UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Matemática

Tesis de Licenciatura

ESTADO DEL ARTE SOBRE EL PROBLEMA DE COLOREO DE GRAFOS Y SUS VARIANTES

Franco Cerisola

Director: Dr. Guillermo A. Durán

2019-05-23

Agradecimientos

A mi familia por todo el apoyo que me dieron, sin ellos me hubiera sido imposible terminar la carrera de matemática.

A mi profesor Guillermo Duran (*Willy*) por haber aceptado ser mi director de tesis y por haber sabido transmitirme interés por la investigación operativa desde la primer clase, convirtiéndose de inmediato en mi materia preferida. Le agradezco tanto por sus cualidades profesionales como también humanas.

A mis amigos de la facultad, los seis integradas del excéntrico grupo de *WhatsApp* conocido como *"La Banda de Aplicada"*:

- El piloto de Copa Airlines Iván Muñoz con quien hice todos los TPs de la carrera. Nuestro motto: *¿por qué no compila?*
- El jugador de rugby, músico y escritor Joaquín del Priore, con sus excelentes recomendaciones de libros y demás. Siempre un placer.
- Su querida amiga Natalia Regalado, que me salvó con sus apuntes de ecuaciones diferenciales entre tantas otras materias.
- El programador de árbitros de basketball Facundo Gutierrez.
- Por ultimo pero no menos importante, el magnifico dúo de los estadistas Ignacio Franco y Nicolás Murrone.

A todos ellos les estoy eternamente agradecido por su ayuda en el estudio y su agradable compañía durante todas las largas horas de cursada.

Al Instituto del Cálculo que en más de una oportunidad puso a nuestra disposición sus instalaciones y sus docentes para sernos de ayuda.

Índice general

1. Introducción **6**
1.1. Algunas notas históricas 6

2. Definiciones y Conceptos Generales **8**
2.1. Definiciones 8
2.2. Caminos y Ciclos 12
2.3. Grafos Conexos 12
2.4. Grafos Bipartitos 14
2.5. Representación de Grafos 16
2.6. Clases de Complejidad 18

3. Coloreo Clásico **21**
3.1. Definiciones 22
3.2. Aplicación: *Register allocation* 23
3.3. Cotas para el número cromático 25
3.4. ¿Cuán lejos puede estar χ(G) de ω(G)? 28
3.5. Grafos Perfectos 31
3.6. Conjeturas de Berge 31
3.7. Reconocimiento de Grafos Berge 36
3.8. Grafos Color-Crítico 40
3.9. Grafos Planares 43
3.10. Coloreo de Aristas 45
3.11. Relación con Matching 46
3.12. Teorema de Vizing 46
3.13. Aplicación para un problema de Scheduling 49
3.14. Coloreo Total 51

4. Variantes del Problema de Coloreo **53**
4.1. Coloreo por Listas 54
4.2. μ-coloreo 54
4.3. (γ, μ)-coloreo 56
4.4. Pre-coloreo 56
4.5. Resolución del juego Sudoku con pre-coloreo 57

4.6. Comparación entre los distintos problemas 59
4.7. Comentarios sobre la complejidad 59
4.8. Resumen de los resultados 64
4.9. Número de elección 65
4.10. Coloreo Suma 67
4.11. Coloreo de aristas por listas 68

5. Algoritmos de Coloreo **69**
5.1. Algoritmo de Conexión-Contracción para calcular $\chi(G)$ . . . 69
5.2. Polinomio Cromático 72
5.3. Polinomio cromático de un grafo ciclo 73
5.4. Coloreo Exacto con Programación Lineal 75
5.5. Un algoritmo 'greedy' para coloreo de vértices 80
5.6. Teorema de Brooks 83
5.7. Variantes del Algoritmo greedy 85
5.7.1. Método LF 85
5.7.2. Método SL 85
5.7.3. Método DSATUR 86
5.7.4. Otros métodos para seleccionar los vértices 89
5.8. Comparación entre los algoritmos de tipo RS 90
5.9. Algoritmo greedy para μ-coloreo de cografos 92
5.10. Coloreo Equitativo exacto con DSATUR 94
5.11. Algoritmo EqDSATUR 99
5.12. (γ, μ)-coloreo en grafos bipartitos completos 102
5.13. Complejidad 103
5.14. Análisis del Peor Caso 103

6. Conclusiones y problemas abiertos **104**

Resumen

Un grafo es una estructura que sirve para representar un conjunto de objetos y relaciones que existen entre ellos. Esta formulación abstracta y de carácter general resulta particularmente útil para modelar una amplia variedad de problemas que en principio parecen ser de naturaleza bien diferente.

Existen numerosas cuestiones relacionadas al estudio de los grafos, en este trabajo nos concentramos en un área denominada coloreo de grafos. La elección del nombre proviene tanto de motivos históricos relacionados al problema original como también por razones de representación gráfica a modo didáctico. Pero a pesar del término, el proceso de coloreo se trata de realizar un etiquetado de los elementos del grafo (vértices o aristas) para poder así agruparlos en distintas clases que permitan cumplir con ciertas restricciones. Llamaremos coloreo clásico a la formulación original y más sencilla de este problema, pero veremos también variantes del problema de coloreo que permiten tratar cuestiones más complejas.

El problema de coloreo es también interesante de un punto de vista algorítmico debido a que se trata de un problema NP-Completo, es decir que obtener una solución exacta requiere de demasiado tiempo de cómputo, independientemente de cuan poderosa sea la computadora utilizada. Como consecuencia se estudian tanto técnicas para reducir la velocidad de cómputo como también métodos para obtener soluciones sub-óptimas.

El objetivo de este trabajo es hacer una recopilación del los principales resultados en cada una de estas áreas para mostrar cual es el estado actual del desarrollo en este campo.

Capítulo 1

Introducción

Coloreo es una de las áreas más estudiadas de teoría de grafos. Desde sus orígenes ha capturado el interés de los investigadores tanto de un punto de vista teórico como por sus numerosas aplicaciones.

El objetivo de este trabajo es presentar una recopilación de los principales resultados en esta área. Se comienza por las definiciones básicas de teoría de grafos. Luego se introducen los aspectos fundamentales de coloreo y se presentan variantes que permiten enriquecer la teoría y nos proveen de herramientas para tratar aplicaciones prácticas. Por último se estudian algunos algoritmos y cuestiones relacionadas a ellos.

1.1. Algunas notas históricas

La teoría de coloreo de grafos inicia en 1852 cuando Francis Guthrie notó que podía colorear un mapa de todas las regiones de Inglaterra de modo tal que dos regiones limítrofes tuvieran colores distintos usando solo cuatro colores. Más aún Guthrie observó que sin importar cuan complejos eran sus dibujos, no le era posible crear un mapa ficticio que requiriera de más de cuatro colores para ser coloreado, estableciendo así la conjetura de los cuatro colores. Enunciada de manera más simple posible esta dice:

"A lo sumo se necesitan cuatro colores para pintar cualquier mapa político de modo tal que dos países limítrofes no tengan el mismo color".

Formalmente se tiene que el problema de coloreo de mapas es equivalente al coloreo de grafos planares. Es decir que todo grafo planar es 4-coloreable. Más adelante veremos el significado preciso de estos términos.

Sorprendentemente es aún hoy en día uno de los resultados más difíciles en este campo. El problema se difundió rápidamente entre la comunidad

matemática. En 1879 Alfred Kempe publicó un paper afirmando haber demostrando el teorema. Tal fue el prestigio que obtuvo Kempe por su logro que fue nombrado miembro de la *Royal Society* y luego presidente de la *London Mathematical Society*. Sin embargo en 1890 Percy John Heawood demostró que el argumento de Kempe era incorrecto. El teorema de los cuatro colores volvía así a ser solo una conjetura. Pero en su paper Heawood logró probar el teorema para cinco colores usando en gran parte las ideas que Kempe había introducido originalmente. Durante casi un siglo la conjetura de los cuatro colores quedó sin resolver a pesar del incesante esfuerzo, hasta que en 1976 Kenneth Appel y Wolfgang Haken lograron probar el teorema con una demostración que fue tanto innovadora como polémica por ser la primera demostración en parte realizada usando una computadora. La idea en esencia es mediante un argumento teórico reducir todos los infinitos grafos a ser analizados a solo una colección finita que puede ser verificada por una computadora. Debido a la gran complejidad de la demostración, el resultado no fue aceptado inmediatamente. Más aún, algunos años más tarde Appel y Haken publicaron una serie de correcciones del paper que generaron incluso más incertidumbre sobre la validez del resultado obtenido. Pero finalmente toda posible duda fue extinguida cuando en 1997 el equipo compuesto por Robertson, Sanders, Seymour y Thomas realizó una nueva demostración siempre basada en el uso de una computadora, pero de manera más clara y ordenada.
Es interesante notar que aún hoy en día no se conoce una demostración del teorema de los cuatro colores que sea puramente matemática.

La demostración del teorema de los cuatro colores no produjo desinterés en el estudio sobre coloreo de grafos, si no que por lo contrario esta área continuó floreciendo y capturando cada vez más la atención de los investigadores.

Capítulo 2

Definiciones y Conceptos Generales

2.1. Definiciones

Definición 2.1.1. Un **grafo** es una terna formada por un conjunto V cuyos elementos se denominan **vértices**, un conjunto E cuyos elementos se denominan **aristas** y una correspondencia que le asigna a cada arista un par de vértices, llamados extremos.

Definición 2.1.2. Un grafo es **finito** si los conjuntos V y E lo son.

Nota: En este trabajo nos ocuparemos exclusivamente de grafos finitos pues son dos ramas bien diferentes dentro de la teoría de grafos.

Definición 2.1.3. Una arista que tiene ambos extremos iguales se denomina **loop**.

Definición 2.1.4. Aristas múltiples son aquellas que tienen los mismos extremos.

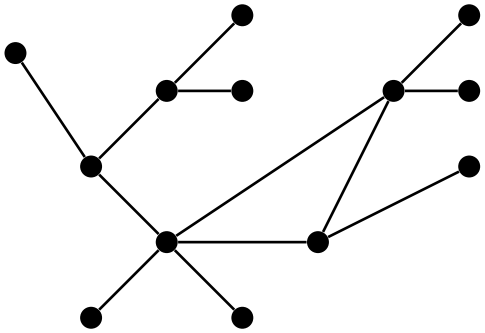
Definición 2.1.5. Decimos que un grafo es **simple** si no tiene ni loops ni aristas múltiples.

Observación: Cuando se trabaja con grafos simples se puede decir directamente que un grafo es un par (V, E) donde V es un conjunto finito cuyos elementos se denominan **vértices** y E es un conjunto de pares (i, j) con $i, j \in V,\ i \neq j$ que se denominan **aristas**.

Notación: La notación usual para un grafo es $G(V, E)$.
Dado un vértice v, en base a la definición anterior, la manera de indicar que v es un vértice de G es: $v \in V$. Pero es común directamente decir que $v \in G$.
Del mismo modo si e es una arista de G se suele notar $e \in G$.

Nota: A menos que se indique lo contrario, en este trabajo nos ocuparemos exclusivamente de grafos simples.

Representación Gráfica: Es usual representar a un grafo como un diagrama indicando a los vértices como puntos y las aristas como líneas que los conectan. Por ejemplo:



Observación: Un grafo puede tener más de una representación gráfica.

Definición 2.1.6. Si en un grafo G distinguimos entre el par (i, j) y el par (j, i), decimos que G es un **grafo orientado**. En este caso las aristas se dibujan como flechas para indicar la dirección. Además se suele usar el término **nodo** para referirse a los vértices y **arco** para las aristas.

Definición 2.1.7. Dos vértices i, j son **adyacentes** si $(i, j) \in E$.

Definición 2.1.8. Si $V = \emptyset$ decimos que G es el grafo **nulo**.
Muchos teoremas no son válidos para el grafo nulo por lo que en general cuando decimos que una propiedad vale para todo grafo implícitamente se está excluyendo el grafo nulo.

Definición 2.1.9. Un grafo G es **trivial** si consiste en un solo vértice y por lo tanto ninguna arista.

Definición 2.1.10. Dado v vértice de G, definimos el conjunto de **vecinos** $N(v)$ como el conjunto de los vértices adyacentes a v, y definimos el **grado** de un vértice $d(v) := |N(v)|$ como la cantidad de vecinos. Además notamos $\delta(G)$, $\Delta(G)$ al grado mínimo y máximo respectivamente, entre todos los vértices de G. Decimos que v es un vértice **universal** si $d(v) = n - 1$ y que v es un vértice **aislado** si $d(v) = 0$. Un grafo se dice **regular** si todos los vértices tienen el mismo grado. En particular, si todos los vértices tienen grado k decimos que es k-regular.

Teorema 2.1.1. Conocido como el *teorema del apretón de manos*, este resultado si bien es bastante intuitivo resulta de gran utilidad en teoría de grafos.

Para todo grafo G la suma de todos los grados de sus vértices es igual al doble de la cantidad de aristas.

$$\sum_{v \in V} d(V) = 2m_G$$

con m_G el número de aristas de G.

Demostración. Por inducción en m_G. Claramente vale para $m_G = 0$. Dado el grafo G, consideremos el grafo H que se obtiene al quitar un arista cualquiera, digamos (i, j). Luego H tiene $m_H = m_G - 1$ aristas y los grados de todos sus vértices son los mismos que los de G con excepción de i y j que resultan ser

$$d_H(i) = d_G(i) - 1 \ y \ d_H(j) = d_G(j) - 1$$

por hipótesis inductiva

$$\sum_{v \in V_H} d(V) = 2m_H$$

entonces

$$d_G(i) - 1 + d_G(j) - 1 + \sum_{v \in V_H \, v \neq i,j} d(V) = 2m_H = 2(m_G - 1)$$

luego

$$\sum_{v \in V} d(V) - 2 = 2m_G - 2$$

□

Observación: El uso de inducción en la demostración anterior no es casual, si no que por el contrario suele ser la herramienta fundamental para realizar demostraciones en teoría de grafos.

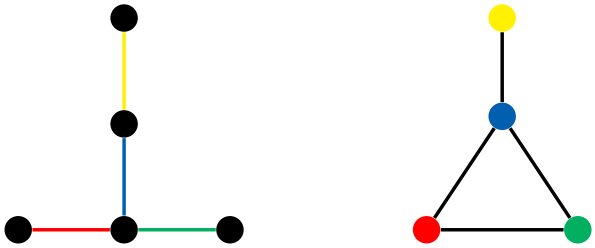
Corolario 2.1.2. Todo grafo tiene un número par de vértices de grado impar.

Definición 2.1.11. Definimos el **complemento** de un grafo G y lo notamos $\overline{G}$ como el grafo que tiene el mismo conjunto de vértices pero ahora dos vértices de $\overline{G}$ son adyacentes si y solo si no lo son en G.

Definición 2.1.12. Dos aristas que comparten un mismo vértice se denominan **incidentes**.

Definición 2.1.13. Dado un grafo G se define el **grafo línea** $L(G)$ como el grafo que se obtiene tomando un vértice por cada arista de G, y conectando dos vértices de $L(G)$ si y solo si las aristas de G son incidentes. El siguiente diagrama muestra un ejemplo. Para una mejor visualización a cada arista

de G y su vértice correspondiente en $L(G)$ se les asignó mismo color.



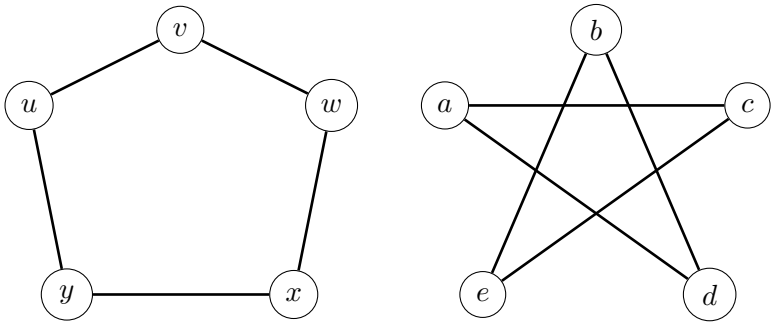
Definición 2.1.14. Decimos que H es un **subgrafo** de un grafo G si $V_H \subseteq V_G$ y $E_H \subseteq E_G$. Si $V_H = V_G$ decimos que H es un **subgrafo generador** de G.

Definición 2.1.15. Dado $X \subseteq V$ definimos el **subgrafo inducido** por X como el subgrafo H de G tal que $V_H = X$ y dos vértices de H son adyacentes si lo son en G. La notación para el subgrafo inducido por X es: $G[X]$.

