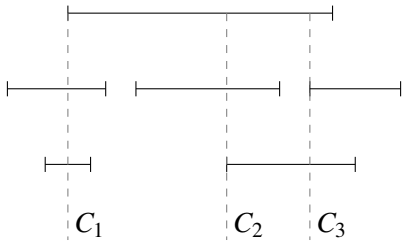

Figura 4.3: Ordenación de los cliques maximales de izquierda a derecha.

Notar ahora que si $v \in V$ pertenece a C_i y a C_j, con $i < j$, entonces I_v interseca a todos los intervalos correspondientes a los nodos de C_i y de C_j, por lo tanto también intersecará a todos los intervalos de los nodos de los cliques entre C_i y C_j, lo que significa que pertenece a esos cliques también, así que es un orden consecutivo de los cliques maximales.

Para la otra implicación, sea $C_1, ..., C_k$ un orden consecutivo de los cliques maximales, entonces para todo $v \in V$ sea $I_v = [\text{mín}\{i : v \in C_i\}, \text{máx}\{j : v \in C_j\}]$; necesitamos ahora verificar que estos intervalos son una representación de G.

Supongamos que $(u,v) \in E$, entonces existe un clique maximal que contiene al eje, sea C_h, por tanto $u \in C_h$ y $v \in C_h$. Por la definición de los intervalos, I_u y I_v contienen a h, por tanto intersecan ($I_u \cap I_v \neq \emptyset$). Por el otro lado, asumimos que $I_u \cap I_v \neq \emptyset$ y que ambos contienen a un entero h. Por tener un orden consecutivo de cliques, entonces $u, v \in C_h$, lo que significa que $(u,v) \in E$. □

En la demostración tenemos un método de reconocimiento de grafos de intervalos, que se divide en dos partes: la primera encontrar todos los cliques maximales, y la segunda probar si los cliques pueden ser ordenados adecuadamente.

En cuanto a encontrar todos los cliques maximales, el problema es muy difícil. Es más difícil que un problema **NP**. Sin embargo, como lo que queremos es encontrar los cliques maximales de un grafo de intervalos, asumiremos que el grafo es cordal (notar que si no lo es, entonces no es de intervalos). Por tanto, sabemos que los cliques maximales son de la forma $Pred(v) \cup \{v\}$, después de un p.e.o. Como máximo pues tendremos n cliques maximales. Veamos la siguiente caracterización.

Lema 4.6. *Sea $v_1, ..., v_n$ un p.e.o. Entonces $C = Pred(v_i) \cup \{v_i\}$ no es un clique maximal si y solo si existe un sucesor v_j de v_i tal que v_i es el último predecesor de v_j e $indeg(v_j) = indeg(v_i) + 1$.*

Demostración. Supongamos que existe un sucesor v_j de v_i. Como v_i es el último predecesor de v_j, todos los predecesores de v_j son v_i o predecesores de v_i, por tanto $Pred(v_j) \subseteq Pred(v_i) \cup \{v_i\} = C$. Como $indeg(v_j) = indeg(v_i) + 1$, se tiene la igualdad, y como v_j es adyacente a todos los vértices en C, entonces $C \cup \{v_j\}$ es un clique más grande.

Para la otra implicación, supongamos que C no es maximal, y sea j el mínimo valor tal que $v_j \notin C$ y $C \cup \{v_j\}$ es clique. v_j no es predecesor de v_i ya que si no estaría en C, así que es sucesor, por tanto $C \subseteq Pred(v_j)$ y entonces $indeg(v_j) \geq indeg(v_i) + 1$.

Supongamos que v_i no es el último predecesor de v_j, entonces v_j tiene un predecesor v_k tal que $i < k < j$, pero como v_k es adyacente a C se tendría que $C \cup \{v_k\}$ es clique, en contradicción con la minimalidad de j. Por tanto, v_i es el último predecesor de v_j; y como además todo predecesor de v_j lo es también de v_i, se tiene que $Pred(v_j) \subseteq C$ y $indeg(v_j) \leq indeg(v_i) + 1$. Esto prueba el resultado. □

4.3. Reconocimiento de los grafos cordales

Rose, Tarjan y Lueker (1976) (ver también Habib, McConnell, Paul y Viennot (2000)) muestran que un p.e.o. se puede encontrar utilizando un algoritmo conocido como búsqueda lexicográfica en amplitud (*Lex-BFS*, en inglés lexicographic breadth-first search).

Algoritmo Lex-BFS

Sea $G = (V, E)$ un grafo. Sea una lista de conjuntos Σ, que inicialmente consta de un solo conjunto, V. También se tiene una lista $\mathscr{V}$, en un principio vacía, que al final del algoritmo será un orden de los vértices de G.

El algoritmo elige paso a paso un vértice v del primer conjunto de Σ, entonces lo elimina de ese primer conjunto y lo añade al final de $\mathscr{V}$. En cada paso, todo $S \in \Sigma$ se divide en dos subconjuntos más pequeños, el primero formado por los vecinos de v en S y el segundo formado por los que no son vecinos de v en S. Realizamos este proceso de división hasta acabar con todos los vértices. Cada vez que se tenga un conjunto vacío se elimina de la lista.

Finalmente, si para todo $v \in V$ tomamos su predecesor w más cercano con el orden obtenido, entonces si para algún v se tiene que $Pred(v) - \{w\} \not\subseteq Pred(w)$, el grafo G no es cordal. En caso contrario, el grafo es cordal y el orden de vértices obtenido es un p.e.o.