Definición 2.1.16. Dos grafos G, H son **isomorfos** si existe una función $f : V_G \to V_H$ biyectiva tal que $(v, w) \in E_G \iff (f(v), f(w)) \in E_H$.
Es decir que conserva las adyacencias.

La noción de isomorfismo nos permite estudiar propiedades estructurales de los grafos, es decir que son independientes de los nombres que se les dan a los objetos. En nuestro caso en particular los problemas de coloreo resultan invariantes por transformaciones biyectivas de un grafo a otro.

El siguiente diagrama muestra dos grafos isomorfos. En este caso la biyección es: $f(u) = a; f(v) = c; f(w) = e; f(x) = b; f(y) = d$



Definición 2.1.17. Decimos que G es **completo** si todo par de vértices es adyacente y lo notamos K_n con n la cantidad de vértices de G.

Definición 2.1.18. Grafo de Intervalos. Dada una colección de intervalos abiertos en $\mathbb{R}$ se representa a cada intervalo con un vértice y se unen dos vértices si los intervalos correspondientes tienen intersección no vacía. No todo grafo puede ser representado como una intersección de intervalos en $\mathbb{R}$, por ejemplo es fácil que un ciclo C_4 no es un grafo de intervalos.

2.2. Caminos y Ciclos

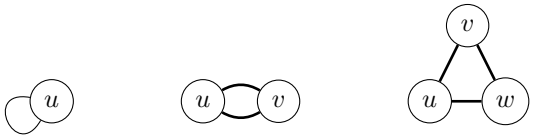
Definición 2.2.1. Un **paseo** es una secuencia de vértices $v_1, v_2, ..., v_n$ donde $(v_i,\, v_{i+1}) \in E$. Llamamos **extremos** a los vértices $v_1\, y\, v_n$. Si $v_1 = v_n$ decimos que es un paseo **cerrado**.

Definición 2.2.2. Dado un paseo $v_1, v_2, ..., v_n$ definimos su **longitud** como su número de aristas.

Definición 2.2.3. Un **recorrido** es un paseo que no repite aristas.

Definición 2.2.4. Un **camino** es un paseo que no repite vértices (en consecuencia tampoco repite aristas). La imagen siguiente muestra caminos de longitud: 0,1,2 respectivamente.

Definición 2.2.5. Un **ciclo** es un paseo cerrado que solo repite el primer y último vértice. La imagen siguiente muestra un ejemplo de ciclos C_1, $C_2\, y\, C_3$.



2.3. Grafos Conexos

Definición 2.3.1. Un grafo es **conexo** si para todo par de vértices v, w existe un camino que los une.

Definición 2.3.2. La **conectividad** de un grafo G es el cardinal del menor subconjunto $S \subseteq V$ tal que $G - S$ es disconexo o tiene un solo vértice. Lo notamos $\kappa(G)$.

Definición 2.3.3. Decimos que un grafo G es **k-conexo** si su conectividad es al menos k.

Definición 2.3.4. Un **corte por vértices** de un grafo G es un subconjunto $S \subseteq V$ tal que $G - S$ es disconexo. Cuando el conjunto S está compuesto por un solo vértice v, decimos que es un **vértice de corte**.

Teorema 2.3.1. Sean G conexo y $v \in G$, son equivalentes:

1. v es un punto de corte.
2. Existen vértices u y w distintos de v, tales que todo camino entre ambos pasa por v.
3. Existe una partición de $V - v$ en conjuntos U, W tales que para todo $u \in U$, $w \in W$ el camino que une u con w pasa por v.

Demostración. $(1 \Rightarrow 3)$ Como v es un punto de corte, $G - v$ es disconexo. Definimos U como una de las componentes conexas obtenidas y sea W el conjunto compuesto por los vértices restantes. Tomamos $u \in U$ y $w \in W$, como pertenecen a componentes conexas distintas entonces todo camino entre ellos pasa por v.
$(3 \Rightarrow 2)$ Basta con tomar $u \in U$, $w \in W$.
$(2 \Rightarrow 1)$ Como v está en todo camino entre u y w luego no puede existir un camino entre ambos vértices en $G - v$. Por lo tanto resulta disconexo, y entonces v es un vértice de corte. □

Definición 2.3.5. Dado un grafo G, un **bloque** es un subgrafo maximal que no contiene vértices de corte.

Definición 2.3.6. Decimos que un grafo $G(V, E)$ es **k-arista-conexo** si $\forall X \subseteq E$ con $|X| < k$ el grafo $G'(V, E \setminus X)$ sigue siendo conexo. Observar que si G es k-arista-conexo en particular es i-arista-conexo para $i = 1, ..., k$. Al mayor k tal que G es k-arista-conexo lo notamos $\lambda(G)$.

Proposición 2.3.2. Se tiene la siguiente desigualdad

$$0 \leq \lambda(G) \leq \delta(G) \leq n - 1$$

Demostración. Sea $v_0 \in G$ el vértice con $\delta(G)$ vecinos, entonces es suficiente con quitar esas $\delta(G)$ aristas para desconectar el grafo. □

Teorema 2.3.3. Un grafo $G(V, E)$ es k-arista-conexo **si y solo si** no existe un subconjunto propio $W \subseteq V$ tal que la cantidad de aristas que hay entre W y $(V \setminus W)$ es menor estricta que k.

Demostración. Si G es k-arista-conexo y existe un subconjunto propio de $W \subseteq V$ tal que entre $(V \setminus W)$ y W hay menos de k aristas, luego quitando a lo sumo $k - 1$ aristas desconectamos al grafo en dos componentes. Llegando así a un absurdo.
Supongamos ahora que no existe un subconjunto W tal que la cantidad de

aristas que hay entre W y $(V \setminus W)$ es menor estricta que k y aun así G no es k-arista-conexo. Luego por no ser k-arista-conexo podemos desconectar a G quitando a lo sumo $k-1$ aristas. Notemos con H una de las componentes conexas obtenidas. Tomando $W = H$ llegamos a un absurdo pues la cantidad de aristas de G que tienen un extremo en H y otro en $V \setminus H$ es menor estricta que k. □

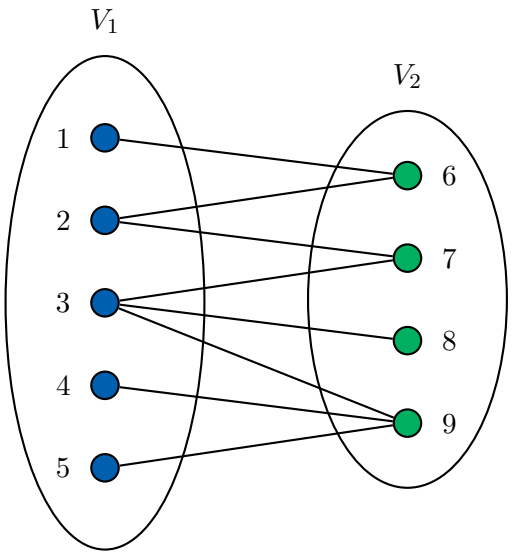
Definición 2.3.7. Valiéndonos del concepto de conexión, podemos definir la **distancia** entre dos vértices v, w como la longitud del camino más corto entre v, w. Es fácil corroborar que esta noción de distancia cumple las hipótesis de un espacio métrico.

Definición 2.3.8. Dados dos grafos $G_1(V_1, E_1)$, $G_2(V_2, E_2)$ definimos su **unión disjunta** como el nuevo grafo $G(V, E)$ que se obtiene tomando la unión disjunta de los conjuntos de vértices $V := V_1 \cup V_2$ y la unión disjunta de los conjuntos de aristas $E := E_1 \cup E_2$. Claramente el nuevo grafo G será disconexo.

2.4. Grafos Bipartitos

Definición 2.4.1. Un grafo G se denomina **bipartito** si existen conjuntos V_1, V_2 disjuntos, no vacíos, tales que $V = V_1 \cup V_2$ y cada arista tiene un extremo en V_1 y otro en V_2. Además decimos que es **bipartito completo** si cada vértice de V_1 es adyacente a todo vértice de V_2. Notamos $K_{r,s}$ al grafo bipartito completo con $|V_1| = r$ y $|V_2| = s$.

Ejemplo de un grafo bipartito:



El siguiente teorema nos provee de una caracterización de los grafos bipartitos que resulta de fundamental importancia por ser considerablemente más practico de utilizar que la propia definición.

Teorema 2.4.1. Un grafo no trivial G es bipartito **si y solo si** no contiene ciclos impares.

Demostración. Podemos suponer que G es conexo, en caso contrario repetimos el siguiendo proceso para cada componente conexa.
Sean V_1, V_2 los conjuntos de vértices independientes de G. Sea $\{w_1, w_2, ..., w_k, w_1\}$ un k-ciclo. Suponiendo sin pérdida de generalidad que $w_1 \in V_1$, como cada arista de G tiene un extremo en V_1 y otro en V_2 entonces $w_2 \in V_2$, $w_3 \in V_1$ y así sucesivamente, tenemos que los vértices del ciclo con índice impar pertenecen a V_1 y los de índice par a V_2. En particular $w_k \in V_2$ por lo tanto es un ciclo par.
Para ver la otra implicación fijamos $v \in G$ y definimos a los conjuntos U, W formados por los vértices que se encuentran a distancia par/impar de v respectivamente. Luego $v \in U$. Queremos ver que U, W forma una bipartición de G. Para esto basta con probar que U no tiene vértices adyacentes y W no tiene vértices adyacentes. Supongamos que W tiene dos vértices adyacentes w_1, w_2. Sean P_1 y P_2 caminos de v a w_1, w_2 respectivamente. Sea z el ultimo vértice que P_1, P_2 tienen en común. Luego los sub-caminos P_1' y P_2' que conectan z con w_1, w_2, son ambos pares o impares, pues notar que P_1, P_2 tienen ambos longitudes impares por estar $w_1, w_2 \in W$. Luego los caminos P_1', P_2' y la arista w_1w_2 forman un ciclo impar. Llegando así a un absurdo. El razonamiento es análogo para el caso en que U tiene dos vértices adyacentes. □

Definición 2.4.2. Un grafo conexo acíclico se denomina **Árbol**. Un **bosque** es una unión de árboles, una **hoja** es un vértice de grado 1.

Proposición 2.4.2. Todo árbol es bipartito [6]

Demostración. Como los árboles son acíclicos, en particular no contienen ciclos impares luego por el teorema anterior (2.4.1) son grafos bipartitos. □

La siguiente definición generaliza el concepto de grafo bipartito.

Definición 2.4.3. Dado $k \in \mathbb{N}$ decimos que G es k-partito si existe una partición del conjunto de sus vértices V, en k conjuntos disjuntos de modo que toda arista tenga extremos en conjuntos distintos.

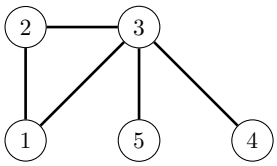
2.5. Representación de Grafos

Existen diferentes maneras de representar a un grafo. Cada una de ellas tiene alguna ventaja que la hace más apropiada en ciertas circunstancias. A continuación se mencionan algunas de las más importantes.

Matriz de adyacencias: Sea G un grafo con n vértices, definimos la matriz $A \in \{0, 1\}^{n \times n}$ como $A = a_{i,j}$ con $a_{i,j} = 1$ si los vértices i, j son adyacentes, y $a_{i,j} = 0$ en caso contrario.

Como consecuencia directa de la definición vemos que A es una matriz real, simétrica, con diagonal de ceros. Además como las columnas/filas de A corresponden a una enumeración arbitraria de los vértices de G, resulta de principal interés estudiar las propiedades de A que son invariantes por permutaciones de filas o columnas [5]. Como por ejemplo el radio espectral de A.

Veamos un ejemplo para el grafo siguiente:



La matriz de adyacencias queda:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Matriz de Incidencias: Se construye una matriz de $n \times m$ es decir que se tiene una fila para cada vértice y una columna para cada arista. En cada columna se colocan dos unos para indicar los vértices que forman los extremos del arista.
Para el grafo anterior si ordenamos las aristas de modo:
$(2, 3)$, $(1, 2)$, $(1, 3)$, $(3, 5)$, $(3, 4)$ nos queda la matriz:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Lista de Vecinos: Para cada vértice i se construye una lista L_i con todos sus vecinos. Este método por ejemplo resulta conveniente para algoritmos como BFS y DFS donde el orden de complejidad es $O(n)$ mientras que si se usara matriz de adyacencias el orden seria $O(n^2)$.

Para el mismo grafo del ejemplo anterior, las listas de vecinos son:
$L_1 = \{2, 3\}$, $L_2 = \{1, 3\}$, $L_3 = \{1, 2, 4, 5\}$, $L_4 = \{3\}$, $L_5 = \{3\}$.

2.6. Clases de Complejidad

Definición 2.6.1. Un **problema de decisión** es un problema cuya respuesta es: SÍ o NO.

Para un problema de decisión π definimos el conjunto D_π como el conjunto de **instancias** de π, es decir el conjunto de todas las especificaciones de sus parámetros. A su vez definimos el conjunto $Y_\pi \subseteq D_\pi$ de todos las instancias cuya respuesta es SÍ.

Definición 2.6.2. Clase P. Un problema de decisión se encuentra en la clase **P** si puede ser resuelto mediante un algoritmo determinístico en tiempo polinomial.

Definición 2.6.3. Clase NP. La manera más simple de introducir este concepto es decir que la clase NP está compuesta por los problemas de decisión para los cuales las instancias donde la respuesta es SÍ, dado un posible certificado pueden ser verificadas mediante un algoritmo determinístico en tiempo polinomial.

Por ejemplo en el caso del TSP (*Travelling salesman problem*), se tiene una lista de ciudades y una matriz de distancias entre cada par de ellas. El objetivo es visitar todas las ciudades y volver a la ciudad inicial, recorriendo la menor distancias posible. El problema de decisión consiste en: fijada una distancia D, responder la pregunta:
¿Existe un recorrido de longitud menor a D?
Claramente si nos dan un recorrido se puede controlar en tiempo polinomial, si la suma de las distancias entre todas las ciudades del recorrido es efectivamente menor a D.

El término NP significa *tiempo polinomial no determinista*. Más formalmente la clase NP está compuesta por los problemas de decisión que pueden ser resueltos usando una cantidad arbitraria (no predefinida) de maquinas de **Turing** (el lector que no esté familiarizado con este término, lo puede tomar como un sinónimo de la palabra *computadora*). Es decir que se comienza a resolver el problema con una maquina de Turing, luego se llega a una situación donde el problema se bifurca en dos ramas y agregamos otra maquina para resolver la segunda rama. Este proceso continúa de esta manera para cada una de las sub-ramas, y así sucesivamente hasta llegar a la solución. Notar que la secuencia de bifurcaciones que llegó a la solución lo hizo en tiempo polinomial. Por lo tanto si uno supiera en cada momento qué rama debería tomar se obtendría un algoritmo polinomial.
De la definición resulta inmediato que $P \subseteq NP$ pero aún hoy en día no se sabe si $P = NP$ o $P \neq NP$.

Definición 2.6.4. Una **reducción polinomial** de un problema π_0 a un problema π es una función $f : D_{\pi 0} \to D_\pi$ que se computa en tiempo polinomial y cumple:

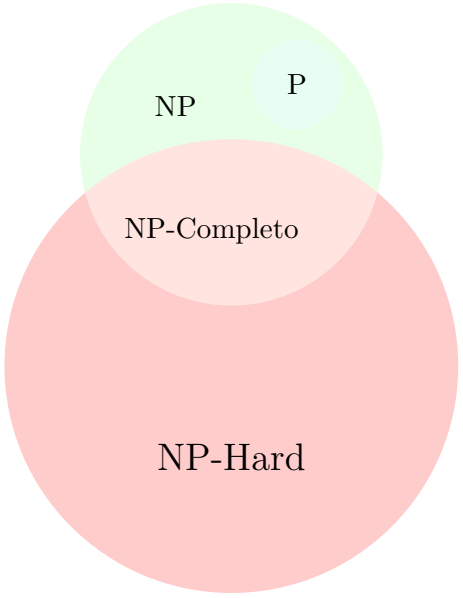
$$\forall d \in D_{\pi 0},\, d \in Y_{\pi 0} \Leftrightarrow f(d) \in Y_\pi$$

La notación utilizada es $\pi_0 \preceq \pi$. Observar que $\preceq$ es transitiva.

Definición 2.6.5. Un problema $\pi \in NP$ es **NP-Completo** si dado cualquier otro problema $\pi_0 \in NP$, este ultimo puede ser reducido a π en tiempo polinomial. Es decir que basta con poder encontrar una manera eficiente de resolver uno solo de los problemas de NP-Completo para poder resolver eficientemente cualquier problema de NP.

Definición 2.6.6. Por ultimo decimos que un problema π es **NP-Hard** si existe un problema $\pi_0 \in$ NP-Completo que es reducible a π en tiempo polinomial.

El diagrama siguiente muestra el caso en el que $P \neq NP$.



Problema de satisfacibilidad booleana (SAT)

Dada una fórmula booleana se quiere saber si sus variables pueden ser reemplazadas por valores **verdadero** o **falso** en modo tal que la evaluación de la fórmula sea **verdadera**. Este problema se suele llamar **SAT**.

Teorema 2.6.1. Cook–Levin. El problema **SAT** es NP-Completo.

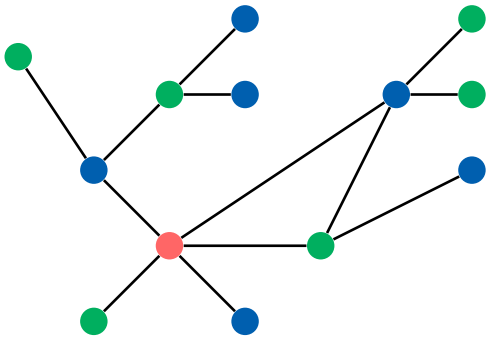
Técnica para probar que un problema π es NP-Completo

- Mostrar que $\pi \in$ NP.
- Elegir un problema $\pi_0 \in$ NP-Completo. Por ejemplo **SAT**.
- Construir una reducción polinomial de π_0 a π.

Capítulo 3

Coloreo Clásico

El objetivo es asignarle a cada vértice un color de manera que dos vértices adyacentes no tengan el mismo color. (Análogamente existe el coloreo de aristas). Claramente es trivial obtener un coloreo si se le asigna a cada vértice un color distinto, pero tanto la dificultad como las aplicaciones prácticas del concepto de coloreo se basan en encontrar la mínima cantidad de colores necesarios. Incluso en ciertos casos, debido a la dificultad del problema, el análisis se limita a encontrar cotas superiores e inferiores.
La imagen siguiente muestra un ejemplo de un coloreo de vértices.



El problema de coloreo es un aérea de gran interés e investigación activa. Además de tener muchas aplicaciones prácticas como por ejemplo: Scheduling, Register allocation, Asignación de Frecuencias, entre otros.

Dicho en términos más generales, supongamos que tenemos un conjunto S, en el cual hay ciertos pares de elementos que son incompatibles. El problema consiste en encontrar un partición de S compuesta por una cantidad mínima de subconjuntos, cada uno formado por elementos compatibles entre sí. Modelamos esta situación con un grafo, representando a cada elemento

del conjunto S con un vértice e indicando las incompatibilidades entre dos elementos agregando una arista que los une. Ahora el problema se reduce en encontrar un coloreo del grafo.

Para proporcionar una definición formal de coloreo, notar que cada color puede ser identificado con un número natural y en esencia nuestro objetivo será el de asignarle a cada vértice un número. De hecho el proceso de coloreo puede ser visto más bien como un proceso de etiquetado.

3.1. Definiciones

Definición 3.1.1. Un **k-coloreo** de los vértices de un grafo G es una función $c : V \to \{1, ..., k\}$ tal que $c(v) \neq c(w)$ si v, w son adyacentes.

Definición 3.1.2. El menor k tal que G es k-coloreable se denomina **número cromático** y se nota $\chi(G)$.

Definición 3.1.3. Definimos una **clique** de G como un subgrafo completo maximal. Notamos al cardinal de una clique máxima $\omega(G)$.

Definición 3.1.4. Definimos un **conjunto independiente** como un conjunto de vértices no adyacentes dos a dos. Notamos $\alpha(G)$ al cardinal de un conjunto independiente máximo.

Notar que $\alpha(G) = \omega(\overline{G})$ pues un conjunto de vértices independientes en G forma un subgrafo completo en $\overline{G}$.

Observación: Es interesante notar que un coloreo es una partición del conjunto de vértices en clases. Dos vértices adyacentes no pueden pertenecer a la misma clase. Sea i uno de los colores, $c^{-1}(i) = \{v \in V : c(v) = i\}$ es la i-esima clase. Luego cada clase forma un conjunto independiente de vértices.

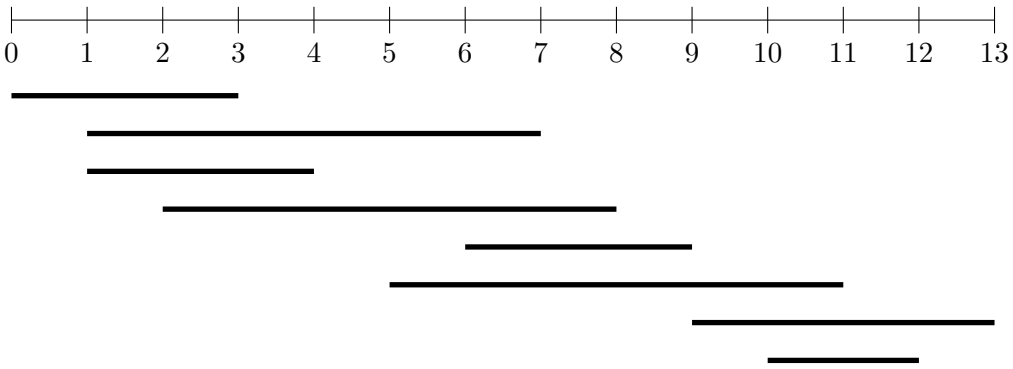
3.2. Aplicación: *Register allocation*

Los procesadores cuentan con memorias de alta velocidad que se denominan registros. La cantidad de registros es escasa por lo tanto se desea usarlos de manera eficiente. Notar que si dos variables no se usan simultáneamente, pueden ser asignadas al mismo registro. Luego para cada variable calculamos el primer y último momento en el cual la variable es usada. Diremos que una variable es activa cuando nos encontramos dentro de este intervalo. Podemos entonces modelar nuestro problema con un grafo de intervalos (ver def. 2.1.18). Representamos a cada variable con un vértice y unimos dos vértices si ambos están activos al mismo tiempo, es decir si sus correspondientes intervalos tienen intersección. El mínimo número de registros necesario para almacenar todas las variables es igual al número cromático del grafo.

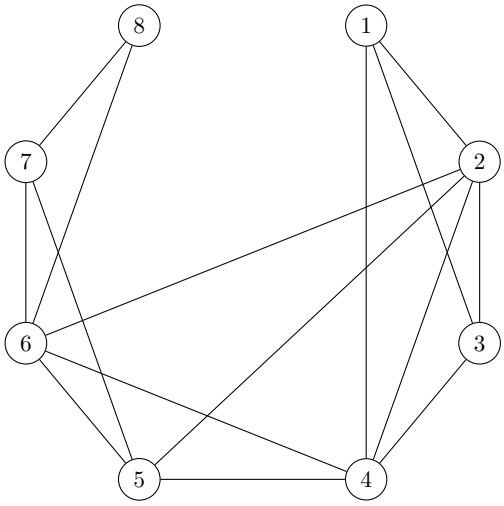
Ejemplo: Supongamos que tenemos 8 variables, cada una se encuentra activa en los intervalos:

$v_1[0, 3]$, $v_2[1, 7]$, $v_3[1, 4]$, $v_4[2, 8]$, $v_5[6, 9]$, $v_6[5, 11]$, $v_7[9, 13]$, $v_8[10, 12]$

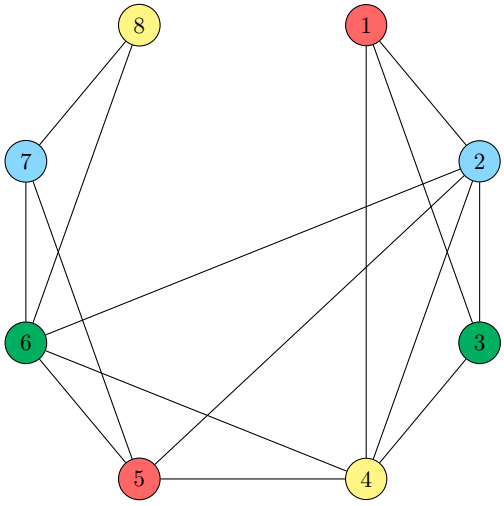
El diagrama siguiente muestra los distintos intervalos.



Procedemos a construir el respectivo grafo de intervalos:



Aplicando alguno de los métodos de la sección de algoritmos obtenemos un 4-coloreo del grafo, como muestra la siguiente figura:



Es decir que para almacenar las 8 variables son necesarios 4 registros.

3.3. Cotas para el número cromático

Proposición 3.3.1. Si G es el grafo completo K_n, entonces $\chi(G) = n$.

Demostración. Es fácil convencerse de que es necesario usar un color distinto para cada vértice, ya que todos están conectados entre sí. □

Proposición 3.3.2. Sea H un subgrafo de G, entonces $\chi(H) \leq \chi(G)$.

Demostración. Supongamos que $\chi(G) = k$, es decir que existe un k-coloreo de G. Pero entonces es también un k-coloreo de H, pues le asigna colores diferentes a cada vértice adyacente de G y por lo tanto en particular de H. □

Corolario 3.3.3. Como resultado del teorema notar que si G contiene un K_n luego $\chi(G) \geq n$. Entonces podemos afirmar que:

$$\forall G \;\; \chi(G) \geq \omega(G)$$

Proposición 3.3.4. Si G es un grafo disconexo con componentes conexas G_i con $i = 1...n$ entonces $\chi(G) = max\{\chi(G_i) : \; i = 1, ..., n\}$.

Proposición 3.3.5. Para todo grafo G con m aristas se tiene:

$$\chi(G) \leq \frac{1}{2} + \sqrt{2m + \frac{1}{4}}$$

Demostración. Sea c un coloreo de G con $k = \chi(G)$ colores. Luego hay por lo menos una arista entre cada clase de colores. De lo contrario dos clases que no están unidas por un arista podrían tener el mismo color. Luego $m \geq \frac{1}{2}k(k-1)$. Resolviendo la desigualdad para k se obtiene el resultado. □

Proposición 3.3.6. $\chi(G) \leq \Delta(G) + 1$

Demostración. (por inducción en el número de vértices)
Claramente la proposición vale para $n = 1$. Sea G un grafo con n vértices. Quitamos un vértice v cualquiera de G. Ahora $(G \setminus v)$ tiene $n - 1$ vértices y por hipótesis inductiva puede ser coloreado con a lo sumo $(\Delta(G \setminus v) + 1)$ colores. Entonces $G \setminus v$ resulta $(\Delta(G)+1)$-coloreable. Como v tiene a lo sumo $\Delta(G)$ vecinos y tenemos $(\Delta(G)+1)$ colores disponibles podemos colorear a v con un color no usado por sus vecinos y obtenemos así un $(\Delta(G)+1)$-coloreo de G. □

Proposición 3.3.7. Dado un grafo G, sea n la cantidad de vértices, luego: $n \leq \chi(G)\alpha(G)$

Demostración. Sea $k = \chi(G)$, tomemos un k-coloreo de G. Sean $c^{-1}(i)$ las clases de cada color. Luego $n = \sum_{i=1,...,k} |c^{-1}(i)| \leq k \times \alpha(G)$ pues $|c^{-1}(i)| \leq \alpha(G) \; \forall i = 1, ..., k$ por ser todos conjuntos independientes. □

Proposición 3.3.8. $\chi(G) + \alpha(G) \leq n + 1$

Demostración. Si reescribimos la desigualdad de modo: $\chi(G) \leq n - \alpha(G) + 1$, resulta más evidente, pues dado un conjunto independiente máximo, es claro que podemos asignar el mismo color a todos sus vértices. □

Proposición 3.3.9. Un grafo G es 2-coloreable **si y solo si** G es bipartito.

Demostración. Es automático ver que si el grafo es 2-coloreable, cada color forma un conjunto de vértices independiente. Por otro lado si el grafo es bipartito, digamos con juntos V_1, V_2 de vértices independientes, luego podemos colorear a todos los vértices de V_1 de un mismo color y todos los vértices de V_2 de otro color. □

Proposición 3.3.10. Sea G un grafo. Entonces $\chi(G) \geq 3$ **si y solo si** G tiene un ciclo impar.

Demostración. Vimos antes (Teorema 2.4.1) que un grafo es bipartito si y solo si no contiene ciclos impares. Luego usando la proposición anterior (3.3.9) tenemos que un grafo es 2-coloreable si y solo si no contiene ciclos impares. □

Proposición 3.3.11. Si G es un ciclo par entonces $\chi(G) = 2$.

Proposición 3.3.12. Si G es un ciclo impar entonces $\chi(G) = 3$.

Demostración. Por la proposición anterior $\chi(G) \geq 3$. Para ver que se cumple la igualdad basta con mostrar una manera de colorear cualquier ciclo impar usando solo 3 colores. Para esto basta con colorear a los vértices de G alternando los colores $1, 2$ sucesivamente hasta llegar al ultimo vértice que queda entre un vértice de color 1 y otro de color 2, por lo tanto es necesario asignare el color 3. □

Teorema 3.3.13. Para todo grafo G con n vértices se cumple:

$$\chi(G) \leq \left\lfloor \frac{n + \omega(G)}{2} \right\rfloor$$

En otras palabras $\chi(G)$ nunca puede estar más cerca de n que de $\omega(G)$. [31]

Demostración. Sea $n = |V(G)|$, la demostración es por inducción en $n - \omega(G)$. Si G es tal que $n - \omega(G) = 0$ entonces es un grafo completo $G = K_n$. Luego $\chi(G) = \omega(G) = n$ y por lo tanto vale el resultado.
Supongamos que existe un entero positivo k tal que para todo grafo H con n' vértices que satisface: $n' - \omega(H) < k$, se cumple la desigualad:

$$\chi(H) \leq \left\lfloor \frac{n' + \omega(H)}{2} \right\rfloor$$

Observar que para $k = 1$ efectivamente vale pues nos queda $n' - \omega(H) < 1$ es decir $n' - \omega(H) = 0$ que es justamente el caso analizado antes.
Sea G un grafo con n vértices tal que $n - \omega(H) = k \geq 1$. Como G no es completo existen vértices u, v no adyacentes. Sea $H := G - u - v$. Entonces H tiene $n-2$ vértices y $\omega(H) = \omega(G)$ o $\omega(H) = \omega(G) - 1$. De todos modos $0 \leq (n-2) - \omega(H) < k$. Luego por hipótesis inductiva:

$$\chi(H) \leq \left\lfloor \frac{n - 2 + \omega(H)}{2} \right\rfloor$$

Entonces existe un $\lfloor \frac{n-2+\omega(H)}{2} \rfloor$-coloreo de H. Para obtener un coloreo de G agregamos un color nuevo y se lo asignamos a u y v pues no son adyacentes. Tenemos entonces:

$$\chi(G) \leq \left\lfloor \frac{n - 2 + \omega(H)}{2} \right\rfloor + 1 = \left\lfloor \frac{n + \omega(H)}{2} \right\rfloor \leq \left\lfloor \frac{n + \omega(G)}{2} \right\rfloor$$

□

Proposición 3.3.14. Sea $G(V, E)$ con $|V| = n$, $|E| = m$ entonces:

$$\frac{n^2}{n^2 - 2m} \leq \chi(G)$$

Demostración. Sea $\chi(G) = k$, sea c un k-coloreo de G y sean V_i con $i = 1...k$ las clases de colores de c y notamos $|V_i| =: n_i$. Luego G tiene la mayor cantidad de aristas posible cuando el grafo es un grafo k-partito completo (def. 2.4.3) y el cardinal de cada clase es lo más cercano posible a $\frac{n}{k}$. Como hay $\binom{k}{2}$ pares de conjuntos y a lo sumo $(\frac{n}{k})^2$ aristas entre ellos tenemos que:

$$m \leq \binom{k}{2} \frac{n^2}{k^2}$$

de donde se obtiene:

$$2m \leq \frac{(k-1)n^2}{k}$$

Luego:

$$n^2 - \frac{(k-1)n^2}{k} \leq n^2 - 2m$$

Si invertimos ambos miembros de la desigualdad y los multiplicamos por n^2 nos queda:

$$\frac{n^2}{n^2 - 2m} \leq \frac{n^2}{n^2 - \frac{(k-1)n^2}{k}} = k = \chi(G)$$

Obtenemos así la desigualdad buscada. □

Teorema 3.3.15. Brooks. Sea G un grafo conexo y no dirigido, luego:

Si G es completo o un ciclo impar entonces $\chi(G) = \Delta(G) + 1$

De lo contrario tenemos la cota: $\chi(G) \leq \Delta(G)$

Demostración. Vimos antes el caso de que G es completo y también el caso de un ciclo impar.