Dado que tanto el algoritmo Lex-BFS como el proceso de probar si una ordenación de vértices es un p.e.o. se pueden realizar en tiempo lineal, es posible reconocer grafos cordales en tiempo lineal. Veamos un ejemplo de aplicación de este algoritmo.

Ejemplo 7. Sea G el grafo de la figura 4.4, veamos si es cordal.

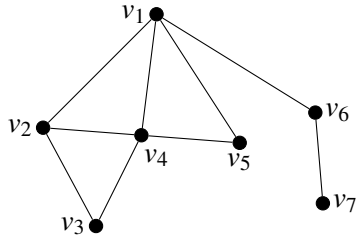

Figura 4.4: Aplicación del algoritmo Lex-BFS sobre G para reconocimiento de un grafo cordal.

En primer lugar, tenemos $\Sigma = \{\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, \}\}$ y $\mathscr{V} = \emptyset$.

Podemos tomar en el primer paso cualquier nodo, empecemos por ejemplo por v_1, incluyéndolo en $\mathscr{V}$ y separando sus vecinos y no vecinos en dos conjuntos; así se tiene $\Sigma = \{\{v_2, v_4, v_5, v_6\}, \{v_3, v_7\}\}$ y $\mathscr{V} = \{v_1\}$.

En el segundo paso, podemos tomar cualquiera de los nodos del primer conjunto, v_2, v_4, v_5, v_6, sea por ejemplo v_2. El vecino de v_2 que está en el primer conjunto es v_4, y en el segundo conjunto está v_3, así que se tiene $\Sigma = \{\{v_4\}, \{v_5, v_6\}, \{v_3\}, \{v_7\}\}$ y $\mathscr{V} = \{v_1, v_2\}$.

Ahora tomamos v_4, y por ser vecino de v_5 se tiene $\Sigma = \{\{v_5\}, \{v_6\}, \{v_3\}, \{v_7\}\}$ y $\mathscr{V} = \{v_1, v_2, v_4\}$.

Podríamos seguir uno a uno con los conjuntos de Σ, pero como son conjuntos de un solo vértice, podemos ponerlos en orden en $\mathscr{V}$, y así tener la ordenación de vértices $\{v_1, v_2, v_4, v_5, v_6, v_3, v_7\}$.

Comprobamos que $Pred(v) - \{w\} \not\subseteq Pred(w)$ para todo $v \in V$ y w su predecesor más cercano,

$$Pred(v_7) - \{v_6\} = \emptyset \subseteq Pred(v_6) = \{v_1\}$$
$$Pred(v_3) - \{v_4\} = \{v_2\} \subseteq Pred(v_4) = \{v_1, v_2\}$$
$$Pred(v_6) - \{v_1\} = \emptyset = Pred(v_1)$$
$$Pred(v_5) - \{v_4\} = \{v_1\} \subseteq Pred(v_4) = \{v_1, v_2\}$$
$$Pred(v_4) - \{v_2\} = \{v_1\} = Pred(v_2)$$
$$Pred(v_2) - \{v_1\} = \emptyset = Pred(v_1),$$

por tanto, el grafo G es cordal.

Bibliografía

[1] ASPNES, J. (2020). *Notes on Computational Complexity Theory - CPSC 468/568*. Yale University.

[2] BIEDL, T. (2005) *CS 762: Graph-theoretic algorithms. Lecture notes of a graduate course*. University of Waterloo.

[3] BOLLOBÁS, B. (1998). *Modern graph theory*. Springer.

[4] BONDY, J. A., & MURTY, U. S. R. (2008). *Graph Theory*. Springer.

[5] BUTENKO, S. (2003). *Maximum Independent Set and Related Problems, with Applications*. University of Florida.

[6] CHAN, B. (2010, 18 de marzo). *Nonseparable Graphs*. Extraído el 31 de mayo de 2022 desde `https://www.math.hkust.edu.hk/~mabfchen/Math4821B/Nonseparable%20Graphs.pdf`.

[7] GARCÍA, A. *Teoría de grafos*. Universidad de Zaragoza.

[8] GAREY, M. R., & JOHNSON, D. S. (1976). The complexity of near-optimal coloring. *Journal of the ACM*, 23(1), 43–49. `https://dl.acm.org/doi/10.1145/321921.321926`.

[9] GAREY, M. R., & JOHNSON, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[10] GONDRAM, M., & MINOUX, M. (1984). *Graphs and Algorithms* (S. Vajda, Trad.). Wiley.

[11] HABIB, M.; MCCONNELL, R.; PAUL, C.; & VIENNOT, L. (2000). Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition, and consecutive ones testing. *Theoretical Computer Science*, 234(1–2), 59–84. `https://doi.org/10.1016/S0304-3975(97)00241-7`.

[12] KHANNA, S., LINIAL N., & SAFRA, S. (2000). On the Hardness of Approximating the Chromatic Number. *Combinatorica*, 20(3), 393-415. `https://doi.org/10.1007/s004930070013`.

[13] LEWIS, R. M. R. (2016). *A Guide to Graph Colouring Algorithms and Applications*. Springer.

[14] LUND, C., & YANNAKAKIS, M. (1994). On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5), 960–981. `https://dl.acm.org/doi/10.1145/185675.306789`.

[15] ROSE, D. J., TARJAN, R. E., & LUEKER, G.S. (1976). Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2), 266–283. `https://doi.org/10.1137/0205021`.