Para probar el resto de la demostración usamos el algoritmo 'greedy' que se introducirá más adelante. La demostración se encuentra en 5.6.1, dentro de la sección de Algoritmos. □

3.4. ¿Cuán lejos puede estar χ(G) de ω(G)?

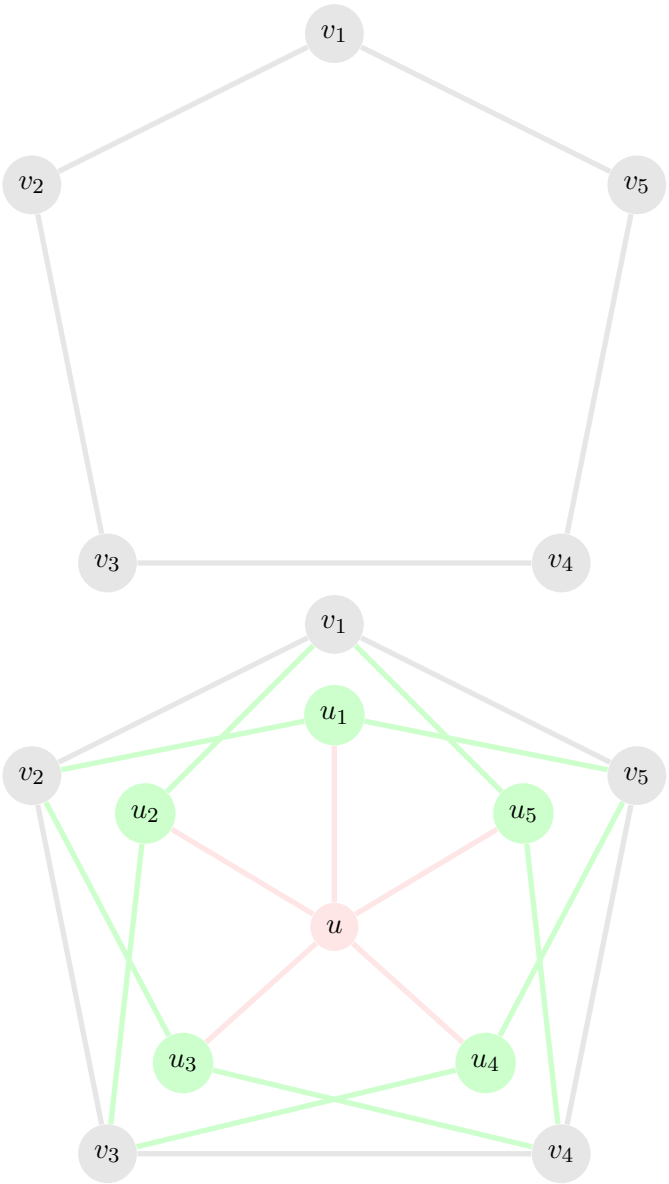
Anteriormente vimos que $\omega(G) \leq \chi(G)$ (Corolario 3.3.3). Uno podría pensar que en esencia el tamaño de la clique máxima determina el valor del número cromático y que luego es cuestión de buscar una manera de colorear los otros vértices alternando apropiadamente los colores. Esta idea intuitiva se formaliza más adelante con la definición de grafos perfectos, pero desafortunadamente en general esto es falso.

Como veremos en el siguiente teorema se pueden construir grafos con clique pequeña y número cromático arbitrariamente grande. Comenzamos por introducir la siguiente definición.

Definición 3.4.1. Dado un grafo H, decimos que el grafo G es **libre** de H si ningún subgrafo inducido de G es isomorfo a H.

Teorema 3.4.1. Para todo entero positivo k existe un grafo G libre de triángulos (i.e. K_3) con número cromático $\chi(G) = k$. Notar que un grafo libre de triángulos tiene clique $\omega(G) = 2$. La prueba que se presenta a continuación es de Jan Mycielski [10].

Demostración. El teorema claramente es válido para $k = 1, 2$. Para $k = 3$ proponemos $G = C_5$ i.e. el ciclo de 5 vértices. C_5 es libre de triángulos y $\chi(C_5) = 3$ por ser un ciclo impar (prop. 3.3.12). Probamos el teorema por inducción en k. Supongamos que existe un grafo H libre de triángulos con número cromático $\chi(H) = k \geq 3$. Queremos construir un grafo G libre de triángulos con número cromático $\chi(G) = k + 1$. Sean $v_1, ..., v_n$ los vértices de H, construimos el grafo G agregando $n + 1$ vértices $u, u_1, ..., u_n$ y conectamos u con cada vértice $u_{i=1,..,n}$ y cada vértice u_i con los vecinos de v_i en H. Las imágenes siguientes muestran un ejemplo en el caso $H = C_5$.



Veamos que efectivamente G en un grafo libre de triángulos y con $\chi(G) = k+1$. Probemos primero que G es libre de triángulos. Como $S = \{u_1, ..., u_n\}$ es un conjunto de vértices independiente y u no es adyacente a los vértices de H se concluye que u no forma triángulos en G. Luego en caso de existir un triángulo en G este debe tener dos vértices en H y uno en S. Notemos tal triángulo por $\{u_i, v_j, v_k\}$. Como u_i es adyacente a v_j y v_k, entonces v_i también es adyacente a v_j y v_k (pues u_i y v_i tienen los mismos vecinos en H). Pero entonces H contiene un triángulo, llegando así a una contradicción.

Veamos ahora que $\chi(G) = k+1$. Como H es un subgrafo de G y $\chi(H) = k$ entonces $\chi(G) \geq k$. Por otro lado dado un k-coloreo de H, asignemos a cada vértice u_i el mismo color del vértice v_i y le asignamos a u el color $k+1$. De este modo obtenemos un $(k+1)$-coloreo de G, por lo tanto $\chi(G) \leq k+1$. Entonces $k \leq \chi(G) \leq k+1$. Supongamos que $\chi(G) = k$ y lleguemos a una contradicción mostrando que H puede ser coloreado con $k-1$ colores. Sin pérdida de generalidad supongamos que el vértice u tiene color k. Necesariamente ningún vértice de S tiene el color k. Como $\chi(H) = k$ por lo menos un vértice de H es de color k. Notar que cada vértice v_i de H con color k puede ser re-coloreado con el color de u_i pues u_i tiene los mismos vecinos que v_i en H, y tener presente que queremos colorear solamente a H y no al grafo G. Esto produce un $(k-1)$-coloreo de H que lleva a una contradicción. Entonces $\chi(G) = k+1$. □

3.5. Grafos Perfectos

Definición 3.5.1. Un grafo G es **perfecto** si para todo subgrafo inducido H se cumple: $\omega(H) = \chi(H)$.

Visto en relación al Teorema anterior (3.4.1), un grafo perfecto no permite un número cromático arbitrariamente grande y una clique pequeña.

Esta familia de grafos introducida por Claude Berge en 1961 [11] es actualmente un área de gran interés. Entre otros motivos se encuentra el hecho de que calcular el tamaño de la clique máxima o el número cromático son problemas NP-completos en general, pero para grafos perfectos pueden ser resueltos en tiempo polinomial.

Teorema 3.5.1. Los grafos bipartitos son perfectos.

Demostración. Sea G un grafo bipartito y H un subgrafo inducido. Si H está compuesto solo por vértices aislados, luego $\chi(H) = \omega(H) = 1$. Si en cambio hay por lo menos dos vértices adyacentes entonces $\chi(H) = \omega(H) = 2$. De todos modos vale: $\chi(H) = \omega(H)$. Por lo tanto G es perfecto. □

Teorema 3.5.2. Los grafos de intervalos son perfectos. [22]

Definición 3.5.2. Sea G un grafo, un ciclo inducido de longitud impar 5 o mayor se denomina **odd hole**, y su complemento **odd antihole**.

3.6. Conjeturas de Berge

Entre 1961 y 1963 Claude Berge propuso dos conjeturas sobre los grafos perfectos.

1. Un grafo es perfecto **si y solo si** su complemento es perfecto.
2. Un grafo es perfecto **si y solo si** no contiene ni *odd holes* ni *odd antiholes*.

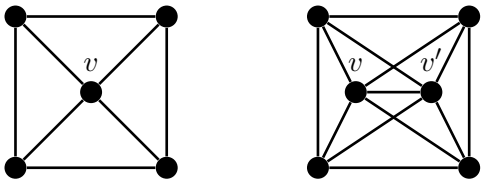
Definición 3.6.1. Sea G un grafo, decimos que G es **Berge** si no contiene ni *odd holes* ni *odd antiholes*.

La primer conjetura fue probada 10 años más tarde, en 1972 por László Lovász. El teorema se conoce como **Teorema (débil) de grafos perfectos**.

Para la segunda conjetura la parte: *un grafo es perfecto entonces es Berge*, fue probada por el mismo Berge, mientras que la otra implicación es un problema considerablemente más difícil de tratar y fue resuelto casi 45 años más tarde, por Maria Chudnovsky, Neil Robertson, Paul Seymour y Robin Thomas. El resultando fue anunciado en 2002 y publicado en 2006. La demostración es compleja y se basa en una descomposición estructural de los grafos de Berge.
Este teorema se conoce como **Teorema fuerte de grafos perfectos**, el motivo es que como la caracterización no es afectada por complementación, este teorema implica inmediatamente el teorema débil.

Para demostrar el teorema débil necesitamos una definición y un resultado adicional.

Definición 3.6.2. Sea G un grafo y v un vértice de G.
Definimos la **replicación** de G con respecto de v y lo notamos $R_v(G)$ al grafo obtenido al agregar un nuevo vértice v' conectado con la vecindad cerrada de v, $N[v]$, es decir conectamos a v' con todos los vecinos de v y con v. La imagen siguiente muestra un ejemplo:



Lema 3.6.1. de Replicación.
Sea G un grafo perfecto y v un vértice de G, entonces $R_v(G)$ es perfecto.

Demostración. Sea $G' = R_v(G)$. Veamos primero que $\chi(G') = \omega(G')$. Separamos el análisis en dos casos según v pertenece a una clique máxima de G o no.

Caso-1: v pertenece a una clique máxima de G. Luego $\omega(G') = \omega(G) + 1$ y como:

$$\chi(G') \leq \chi(G) + 1 = \omega(G) + 1 = \omega(G')$$

entonces tenemos que: $\chi(G') = \omega(G')$.

Caso-2: v no pertenece a una clique máxima de G. Definimos k como $\chi(G) = \omega(G) =: k$. Sea c un k-coloreo de G, usando los colores $1, ..., k$. Sin pérdida de generalidad podemos asumir que $c(v) = 1$. Sea $V_1 := c^{-1}(v)$, luego $v \in V_1$. Como $\omega(G) = k$ entonces cada clique máxima de G debe contener un vértice de cada color. Como v no pertenece a una clique máxima,

luego $|V_1| \geq 2$. Sea $U_1 := V_1 - \{v\}$. Como todas las cliques máximas de G contienen un vértice de U_1 tenemos que $\omega(G - U_1) = \omega(G) - 1 = k - 1$. Como G es perfecto, $\chi(G - U_1) = k - 1$. Tomemos un $(k-1)$-coloreo de $G - U_1$ usando los colores $1, ..., k-1$. Como V_1 es un conjunto independiente, entonces también lo es el conjunto $U_1 \cup \{v'\}$. Si asignamos el color k a los vértices de $U_1 \cup \{v'\}$ obtenemos un k-coloreo de G'. Entonces:

$$k = \omega(G) \leq \omega(G') \leq \chi(G') \leq k$$

por lo tanto $\chi(G') = \omega(G')$.

Nos falta probar que $\chi(H) = \omega(H)$ para todo subgrafo inducido H de G'. Claramente el resultado vale para todo subgrafo H de G por ser perfecto. Si $v' \in H$ pero $v \notin H$ luego H es isomorfo a $G[(V_H - \{v'\}) \cup \{v\}]$ y por lo tanto $\chi(H) = \omega(H)$. Si $v \in H$ y $v' \in H$ pero H no es isomorfo a G', luego H es la replicación de un grafo $G[V_H - \{v'\}]$ y por lo tanto podemos repetir el análisis que usamos para ver que $\chi(G') = \omega(G')$ para mostrar que $\chi(H') = \omega(H')$. □

Teorema 3.6.2. *Teorema (débil) de grafos perfectos. (Lovász)* [47]
Un grafo es perfecto si y solo si su complemento es perfecto.

Demostración. Como $\overline{\overline{G}} = G$, es suficiente con probar que si G es un grafo perfecto entonces $\overline{G}$ es perfecto. La demostración es por inducción en n (la cantidad de vértices). Para $n = 1$, G es perfecto y como $G = \overline{G}$ entonces $\overline{G}$ también es perfecto. Sea $n \geq 2$, supongamos que la propiedad vale para $n - 1$ y veamos que vale para n. Entonces dado un grafo perfecto G con n vértices queremos ver que $\overline{G}$ es perfecto, es decir que para todo subgrafo inducido F de $\overline{G}$ se cumple que $\chi(F) = \omega(F)$.
Para cada subgrafo inducido propio F de $\overline{G}$ existe un subgrafo inducido propio H de G tal que $\overline{H} = F$. Como G es perfecto, H es perfecto. Entonces por hipótesis inductiva F es perfecto.
Nos falta probar que $\chi(\overline{G}) = \omega(\overline{G})$, pero como $\chi(\overline{G}) \geq \omega(\overline{G})$ para todo grafo, entonces basta con probar que:

$$\chi(\overline{G}) \leq \omega(\overline{G})$$

Sea S el conjunto cuyos elementos son todos los conjuntos de vértices que forman cliques en G. Entonces si $U \in S$, U es un conjunto independiente en $\overline{G}$. Sea T el conjunto formado por todos conjuntos independientes máximos en G. Luego si $W \in T$, $|W| = \alpha(G)$.
Supongamos que existe un conjunto $U \in S$ tal que:

$$U \cap W \neq \emptyset \ \forall \ W \in T$$

Notar que dado $W \in T$, $U \cap W$ no puede contener más de un elemento, pues si v_1, $v_2 \in U \cap W$ luego por estar en W v_1, v_2 son independientes, pero por

estar en U son adyacentes. Absurdo.
Tenemos entonces que:

$$\omega(\overline{G} - U) = \alpha(G - U) = \alpha(G) - 1 = \omega(\overline{G}) - 1$$

Entonces por hipótesis inductiva:

$$\chi(\overline{G}) \leq \chi(\overline{G} - U) + 1 = \omega(\overline{G} - U) + 1 = \omega(\overline{G})$$

Nos falta probar entonces que efectivamente existe un conjunto $U \in S$ que cumple $U \cap W \neq \emptyset \ \forall \ W \in T$. Supongamos por el contrario que no existe un tal conjunto U. Luego para todo conjunto $U \in S \ \exists W_U \in T$ tal que $U \cap W_U = \emptyset$.
Para cada $x \in V_G$ definimos $n_x = |\{U \in S : x \in W_U\}|$. Ahora construimos un nuevo grafo G' a partir de G. Comenzamos por quitar todos los vértices x tales que $n_x = 0$. Para x tal que $n_x \geq 1$, reemplazamos el vértice x por un grafo completo de n_x vértices. Notamos a este grafo como G_x. Si x, y son adyacentes en G luego en G' conectamos todos los vértices de G_x con todos los vértices de G_y. Entonces el grafo G' queda definido como:

$$V(G') = \bigcup_{x \in V(G)} V(G_x)$$

y $(u, v) \in E(G')$ si y solo si $u \in V(G_x)$, $v \in V(G_y)$ y tal que $x = y$ o $(x, y) \in E(G)$.
Sea $H = G[\{x \in V(G) : n_x \geq 1\}]$. Como H es un subgrafo inducido del grafo perfecto G, tenemos que H es perfecto. Como el grafo G' puede ser obtenido de H a través de una serie de replicaciones de vértices, luego por el lema de replicación (3.6.1) G' es perfecto, entonces:

$$\chi(G') = \omega(G') \tag{3.1}$$

Analicemos ambos miembros de la igualdad. Comenzamos por $\omega(G')$. Por construcción de G' tenemos que cada subgrafo completo maximal de G' es de la forma $G'[U_{x \in X} V(G_x)]$ para algún $X \in S$. Luego existe un conjunto $Y \in S$ tal que:

$$\omega(G') = \sum_{y \in Y} n_y = \sum_{y \in Y} |\{U \in S : y \in W_U\}|$$

$$= \sum_{U \in S} |Y \cap W_U|$$

Como $G[Y]$ es completo y W_U es un independiente en G, tenemos que $|Y \cap W_U| \leq 1$ para todo $U \in S$. Más aún como consecuencia de nuestra suposición tenemos que $|Y \cap W_Y| = 0$. Entonces:

$$\omega(G') \leq |S| - 1 \tag{3.2}$$

Ahora consideramos $\chi(G')$. Notar que:

$$|V(G')| = \sum_{x \in V(G)} n_x = \sum_{x \in V(G)} |\{U \in S : x \in W_U\}|$$

$$= \sum_{U \in S} |W_U| = |S|\alpha(G)$$

Como $\alpha(G') \leq \alpha(G)$ por construcción de G', tenemos que:

$$\chi(G') \geq \frac{|V(G')|}{\alpha(G')} \geq \frac{|V(G')|}{\alpha(G)} = |S| \tag{3.3}$$

Entonces de las desigualdades (3.2) y (3.3), tenemos:

$$\chi(G') \geq |S| > |S| - 1 = \omega(G')$$

Que contradice la igualdad (3.1): $\chi(G') = \omega(G')$. □

Proposición 3.6.3. G perfecto $\Rightarrow$ G Berge.

Demostración. Sea G un ciclo impar de longitud mayor o igual a 5. Vimos anteriormente que $\omega(G) = 2$ y $\chi(G) = 3$. Entonces los grafos perfectos no pueden contener un *odd hole*. Por el teorema débil sabemos que $\overline{G}$ también es perfecto y por lo tanto no puede contener un *odd hole*. Como un *odd hole* en $\overline{G}$ es un *odd antihole* en G, entonces G no puede contener ni *odd holes* ni *odd antiholes*, es decir G es Berge. □

Teorema 3.6.4. *Teorema fuerte de grafos perfectos.* [48]
Un grafo es perfecto **si y solo** si es Berge.

Corolario 3.6.5. El teorema fuerte implica el teorema débil.

Demostración. Recordando que un *odd antihole* es el complemento de un *odd hole*, podemos reescribir el enunciado del teorema fuerte del siguiente modo:
G es perfecto **si y solo si** ni G ni $\overline{G}$ contienen *odd holes*.
Resulta claro que si G es perfecto entonces $\overline{G}$ es perfecto, y viceversa. □

3.7. Reconocimiento de Grafos Berge

A continuación se presentan las ideas generales de un algoritmo desarrollado por *Cornuejols, Chudnovsky, Liu, Seymour y Vuskovic* que permite decidir si un grafo dado es Berge en tiempo polinomial. En particular el orden de complejidad del algoritmo es $O(n^9)$. [50]
Notar que gracias al teorema fuerte de grafos perfectos, con este algoritmo podemos decidir si un grafo es perfecto.

El algoritmo aplicado a un grafo G determina si:

- G no es perfecto
- G no contiene ciclos impares

Luego para determinar si G es Berge se aplica el algoritmo a G y a $\overline{G}$.

Definición 3.7.1. Sea C un *odd hole* de longitud mínima, en G. Dado $v \in V(G) - V(C)$, decimos que v es un **C-major** si el conjunto de sus vecinos en C no está contenido un camino de longitud 3 en C.
Además decimos que C es **clean** si no hay vértices *C-major*
Por último dado un conjunto $X \subset V(G)$ decimos que es **cleaner** para C si $X \cap V(C) = \emptyset$ y todo vértice *C-mayor* pertenece a X.

Definición 3.7.2. Sea G un grafo que contiene un triangulo (b_1, b_2, b_3) i.e. un K_3. Una **pirámide** es un subgrafo inducido formado por la unión del triangulo (b_1, b_2, b_3), un cuarto vértice a y tres caminos P_1, P_2, P_3 tales que:

- P_i conecta b_i con a, para todo i
- para $1 \leq i < j \leq 3$, el único vértice en la intersección de P_i y P_j es a
- para $1 \leq i < j \leq 3$, la única arista de G entre $V(P_i) - \{a\}$ y $V(P_j) - \{a\}$ es (b_i, b_j)
- a es adyacente (a lo sumo) a uno de los vértices b_1, b_2, b_3

De la definición se deduce que toda pirámide es una subdivisión del grafo K_4. Notar que una pirámide queda determinada dados los caminos P_1, P_2, P_3.

Proposición 3.7.1. Sea G un grafo, si G contiene una pirámide, entonces G contiene un *odd hole*. [50]

Algoritmo para búsqueda de pirámides: [50]

Entrada: un grafo G.

Salida: el algoritmo encuentra una pirámide (por lo tanto un *odd hole* gracias a la prop. anterior) o determina que G no contiene pirámides.

Complejidad: $O(n^9)$

Definición 3.7.3. Sea G un grafo, dada un secuencia $\{v_1, v_2, v_3, v_4, v_5\}$ de vértices distintos y un camino P, decimos que forman un **joya** si:

- $(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_1) \in E(G)$
- $(v_1, v_3), (v_2, v_4) \notin E(G)$
- P es un camino inducido en G entre v_1, v_4 con la propiedad que v_2, v_3, v_5 no son adyacentes a los vértices interiores de P.

Proposición 3.7.2. Sea G un grafo, si G contiene una joya, entonces G contiene un *odd hole*.

Demostración. Sea $\{v_1, v_2, v_3, v_4, v_5, P\}$ una joya.
Si P es de longitud impar entonces $\{v_1, P, v_4, v_5, v_1\}$ en un *odd hole*.
Si P es de longitud par, $\{v_1, P, v_4, v_3, v_2, v_1\}$ es un *odd hole*. □

Algoritmo para búsqueda de joyas: [50]

Entrada: un grafo G.

Salida: el algoritmo decide si existe una joya en G (y por lo tanto un *odd hole*) o determina que no contiene joyas.

Complejidad: $O(n^6)$

Algoritmo para *odd hole* de longitud mínima [50]

Entrada: un grafo G sin pirámides ni joyas.

Salida: determina una de las siguientes alternativas:

1. G contiene un *odd hole*.
2. G no contiene un *clean odd hole* de longitud mínima.

Complejidad: $O(n^4)$

Algoritmo para búsqueda de *cleaners* [50]

Entrada: un grafo G.

Salida: determina una de las siguientes alternativas:

1. G no es Berge pues contiene un *odd hole / antihole*.
2. Genera una cantidad polinomial ($O(n^5)$) de subconjuntos $X \subset V(G)$ tales que si C es un *odd hole* de longitud mínima alguno de estos subconjuntos es un *cleaner* para C.

Complejidad: $O(n^6)$

Algoritmo para determinar si G es Berge [50]

Entrada: un grafo G.

Salida: determina si G es Berge.

Complejidad: $O(n^9)$

Esencialmente la idea detrás del algoritmo para determinar si G es Berge es la siguiente:

```
AlgoritmoBerge(G):

1. Ejecutar los algoritmos: [búsqueda de pirámides] y [búsqueda de joyas]
      si G contiene una piramide o una joya:
         Terminar. Return: G No es Berge
2. Ejecutar el algoritmo [búsqueda de cleaners]
      si G contiene un odd hole o antihole:
         Terminar. Return: G No es Berge
3. Para cada subconjunto X obtenido en el paso anterior:
      Para cada subconjunto Y de X tal que |Y| <= 3:
         Ejecutar el algoritmo [odd hole de longitud mínima] a G(X\Y)
         si G contiene un odd hole:
            Terminar. Return: G No es Berge
4. Return True


Inicio:

Ejecutar AlgoritmoBerge(G)

Ejecutar AlgoritmoBerge(Complemento de G)

si AlgoritmoBerge(G) & AlgoritmoBerge(Complemento de G):
   Return: G es Berge
```

Un comentario adicional:

Utilizando este algoritmo no es posible determinar si G contiene un *odd hole*. Si bien en un primer momento parecería que esta respuesta puede obtenerse con el algoritmo de reconocimiento de grafos Berge, esto no es así. El problema es que si en un determinado momento el algoritmo detecta que el grafo complemento $\overline{G}$ contiene un *odd hole*, se concluye que G no es Berge y la ejecución del algoritmo se detiene. Entonces en este caso no podemos saber si G contiene o no un *odd hole*.

Este problema fue presentado en el paper original de reconocimiento y ha sido un problema abierto por más de 10 años. Recientemente M. Chudnovsky, A. Scott, P. Seymour, S. Spirkl publicaron un paper donde muestran un algoritmo polinomial para detectar *odd holes*. [49]

Si bien este nuevo algoritmo no mejora el orden de complejidad, (sigue siendo $O(n^9)$), nos provee de un nuevo algoritmo para reconocer grafos perfectos que resulta considerablemente más simple.

3.8. Grafos Color-Crítico

Dado un grafo $G(V, E)$ con número cromático $\chi(G) = k$, tomamos $v \in V$. Se presentan dos alternativas, $\chi(G - v) = k$ o $\chi(G - v) = k - 1$, pues si al quitar el vértice v el número cromático se reduce más de uno por ejemplo $\chi(G - v) = k - 2$ entonces podríamos volver a agregar al vértice y asignare un color nuevo. Pero entonces obtenemos un $(k - 1)$-coloreo de G y por lo tanto un absurdo. Este hecho motiva la siguiente definición.

Definición 3.8.1. Un grafo es **color-crítico** si $\chi(H) < \chi(G)$ para todo subgrafo propio H de G. Si un grafo G que es color-crítico tiene número cromático $\chi(G) = k$ decimos que G es **k-crítico**.

Proposición 3.8.1. Sea G un grafo color-crítico entonces G es conexo.

Demostración. Supongamos por el contrario que G tiene al menos dos componentes, las notamos H_1, H_2. Luego una de ellas tiene número cromático menor o igual a la otra. Sin pérdida de generalidad supongamos $\chi(H_1) \leq \chi(H_2)$. Pero entonces si quitamos un vértice o arista de H_1 el número cromático de H_2 se mantiene igual y por lo tanto también el de G. Contradiciendo así la hipótesis de ser color-crítico. □

Proposición 3.8.2. El grafo K_2 es el único grafo 2-crítico.

Demostración. Claramente K_2 es 2-crítico. Para ver que es único observar que si existe una arista se necesitan al menos dos colores. □

Proposición 3.8.3. Los grafos completos K_n son n-críticos

Demostración. Sabemos que $\chi(K_n) = n$, además observar que $K_n - v = K_{n-1}$. Si en cambio quitamos un arista de G, podemos usar el algoritmo de conexión y contracción (5.1) para demostrar que el número cromático es efectivamente menor. □

Proposición 3.8.4. Los ciclos impares son los únicos grafos 3-críticos.

Demostración. Vimos antes (prop. 3.3.12) que si G es un ciclo impar $\chi(G) = 3$. Ahora si quitamos un vértice o arista de G podemos colorear al grafo usando solo 2 colores pues resulta un subgrafo de un ciclo par (prop. 3.3.11). Supongamos ahora que G es 3-crítico, queremos ver que G es un ciclo impar. Usando la proposición (3.3.10) sabemos G contiene un ciclo impar. Si G no fuera solamente un ciclo impar, pero además tuviera otros vértices (o aristas), entonces podríamos quitar uno de estos vértices (o aristas) y seguir teniendo un ciclo impar. Pero entonces seguirían siendo necesarios 3 colores para colorear a G, contradiciendo la hipótesis de ser 3-crítico. □

Teorema 3.8.5. Todo grafo $G(V, E)$ k-crítico, $k \geq 2$, es $(k-1)$-aristas-conexo.

Demostración. Por lo visto en las dos proposiciones anteriores K_2 es el único grafo 2-crítico, además es 1-arista-conexo. Los ciclos impares son los únicos grafos 3-críticos, además son 2-aristas-conexos. Entonces queremos probar el teorema para $k \geq 4$. Supongamos que existe un grafo G k-crítico que no es $(k-1)$-aristas-conexo. Luego por el Teorema (2.3.3) existe una partición de los vértices de G en dos conjuntos V_1, V_2 tales que la cantidad de aristas que hay entre V_1, V_2 es a lo sumo $k-2$. Como G es k-crítico, los dos subgrafos inducidos $G_1 = G[V_1]$, $G_2 = G[V_2]$ son $(k-1)$-coloreables. Tomemos un $(k-1)$-coloreo para G_1 y otro para G_2. Sea $E' \subseteq E$ el conjunto de aristas de G que tienen un extremo en V_1 y otro en V_2. No puede suceder que toda arista de E' une vértices de colores distintos, pues de ser así G seria $(k-1)$-coloreable. Luego existen aristas en E' que unen vértices que son del mismo color. Mostraremos a continuación que existe una permutación de los colores asignados a los vértices de V_1 en la cual toda arista de E' une vértices de colores distintos, produciendo de este modo un $(k-1)$-coloreo de G y por lo tanto generando una contradicción.
Sean $U_1, ..., U_t$ las clases de colores de G_1 (i.e. $U_i = c(i)^{-1}$) para las cuales existen vértices que son adyacentes a vértices de V_2. En particular supongamos que hay k_i aristas que conectan vértices de U_i con V_2. Por construcción $k_i \geq 1 \forall i$ y además $\sum_{i=1}^{t} k_i \leq k-2$, pues dijimos que la cantidad de aristas entre V_1 y V_2 es a lo sumo $k-2$. Si para cada vértice $u_1 \in U_1$ sus vecinos en V_2 no tienen el mismo color que u_1, entonces el conjunto U_1 queda como está. Si en cambio existe un vértice $u_1 \in U_1$ con el mismo color que un vértice en V_2 entonces podemos permutar los $(k-1)$ colores usados para colorear a G_1 de modo tal que no queden vértices en U_1 adyacentes a vértices en V_2 con el mismo color. Esto es posible pues hay a lo sumo k_1 colores prohibidos para colorear los vértices de U_1 y por lo menos $k-1-k_1 \geq 1$ colores posibles para colorear a U_1. Si luego de realizar la permutación de colores mencionada no hay ningún vértice $u_2 \in U_2$ que tiene el mismo color que un vértice adyacente en V_2, entonces no es necesario modificar los colores. Si en cambio existe un vértice $u_2 \in U_2$ que tiene el mismo color que uno de sus vecinos en V_2. En este caso volvemos a realizar una permutación de los colores asignados a los

vértices de G_1 pero sin modificar los colores de U_1. Este proceso es posible pues hay a lo sumo k_2+1 colores prohibidos para colorear los vértices de U_2 y $(k-1)-(k_2+1) \geq (k-1)-(k_2+k_1) \geq 1$ colores posibles para utilizar. Repetimos este proceso hasta obtener un $(k-1)$-coloreo de G. □

Teorema 3.8.6. Para todo grafo G vale

$$\chi(G) \leq max\{\lambda(H)\} + 1$$

donde el máximo se toma sobre todos los subgrafos H de G y $\lambda(G)$ es el mayor k tal que G es k-aristas-conexo (ver definición 2.3.6).

Demostración. Sea F un subgrafo color-crítico tal que $\chi(G) = \chi(F)$. Por el teorema anterior (3.8.5) F es $(\chi(F)-1)$-aristas-conexo. Pero entonces $\lambda(F) \geq \chi(F) - 1$. Obtenemos entonces:

$$\chi(G) = \chi(F) \leq \lambda(F) + 1 \leq max\{\lambda(H)\} + 1.$$

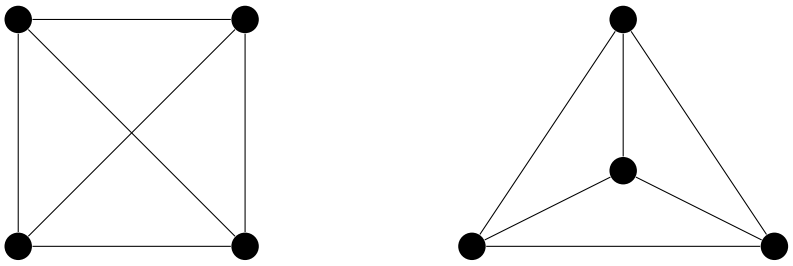
□

Corolario 3.8.7. Si G es un grafo color-crítico entonces $\chi(G) \leq \delta(G) + 1$

Demostración. Simplemente porque λ(G) $\leq\delta$(G) (ver proposición 2.3.2) □

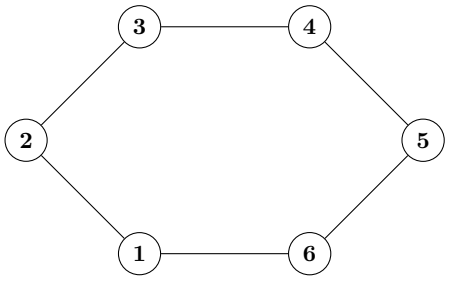
3.9. Grafos Planares

Un grafo es planar cuando puede ser dibujado en el plano de modo tal que no haya aristas que se crucen.
A continuación se muestran dos representaciones del grafo completo K_4. La primera no es planar mientras que la segunda sí lo es.

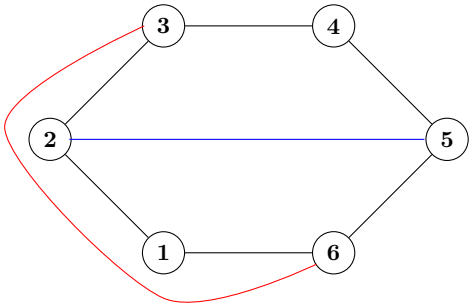


Proposición 3.9.1. El grafo $G = K_{3,3}$ no es planar.

Demostración. El método que se utilizará a continuación es la técnica clásica para este tipo de problemas [9].
Sean V_1, V_2 los conjuntos de vértices independientes de G. Notemos a los vértices $v_1, ..., v_6$, de modo que $V_1 = \{v_1, v_3, v_5\}$ y $V_2 = \{v_2, v_4, v_6\}$. Luego la secuencia vértices $v_1, v_2, v_3, v_4, v_5, v_6, v_1$ forma un ciclo en G. Pero entonces cualquier representación gráfica de G debe contener este ciclo, que sin pérdida de generalidad puede ser representado en forma de hexágono como muestra la siguiente figura.



Consideremos por ejemplo los vértices v_2 y v_5 que son adyacentes. Luego la arista que los une debe estar en el interior del hexágono o en le exterior, pero no en ambos simultáneamente si pretendemos que no se cruce con las otras aristas. Supongamos que se encuentra en el interior, el otro caso se resuelve de manera similar. Miremos ahora los vértices v_3, v_6 que también son adyacentes. Entonces la arista que los une debe estar completamente fuera del hexágono pues de lo contrario se cruzaría con la arista v_2v_5. Nos encontramos entonces en la situación que se indica en la siguiente figura:



Pero entonces no hay manera de unir los dos vértices adyacentes v_1, v_4 sin que la arista se cruce con alguna otra. Concluimos entonces que $K_{3,3}$ no es planar. □

La caracterización de los grafos planares está dada por el teorema de Kuratowski.

Teorema 3.9.2. Kuratowski. Un grafo es planar **si y solo si** no contiene como subgrafo una subdivisión de un $K_{3,3}$ o un K_5.

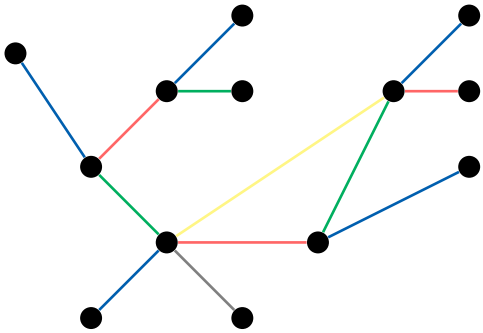
La definición de grafos planares nos permite enunciar el teorema de los cuatro colores de manera más precisa.

Teorema de los Cuatro Colores

Todo grafo planar es 4-coloreable.

3.10. Coloreo de Aristas

Consiste en colorear las aristas de un grafo de manera que dos aristas que inciden sobre un mismo vértice no tengan el mismo color.



Definición 3.10.1. Definimos el **Índice Cromático** como el menor número de colores necesarios para realizar un coloreo de aristas. Notación: $\chi'(G)$.

Observación: Un coloreo de los vértices de G es un coloreo de aristas del grafo línea $L(G)$, en particular tenemos que: $\chi'(G) = \chi(L(G))$.

Complejidad: En 1981 Ian Holyer probó que el problema de coloreo de aristas es NP-Completo. [27]

Proposición 3.10.1. Sea G un grafo, entonces $\chi'(G) \geq \Delta(G)$

El resultado anterior es evidente. Sin embargo es interesante notar que para grafos bipartitos se cumple la igualdad, como muestra la siguiente proposición.

Teorema 3.10.2. (König) Para todo grafo bipartito G se cumple:

$$\chi'(G) = \Delta(G)$$

Demostración. Por inducción en m (cantidad de aristas).
Vamos a notar $\Delta := \Delta(G)$.

El resultado vale para $m = 0$. Supongamos $m \geq 1$ y que el resultado vale para grafos con menos de m aristas.

Seleccionamos un arista cualquiera (x, y). Por hipótesis inductiva existe un Δ-coloreo de aristas de $G - (x, y)$. Por comodidad si un arista es de color α diremos que es una arista-α , y del mismo modo para los otros colores.

En el grafo $G - (x, y)$ los vértices x, y tienen a lo sumo $\Delta - 1$ aristas incidentes. Luego existen colores $\alpha, \beta \in \{1, ..., \Delta\}$ tales que x no es incidente

con una arista-α e y no es incidente con una arista-β. Si $\alpha = \beta$ podemos colorear la arista (x, y) con este color y así obtenemos el Δ-coloreo de aristas de G. Supongamos entonces que $\alpha \neq \beta$ y que x es incidente a una arista-β.

Ahora construimos un camino maximal W partiendo de x cuyas aristas son de color β, α alternadamente. Notar que W no repite vértices pues de lo contrario se tendrían dos aristas del mismo color incidentes pues todo vértice precedente tiene simultáneamente un arista-α y un arista-β, salvo por x que tiene solo una arista-β, pero x no puede tener arista-α. Entonces no es posible repetir vértices pues tenemos un coloreo de aristas. Además W no contiene a y pues de lo contrario W terminaría en y con una arista-α (pues por hipótesis y no es incidente con una arista-β) y entonces W tendría longitud par. Pero entonces $W+(x, y)$ sería un ciclo impar, imposible pues G es bipartito (prop. 2.4.1). Ahora recoloreamos las aristas de W intercambiando los colores α y β. Por ser W maximal el nuevo coloreo sigue asignando colores distintos para las aristas incidentes. De esta forma hemos obtenido un Δ-coloreo de aristas de $G-(x, y)$ donde ni x ni y son incidentes con una arista-β. Coloreando la arista (x, y) con β obtenemos un Δ-coloreo de aristas de G. □

3.11. Relación con Matching

Definición 3.11.1. Un **matching** en un grafo G es un conjunto de aristas tales que ningún par de ellas es adyacente. Un **matching perfecto** es un matching donde las aristas tocan todos los vértices del grafo. Un **matching máximo** es un matching que tiene la mayor cantidad posible de aristas. Al tamaño de un matching máximo lo notamos $\alpha'(G)$.

Sea G un grafo y $c_e : V \to \{1, ..., n\}$ un coloreo de aristas de G. Luego el conjunto de todas las aristas que recibieron el mismo color ($c_e^{-1}(i)$) forman un matching de G pues no hay dos aristas adyacentes. Entonces si miramos a todo el coloreo de aristas, lo que tenemos es una partición del grafo en matchings disjuntos.

3.12. Teorema de Vizing

Vimos antes que $\Delta(G)$ es una cota inferior para $\chi'(G)$.
En 1964 Vizing demostró que $(\Delta + 1)$ es una cota superior óptima para el índice cromático de un grafo.
Este teorema es sin dunda alguna el resultado más importante en el área de coloreo de aristas.

Teorema 3.12.1. Vizing.

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$$

Clasificación de grafos: El teorema di Vizing nos permite clasificar a los grafos en dos conjuntos. Los grafos **clase-1** que son aquellos que admiten un Δ-coloreo de aristas, y lo grafos **clase-2** que son aquellos que necesitan $\Delta + 1$ colores.

Observación: Por el teorema de König (3.10.2) tenemos que los grafos bipartitos son de clase-1.

Teorema 3.12.2. Sea K_n el grafo completo de n vértices, luego:

$$\chi'(K_n) = \begin{cases} n-1 \ si \ \ n \ es \ par \\ n \qquad si \ \ n \ es \ impar \end{cases}$$

Que es lo mismo que decir que K_n es de clase-1 si n es par y de clase-2 si n es impar pues $\Delta(K_n) = n - 1$.

Para demostrar el teorema necesitamos introducir una definición y dos resultados adicionales.

Definición 3.12.1. Sea G un grafo, definimos un **factor** como un subgrafo generador. Si el factor es k-regular (def. 2.1.10) lo llamamos **k-factor**. Decimos que G puede ser **factorizado** en factores F_i $i = 1, ..., n$ si los conjuntos de aristas de cada factor son disjuntos dos a dos y $\cup_{i=1,...,n} E(F_i) = E(G)$. Es decir si los factores forman una partición del conjunto de aristas de G. Si existe k tal que G puede ser factorizado de modo tal que todos los factores sean k-factores, decimos que G es **k-factorizable**.

Ejemplo: Notar que un 1-factor es un matching perfecto y una 1-factorización de un grafo k-regular es un k-coloreo de aristas del grafo.

Proposición 3.12.3. Para todo entero positivo k, el grafo completo K_{2k} es 1-factorizable. [8]

Proposición 3.12.4. Sea $G(V, E)$ un grafo con $|E| = m$. Entonces:

$$\frac{m}{\alpha'(G)} \leq \chi'(G)$$

con $\alpha'(G)$ el tamaño de un matching máximo. [8]

Demostración. (*del teorema* 3.12.2)

Si n es par, por la proposición (3.12.3) sabemos que K_n es 1-factorizable, es decir que existen $(n-1)$ 1-factores F_i $i = 1, ..., n-1$ que factorizan a K_n. Basta entonces con asignarle el color i a todas las aristas del factor F_i, y repetir el proceso para todos los factores. De esta forma se obtiene un $(n-1)$-coloreo de aristas de K_n.

Si n es impar, entonces $\alpha'(K_n) = (n-1)/2$. Por ser K_n completo entonces $m = n(n-1)/2$. Luego por la proposición (3.12.4) tenemos:

$$\chi'(K_n) \geq \frac{m}{\alpha'(K_n)} = n$$

Por el teorema de Vizing $\chi'(K_n) = n$. □

Nota sobre la complejidad: Existen algoritmos para colorear las aristas de un grafo con $\Delta + 1$ colores en tiempo polinomial, por ejemplo: [34] pero el problema de decidir si un grafo es de clase-1 o clase-2 es NP-Completo.

Vizing probó además el siguiente resultado.

Teorema 3.12.5. Todo grafo G de clase-2 tiene por lo menos tres vértices de grado máximo. [45]

Vamos a usar este resultado para probar el siguiente teorema.

Teorema 3.12.6. Casi todos los grafos son de clase-1. [43]

Este teorema es un resultado de Paul Erdös y Robin J. Wilson publicado en 1977. La afirmación se basa en un argumento probabilístico. La idea es que si $P(G_n)$ es la probabilidad de que un grafo aleatorio de n vértices sea de clase 1. Entonces $P(G_n) \to 1$ si $n \to \infty$. Notar que la probabilidad $P(G_n)$ se calcula tomando el cociente entre todos los grafos de n vértices que son de clase-1 y todos los grafos que se pueden formar con n vértices. En particular los grafos analizados son del tipo *Erdős-Rényi*, es decir que dados dos vértices, con probabilidad $\frac{1}{2}$ existe la arista entre ellos.

Para probar el teorema necesitamos el siguiente lema:

Lema 3.12.7. Casi todos los grafos tienen un único vértice de grado máximo. [43]

Demostración. (*del teorema* 3.12.6). Usando el lema anterior y el teorema 3.12.5 sabemos que casi todos los grafos tiene un solo vértice de grado máximo, y para ser de clase-2 se deben tener por lo menos 3 vértices de grado máximo. Luego casi todos los grafos son de clase-1. □

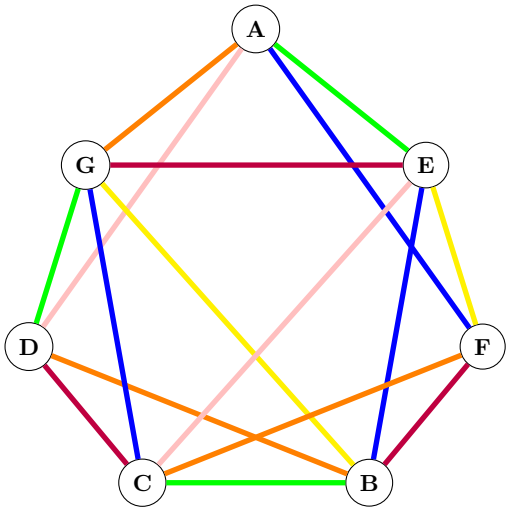
3.13. Aplicación para un problema de Scheduling

Se quiere organizar un torneo entre 7 equipos $\{A, B, C, D, E, F, G\}$ donde todos juegan contra todos con las siguientes excepciones:

- A no puede jugar contra B, C
- D no puede jugar contra E, F
- G no puede jugar contra F

Si cada equipo no puede jugar más de una vez por día, ¿cual es la menor cantidad de días necesarios para poder realizar el torneo?

Solución: Se representa a cada equipo con un vértice y se unen dos vértices si ambos equipos se enfrentan en un partido. La respuesta a nuestra pregunta consiste en encontrar la mínima cantidad de colores para colorear las aristas del grafo. A continuación se muestra el grafo obtenido y un coloreo de sus aristas.



Como se puede ver en el diagrama anterior fueron necesarios 6 colores para colorear las aristas de G, por lo tanto para el torneo serán necesarios 6 días, y los partidos serán:

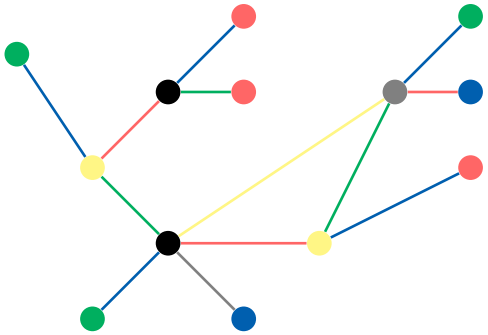
- Día-1 (verde): [A vs E] [B vs C] [D vs G]
- Día-2 (azul): [A vs F] [B vs E] [C vs G]
- Día-3 (amarillo): [B vs G] [E vs F]
- Día-4 (rosa): [A vs D] [C vs E]
- Día-5 (naranja): [A vs G] [B vs D] [C vs F]
- Día-6 (púrpura): [B vs F] [C vs D] [E vs G]

Si bien el ejemplo anterior fue solo a modo didáctico, hay aplicaciones importantes en área de *sports scheduling* como es el caso la liga de Football Americano **NFL** que utiliza una algoritmo de coloreo de aristas para asignar los partidos a cada fin de semana.

3.14. Coloreo Total

Terminamos esta sección con una última definición que considera simultáneamente el coloreo de vértices y el coloreo de aristas.

Definición 3.14.1. Coloreo Total. Se quieren colorear simultáneamente todos los vértices y todas las aristas de G de modo tal que vértices y aristas adyacentes tengan colores diferentes, y además cada vértice y sus respectivas aristas tengan colores diferentes. La siguiente imagen muestra un ejemplo.



Definición 3.14.2. El **número cromático total** es la mínima cantidad de colores necesarios para realizar un coloreo total de un grafo G. Se nota $\chi''(G)$.

Proposición 3.14.1. Para todo grafo G se tiene: $\chi''(G) \geq 1 + \Delta(G)$.

Demostración. Sea v un vértice de grado $\Delta(G)$. Luego se le debe asignar un color distinto a cada una de las $\Delta(G)$ aristas y un color más para v. □

Conjetura sobre coloreo total: En los años '60 Vizing propuso que para todo grafo G se tiene: $\chi''(G) \leq 2 + \Delta(G)$.

Teorema 3.14.2. Para todo grafo G vale: $\chi''(G) \leq \chi(G) + \chi'(G)$.

Demostración. Tomamos un k-coloreo de vértices de G con los valores $1, ..., k$ y un j-coloreo de aristas de G con los valores $k+1, ..., k+j$. Aplicando ambos coloreos simultáneamente obtenemos la cota buscada pues como el coloreo de vértices y el coloreo de aristas no comparten colores entre sí, obtenemos un coloreo total válido pero posiblemente no óptimo. □

Teorema 3.14.3. Para todo $r \in \mathbb{N}$ se cumple:

$$\chi''(K_{r,r}) = \chi(K_{r,r}) + \chi'(K_{r,r})$$

Demostración. Sea $G = K_{r,r}$. Notemos $k := \chi''(G)$.
Es fácil ver que $|E(G) \cup V(G)| = r^2 + 2r$ pues por definición $K_{r,r}$ tiene $2r$ vértices y cada uno de los r vértices de una de las biparticiones está conectado con todos los vértices de la otra, es decir tiene r vecinos. Luego en total se tiene r^2 aristas.

Luego el conjunto $E(G) \cup V(G)$ puede ser particionado en conjuntos X_1, ..., X_k. Cada uno compuesto por vértices independientes, aristas independientes y tales que ninguna arista incide en uno de los vértices. Como consecuencia de estas restricciones se tiene que $|X_i| \leq r \, \forall i$ pues es claro que el cardinal del conjunto independiente máximo tanto de vértices como de aristas es r, si se tuvieran más de r elementos en total debería haber una arista que incide un uno de los vértices seleccionados. Luego $k \geq r + 2$ pues como en total hay k conjuntos X_i tenemos que $kr \geq r^2 + 2r$.

Ahora aplicando el teorema de König (3.10.2) tenemos que $\chi'(K_{r,r}) = r$. Además por ser un grafo bipartito sabemos que $\chi(K_{r,r}) = 2$ (prop. 3.3.9). Entonces por el teorema (3.14.2) $\chi''(K_{r,r}) \leq r + 2$. Juntando las dos desigualdades anteriores obtenemos:

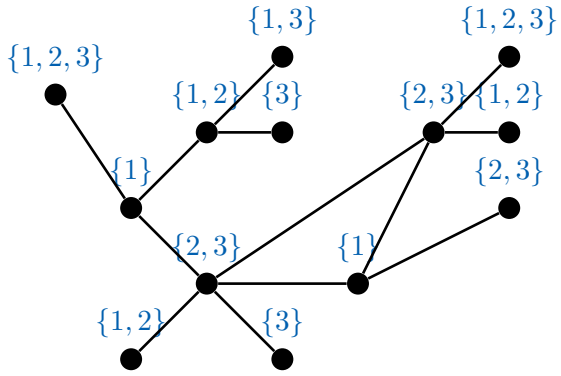
$$\chi''(K_{r,r}) = r + 2 = \chi(K_{r,r}) + \chi'(K_{r,r})$$

□

Capítulo 4

Variantes del Problema de Coloreo

El concepto central en esta sección es **coloreo por listas**, introducido en los años '70 independientemente por Vizing [3] y por el grupo Erdős, Rubin, Taylor [4]. Pero comenzó a hacerse popular apenas a inicios de los '90. Desde entonces ha capturado el interés de muchos investigadores y se ha vuelto un área de gran importancia.



4.1. Coloreo por Listas

Esta es una variante del problema de coloreo que permite agregar restricciones adicionales y de esta forma ser más flexible para modelar aplicaciones prácticas. Esencialmente la idea consiste en asignarle a cada vértice i una lista L_i de posibles colores y luego buscar un k-coloreo de G sujeto a las restricciones $c(i) \in L_i \; \forall i \in V$. Notar que si para todo vértice i se tiene la lista $L_i = \{1, ..., k\}$ el problema se reduce al k-coloreo clásico.

4.2. μ-coloreo

Consiste en un caso particular de coloreo por listas donde se tiene una función $\mu : V \to \mathbb{N}$. Se quiere colorear a G de modo que $c(i) \leq \mu(i)$ donde $c(i)$ es la función de coloreo. Se obtiene así una lista de colores para cada vértice pero menos flexible que antes. La ventaja como veremos más adelante es que la pérdida de flexibilidad es compensada por una reducción en la dificultad de resolución. Notar que si se toma $\mu(i) \equiv k$ se está en el caso de k-coloreo clásico.

A continuación introducimos el concepto de cografo, y luego mostramos que se puede proporcionar una formulación equivalente en términos de μ-coloreo.

Definición 4.2.1. Un **cografo** es un grafo que puede ser construido usando las reglas siguientes:

1. el grafo compuesto por un solo vértice es un cografo.
2. si G es un cografo, también lo es su complemento $\overline{G}$.
3. sean G, H cografos, luego su unión disjunta (2.3.8) también los es.

Caracterización: Los cografos son los grafos libres de P_4.

Vimos antes que un grafo G es **perfecto** si para todo subgrafo inducido H se cumple: $\omega(H) = \chi(H)$. Existe otra manera de definir los grafos perfectos en términos de coloreo:

Proposición 4.2.1. La siguiente definición alternativa de grafo perfecto es equivalente a la usual. [18]

Definición 4.2.2. Grafos Perfectos. G es perfecto si para todo subgrafo inducido H y para todo k entero positivo se cumple que:
H es k-coloreable si y solo si toda clique de H es k-coloreable. [18]

Basándonos en la definición alternativa de grafos perfectos, proponemos una generalización del concepto reemplazado al k-coloreo por un μ-coloreo.

Definición 4.2.3. Grafos M-perfectos. G es M-perfecto si para todo subgrafo inducido H y para toda función $\mu : V \to \mathbb{N}$ se cumple que:
H es μ-coloreable si y solo si toda clique de H es μ-coloreable. [18]

Observación: Los grafos M-perfectos son perfectos pues k-coloreo es un caso particular de μ-coloreo. Es decir como la definición anterior pide que se cumpla para toda función μ, en particular tomamos $\mu \equiv k$ que es exactamente k-coloreo y por lo tanto obtenemos la definición alternativa de grafo perfecto.

Definición 4.2.4. Un coloreo c es **minimal** cuando para cada vértice v y para cada color i tal que $i \leq c(v)$, se cumple que v tiene un vecino w_i con $c(w_i) = i$.

Observación: Para obtener un k-coloreo minimal basta con aplicar el algoritmo *greedy* (ver 5.5) con los vértices ordenados de manera de producir un coloreo óptimo.

Lema 4.2.2. Sean G un cografo, x uno de sus vértices, c_m un coloreo minimal de $G - x$, T un entero positivo. Si c_m no puede ser extendido a G de manear que $c_m(x) \leq T$, luego existe un subconjunto de vértices $H \subseteq N(x)$ de cardinal T y tal que $c_m(H) = \{1, ..., T\}$, más aún el conjunto H forma un subgrafo completo. [18]

Teorema 4.2.3. G es M-perfecto **si y solo si** G es un cografo. [18]

Demostración. (M-perfecto $\Rightarrow$ Cografo) Supongamos que los vértices v_1, v_2, v_3, v_4 inducen un P_4. Sea μ definida como:

$$\mu(v_1) = \mu(v_4) = 1$$

$$\mu(v_2) = \mu(v_3) = 2$$

notar que cada clique es μ-coloreable, pero P_4 no lo es. Abs.

(Cografo $\Rightarrow$ M-perfecto) Supongamos que existe un grafo G libre de P_4 y que no es M-perfecto. Supongamos además que G es mínimo, es decir que $G - v$ es M-perfecto $\forall v \in V$. Sea $\mu(v)$ tal que todas las cliques de G son μ-coloreables pero G no lo es. Sea x un vértice de G con $\mu(x)$ máximo. El grafo $G-x$ es M-perfecto y como las cliques de G son μ-coloreables, entonces también lo son las cliques de $G - x$. Por lo tanto $G - x$ es μ-coloreable.
Sea c_m un μ-coloreo minimal de $G - x$. Como G no es μ-coloreable, c_m no puede ser extendido a un μ-coloreo de G. Por lo tanto por el lema 4.2.2

$N(x)$ contiene un subgrafo completo K de tamaño $\mu(x)$. Si consideramos ahora $H = K \cup x$, vemos que H es completo (y de tamaño $\mu(x) + 1$) pues x es adyacente a todos los vértices de K porque son sus vecinos. Es decir que G contiene un subgrafo completo H de tamaño $\mu(x) + 1$ donde $\mu(v) \leq \mu(x)\ \ \forall v \in H$ pues tomamos $\mu(x)$ máximo. Esto produce una contradicción pues todas las cliques de G son μ-coloreables. □

Algoritmo: En el capítulo siguiente presentamos un algoritmo para μ-colorear cografos. Ver sección 5.9.

4.3. (γ, μ)-coloreo

Esta es una generalización de μ-coloreo donde tenemos también una cota inferior. Dadas funciones μ, $\gamma : V \to \mathbb{N}$, la condición que debe cumplirse es: $\gamma(v) \leq c(v) \leq \mu(v)$. Notar que si se toma $\gamma\,(v) \equiv 1$ se está en el caso de μ-coloreo. [19]

4.4. Pre-coloreo

Sean G grafo y H subgrafo de G. Primero se colorea al grafo H y luego se intenta extender el coloreo a todo G. Es decir si c' es un coloreo de H, se quiere obtener una función c que sea un k-coloreo de G tal que: $c(v) = c'(v)\, \forall\, v \in H$. Claramente un k-coloreo es un caso particular de pre-coloreo tomando $H = \emptyset$. Además un pre-coloreo es un caso particular de (γ, μ)-coloreo tomando $\gamma(i) = \mu(i)$ para aquellos vértices que deben estar pre-coloreados. El concepto de pre-coloreo surgió durante el estudio de un problema práctico para asignar planes de vuelo a una flota de aviones con la restricción de que cada avión debe someterse a rutinas de mantenimiento preventivo [13] [14] [15].

Observación: Es interesante notar que el problema de coloreo por listas puede ser reducido a un problema de pre-coloreo en tiempo polinomial. El modo de hacer esto es: dado el vértice i, para construir la lista L_i, se agrega un nuevo vértice por cada color que i no puede tener, se lo colorea de ese color y se conecta con i.
Pero la gran diferencia entre esta reducción y las anteriores es que con este método estamos modificando al grafo y no todas las propiedades se conservan pues ciertas clases de grafos no son cerradas con respecto a esta operación.

A continuación se presenta una aplicación práctica del problema de Pre-Coloreo.

4.5. Resolución del juego Sudoku con pre-coloreo

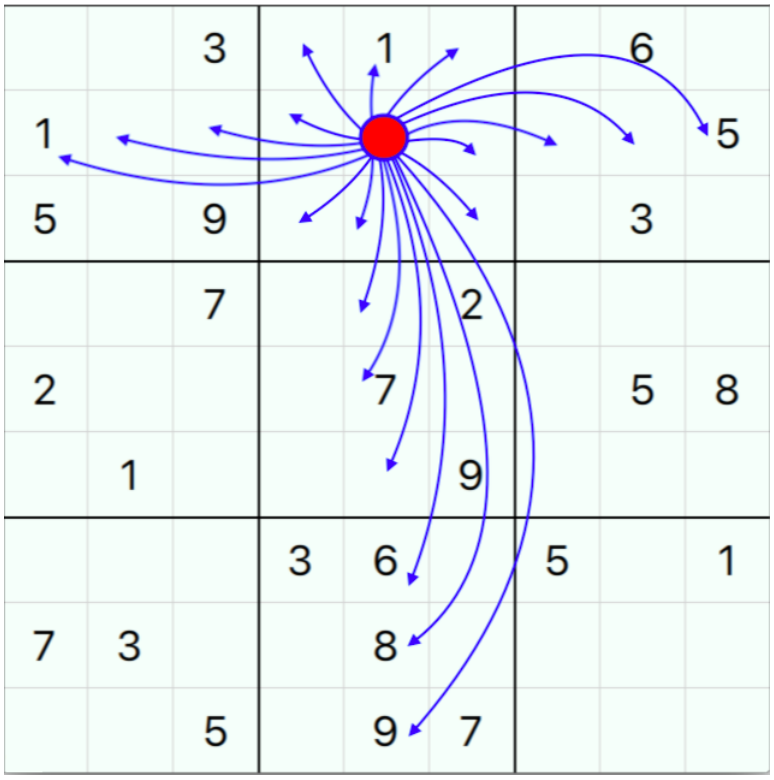
El Sudoku consiste en un tablero cuadriculado de 9×9 celdas. El objetivo es numerar cada celda con los valores $\{1, ..., 9\}$ de modo tal que:

- no pueden repetirse números en la misma fila
- no pueden repetirse números en la misma columna
- el tablero está a su vez dividido en 9 bloques de 3×3, y no pueden repetirse números dentro del mismo bloque.

Al comenzar el juego el tablero ya tiene algunas celdas pre-numeradas de modo tal que el jugador quede sujeto a ciertas restricciones.

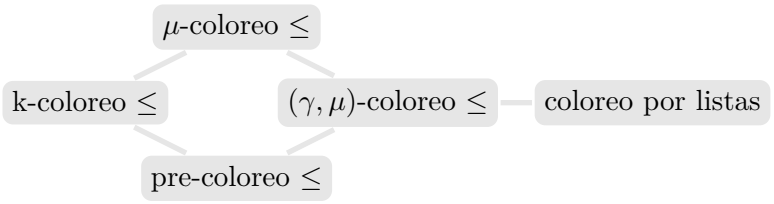
Modelo utilizado: La idea es muy simple. Se representa a cada celda con un vértice, y se unen dos vértices si no se les puede asignar el mismo número. Las celdas que están pre-numeradas son justamente los vértices pre-coloreados.
Entonces encontrar una extensión del pre-coloreo del grafo es equivalente a resolver el Sudoku.

La imagen siguiente muestra un ejemplo de las aristas entre un vértice (marcado con un punto rojo) y todos los demás vértices. De manera similar se construyen las otras aristas.


3 1 6
1 5
5 9 3
7 2
2 7 5 8
1 9
3 6 5 1
7 3 8
5 9 7

4.6. Comparación entre los distintos problemas

El siguiente diagrama muestra el orden de dificultad de manera creciente de las distintas variantes de coloreo.



Como conclusión tenemos que si para una familia de grafos el problema de coloreo por listas puede ser resuelto en tiempo polinomial, entonces también podrán ser resueltas en tiempo polinomial las otras variantes de coloreo. Mientras que si para una familia de grafos el problema de k-coloreo es NP-Completo, entonces será NP-Completa cualquiera de las otras variantes.

4.7. Comentarios sobre la complejidad

Si bien el problema de coloreo es NP-Completo [7], existen clases de grafos donde el problema puede ser resuelto en tiempo polinomial, como por ejemplo en el caso de los grafos perfectos [7]. Pero al analizar la complejidad para el problema de coloreo por listas, se encuentra que para grafos perfectos en general el problema sigue siendo NP-Completo.
A continuación presentamos distintas clases de grafos y analizamos la complejidad de las diferentes variantes del problema de coloreo.

Grafos de Intervalos

Como los grafos de intervalos son grafos perfectos [22] luego k-coloreo puede ser resuelto en tiempo polinomial. Mientras que pre-coloreo y μ-coloreo son NP-completos [23][19] por lo tanto también los son (γ, μ)-coloreo y coloreo por listas.

Grafos Bipartitos

Claramente un grafo bipartito puede ser coloreado en tiempo polinomial. Mientras que pre-coloreo y coloreo por listas son NP-completos [16]. De hecho se tiene un resultado aún más fuerte, coloreo por listas es NP-completo para grafos bipartitos completos [21].
Vamos a probar que μ-coloreo es NP-Completo y por lo tanto también lo es (γ, μ)-coloreo.

Teorema 4.7.1. μ-coloreo es NP-Completo para grafos bipartitos. [18]

Demostración. Dado que coloreo por listas es NP-Completo [16], vamos a construir una reducción polinomial de un problema de coloreo por listas de un grafo G a un problema de μ-coloreo de un grafo G'.
Sea X, Y una bipartición de G. Sea $L(v)$ una función que devuelve las listas asociadas a cada vértice. Definimos $k := |\cup_{v \in V} L(v)|$. Podemos asumir sin pérdida de generalidad que $\cup_{v \in V} L(v) = \{1, ...k\}$. Para construir el nuevo grafo G' agregamos a G dos conjuntos X' e Y' de cardinal k cada uno, de modo que X, X', Y, Y' resulten disjuntos dos a dos. G' también será un grafo bipartito con bipartición $X \cup X'$ e $Y \cup Y'$. Ahora dados $x'_i \in X'$ e $y'_j \in Y'$ agregamos una arista entre ambos si y solo si $i \neq j$. Por otro lado x'_i se conecta con todos los $y \in Y$ talas que $i \notin L(y)$, de manera análoga se conecta y'_i con todos los $x \in X$ tales que $i \notin L(x)$. Definimos al función μ como: $\mu(x) = \mu(y) = k$ y $\mu(x'_i) = \mu(y'_i) = i$. Notar que entonces los vértices x'_i e y'_i reciben el color i. Pues para x'_1 no hay otra alternativa que recibir el color 1 (idem para y'_1), para x'_2 tenemos que $c(x'_2) \leq \mu(x'_2) = 2$ es decir que solo puede recibir el color 1 o 2, pero como x'_2 es adyacente a y'_1 entonces $c(x'_2) = 2$ (idem para y'_2). Para x'_3 el razonamiento es análogo pues es adyacente a y'_1 e y'_2. Entonces si bien el color de todos los vértices de X' e Y' están predefinidos, tenemos que describir este hecho usando una función μ de otra forma no se estaría realizando una reducción a un μ-coloreo.
Notar que entonces los conjuntos X' e Y' contiene ambos todos los colores que podemos utilizar, luego el rol de estos conjuntos es el de formar una lista de colores prohibidos para cada vértice de G, pues recordar que conectamos a cada vértice $x \in X$ con todos los vértices de Y' que no están en su lista, que es lo mismo que decir con todos los colores que no pueden ser usados.
De este modo se obtuvo una trasformación polinomial tal que G es coloreable por listas si y solo si G' es μ-coloreable. □

Grafos bipartitos completos

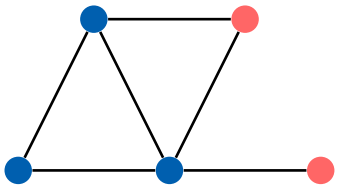
Teorema 4.7.2. Para grafos bipartitos **completos** (γ, μ)-coloreo puede ser resuelto en tiempo polinomial.

Demostración. En la sección 5.12 se presenta un algoritmo polinomial para resolver el problema. □

Grafos Split

Definición 4.7.1. Un grafo G se dice **split** si su conjunto de vértices puede ser particionado en una clique y un conjunto independiente. Si el grafo contiene todas las posibles aristas entre vértices de la clique y vértices del conjunto independiente, entonces decimos que es un grafo *split* completo.
Observar que la partición puede no ser única.

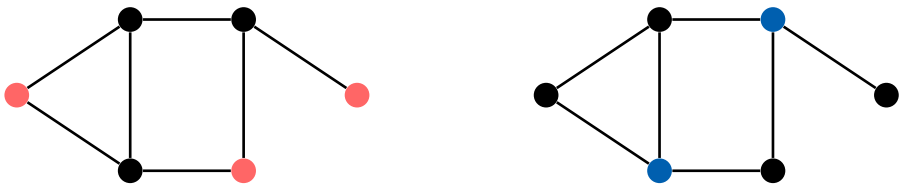
La imagen siguiente muestra un ejemplo. Los vértices que componen la clique se colorearon de azul y los que forman el conjunto independiente de rojo.



Para grafos *split* se produce la situación interesante donde pre-coloreo y μ-coloreo tienen diferente complejidad (a menos que $P = NP$). El problema de pre-coloreo en grafos *split* puede ser resuelto en tiempo polinomial. [17]
Mientras que μ-coloreo es NP-Completo para grafos *split*. [19]
Por lo tanto también son NP-Completos (γ, μ)-coloreo y coloreo por listas.

Vamos a probar que μ-coloreo es NP-Completo realizando una reducción polinomial del problema de **conjunto dominante** (ver def. a continuación) que es NP-Completo en grafos *split*. [20]

Definición 4.7.2. Dado un grafo $G(V, E)$, decimos que un conjunto $D \subseteq V$ es un **conjunto dominante** si todo vértice de G o bien pertenece a D o es vecino de un vértice en D. Notar que puede existir más de una forma de construir un conjunto dominante, como se puede apreciar en la siguiente imagen. Definimos también el **número dominante** denotado con $\gamma(G)$, como el cardinal de un conjunto dominante más pequeño posible.



En este caso $\gamma(G) = 2$.

Problema del conjunto dominante: Dado un grafo G y un entero positivo k, se quiere saber si $\gamma(G) \leq k$.

Teorema 4.7.3. μ-coloreo es NP-Completo para grafos *split*. [19]

Demostración. Vamos a construir una reducción polinomial del problema de conjunto dominante de cardinal k al problema de μ-coloreo.
Sean k un entero positivo y $G(V, E)$ un grafo *split* conexo con $V = K \cup I$ donde K es completo e I un conjunto independiente. Podemos asumir que

$k \leq |K|$ pues claramente el conjunto K resulta dominante porque I es un conjunto independiente y G es conexo. Dicho de otro modo los vértices de I no son adyacentes entre sí, y para cada vértice $w \in I$ existe un vértice $v \in K$ tal que v, w son adyacentes.
Vamos a construir un nuevo grafo G' y una función $\mu : V_{G'} \to \mathbb{N}$ tal que G' es μ-coloreable si y solo si G admite un conjunto dominante de cardinal a lo sumo k.

Grafo G'

- $V_{G'} = V_G$
- K es completo en G'
- I es un conjunto independiente en G'
- dado $v \in K$ y $w \in I$: $(v, w) \in E(G') \Leftrightarrow (v, w) \notin E(G)$

Función μ

$$\mu(v) := \begin{cases} |K| \, si \; v \in K \\ k \quad si \; v \in I \end{cases}$$

($\Leftarrow$) Comenzamos por asumir que G admite un conjunto dominante D con $|D| \leq k$. Como G es conexo entonces un tal conjunto D puede ser tomando de manera que $D \subseteq K$ pues si el conjunto D seleccionado posee vértices en I, estos pueden ser reemplazados por uno de sus vecinos en K.

Realizamos un μ-coloreo del siguiente modo:

- colorear los vértices de D usando los colores $\{1, ..., |D|\}$
- colorear los vértices de $K - D$ usando los colores $\{|D| + 1, ..., |K|\}$
- $\forall v \in I$ tomar $v' \in D$ adyacente en G, por lo tanto $(v, v') \notin E(G')$
- colorear v con el color de v'

($\Rightarrow$) Supongamos que G' es μ-coloreable. Sea $c : V_{G'} \to \mathbb{N}$ su μ-coloreo. Por lo dicho antes $\mu(v) = |K| \; \forall v \in K$ y K es completo en G', luego $c(K) = \{1, ..., |K|\}$. Además tenemos que $k \leq |K|$ entonces $\forall w \in I \; \exists w' \in K$ tal que $c(w) = c(w') \leq k$. Luego $(w, w') \notin E(G')$ entonces $(w, w') \in E(G)$. Por lo tanto $\{v \in K : c(v) \leq k\}$ es un conjunto dominante de cardinal k. □

Grafos Split Completos: Si agregamos como hipótesis que los grafos *split* sean además completos entonces (γ, μ)-coloreo puede ser resuelto en tiempo polinomial [19], mientras que para coloreo por listas el problema sigue siendo NP-Completo (Jansen y Scheffler). [21]

Cografos

Para cografos pre-coloreo puede ser resuelto en tiempo polinomial [17], mientras que coloreo por listas es NP-Completo [21].
En la sección 5.9 probamos que μ-coloreo es polinomial utilizando el algoritmo greedy (RS).

Grafos Línea

Recordar que coloreo de un grafo línea es equivalente al problema de coloreo de aristas de un grafo. Si bien este problema en general es NP-Completo, puede ser resuelto en tiempo polinomial para grafos completos y para grafos bipartitos [26]. Además sean $L(K_n)$ el grafo línea de un grafo completo y $L(K_{n,n})$ el grafo línea de un grafo bipartito completo. Se tiene que en ambos casos μ-coloreo y pre-coloreo son NP-Completos. [19]
Por lo tanto también son NP-Completos (γ, μ)-coloreo y coloreo por listas.

4.8. Resumen de los resultados

La tabla siguiente muestra los distintos tipos de grafos mencionados anteriormente y sus respectivas complejidades según la variante de coloreo elegida.

	k-col	pre-col	μ-col	(γ, μ)-col	col-listas
grafo intervalos	P	NP-C	NP-C	NP-C	NP-C
grafo bipartito	P	NP-C	NP-C	NP-C	NP-C
bipartito completo	P	P	P	P	NP-C
grafo split	P	P	NP-C	NP-C	NP-C
split completo	P	P	P	P	NP-C
cografo	P	P	P	?	NP-C
linea $L(K_n)$	P	NP-C	NP-C	NP-C	NP-C
linea $L(K_{n,n})$	P	NP-C	NP-C	NP-C	NP-C

Notar que la complejidad de (γ,μ)-coloreo para cografos es un problema abierto.

4.9. Número de elección

Definición 4.9.1. Dado un grafo G y una función $f : V \to \mathbb{N}$, decimos que G es **f-elegible** si existe un coloreo por listas de G para cualquier conjunto de listas que cumple la condición: $|L_v| = f(v) \, \forall v \in V$. Es decir que solo interesa el cardinal de las listas y no como están compuestas.
En particular si $f(v) \equiv k$ decimos que G es **k-elegible**.

Definición 4.9.2. El **número de elección**, es el menor entero positivo k tal que G es k-elegible y lo notamos $ch(G)$.

Claramente vale la desigualdad $\chi(G) \leq ch(G)$. El siguiente ejemplo muestra un caso donde el número de elección es estrictamente mayor.

Ejemplo: Si $G = K_{3,3}$, entonces $\chi(G) < ch(G)$.
Sabemos que $\chi(K_{3,3}) = 2$ por ser un grafo bipartito. Notemos $\{v_1, v_2, v_3\}$ y $\{w_1, w_2, w_3\}$ a los vértices de cada uno de los conjuntos independientes de G. Es fácil ver a mano que si tomamos como listas: $L(v_i) = L(w_i) = \{1, 2, 3\} \setminus \{i\}$ entonces $K_{3,3}$ no es 2-elegible.

El siguiente teorema generaliza el ejemplo anterior, siendo este último el caso $k = 2$.

Teorema 4.9.1. Para todo entero $k \geq 2$ existe un grafo bipartito G con número de elección $ch(G) > k$.
En particular vamos a probar que el grafo bipartito completo $K_{m,m}$ con $m = \binom{2k-1}{k}$ no es m-elegible. [4]

En esencia lo que nos dice el teorema es que existen grafos con número cromático $\chi(G) = 2$ y número de elección $ch(G)$ arbitrariamente grande.

Demostración. Comenzamos por notar que m representa la cantidad de subconjuntos de k elementos de un conjunto de $2k - 1$ elementos. Sea $\{1, ..., 2k-1\}$ el conjunto de colores. Como es habitual, notamos V_1, V_2 a los conjuntos independientes de G. Consideramos las listas de colores compuesta por cada uno de los m subconjuntos de k elementos, y se las asignamos a los m vértices de V_1. Independientemente para V_2. Observar que los vértices de V_1 deben ser coloreados con k colores distintos de lo contrario como hay $2k - 1$ colores disponibles en total, habría por lo menos k colores no utilizados y entonces podríamos tomar un conjunto E de cardinal k formado por algunos de estos colores. Necesariamente habría un vértice $v_0 \in V_1$ al cual se le asignó el conjunto E. Llegando así a un absurdo pues todos los colores posibles de v_0 son justamente los colores no utilizados. Concluimos entonces que V_1 debe ser coloreado con k colores. Si ahora consideramos el conjunto

formado por los k colores usados para V_1 vemos que por ser un conjunto de cardinal k, necesariamente debe haber un vértice en V_2 que tiene asignado ese conjunto, pero entonces no es posible asignarle un color, pues es adyacente a todos los vértices de V_1. Por lo tanto $K_{m,m}$ no es k-elegible. □

Definición 4.9.3. (f, g)-elegible. La siguiente definición generaliza el concepto de f-elegibilidad. Dadas dos funciones $f, g : V \to \mathbb{N}$ con $g(v) \leq f(v)\ \forall v$ y un grafo G, decimos que G es **(f,g)-elegible** si para cualquier colección de listas de colores con $|L_i| = f(i)\ \forall i$ existe un subconjunto $S_i \subseteq L_i$ de cardinal $|S_i| = g(i)$ tal que $S_i \cap S_j = \emptyset$ para todo par de vértices adyacentes. Notar que si se toma $g \equiv 1$ se está en el caso de f-elegibilidad.

Proposición 4.9.2. Para todo grafo G se cumple: $ch(G) \leq \Delta(G) + 1$.

En efecto, si tomamos para cada vértice una lista de longitud $\Delta(G) + 1$ podemos usar el algoritmo *greedy* introducido en la sección (5.5) para colorear a G. Es decir que G resulta $(\Delta(G) + 1)$-elegible.

Proposición 4.9.3. Todo árbol es 2-elegible.

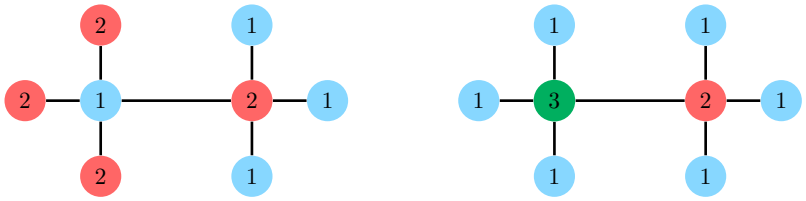
Demostración. Se puede ver fácilmente por inducción. Sea n la cantidad de vértices de G, claramente la propiedad vale para $n = 1, 2$. Supongamos que vale para k y veamos que vale para $k + 1$. Para esto basta con quitar una hoja de G (recordar que todo árbol tiene al menos dos hojas [6]), ahora nos queda un grafo de k vértices y por hipótesis inductiva es 2-elegible. Solo nos falta colorear la hoja que quitamos, pero esto es posible pues tiene por definición un solo vecino y dos colores posibles para elegir. □

4.10. Coloreo Suma

Esta es una variante del problema de coloreo con la particularidad que no se intenta minimizar la cantidad de colores utilizados.

Definición 4.10.1. Sea $G(V, E)$ un grafo, definimos la **suma cromática** de G, y la notamos $\sum(G)$ como la mínima suma sobre todos los posibles coloreos de G, $\sum_{v \in V} c(v)$.

Notar que para obtener una suma mínima puede ser necesario usar más colores que $\chi(G)$ como lo muestra la siguiente figura de un *caterpillar* con número cromático $\chi(G) = 2$.
El 2-coloreo tiene un suma: $\sum_{v \in V} c(v) = 12$
Mientras que la mínima suma es $\sum(G) = 11$, pero requiere de un 3-coloreo.



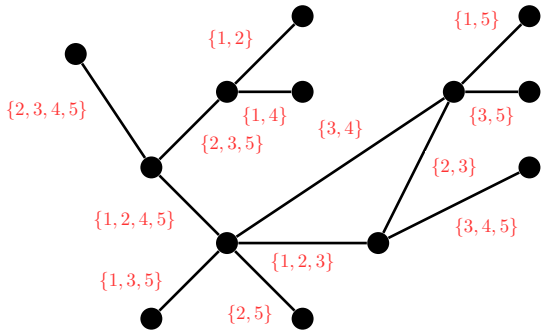
Complejidad: Dado un grafo G y un entero positivo K, el problema de decisión: "¿existe un c coloreo de G tal que $\sum_{v \in V} c(v) \leq K$?" es NP-Completo. [44]

La diferencia entre el número cromático de un grafo $\chi(G)$ y la cantidad de colores necesaria para obtener un coloreo de suma mínima puede ser arbitrariamente grade.

Proposición 4.10.1. Sea T un árbol, sabemos que $\chi(T) = 2$. Dado K entero positivo, existe un árbol T_K tal que su coloreo de suma mínima requiere de K colores. [44]

4.11. Coloreo de aristas por listas

De manera análoga al coloreo por listas de vértices, se define el coloreo por listas de las aristas de G.



Definición 4.11.1. El menor entero k para el cual G tiene un coloreo de aristas para cualquier familia de listas de cardinal k, se denomina **índice de elección** y lo notamos $ch'(G)$.

Formalmente se tiene $ch'(G) = ch(L(G))$ con $L(G)$ el grafo línea de G.

Vimos anteriormente (Teorema 4.9.1) que el número cromático y el número de elección de un grafo pueden diferir en una cantidad arbitraria. Pero para coloreo de aristas no se conoce un ejemplo donde esto suceda. De hecho se tiene la siguiente conjetura.

Conjetura de coloreo por listas: $\forall G,\ ch'(G) = \chi'(G)$.

Capítulo 5

Algoritmos de Coloreo

Al colorear un grafo se intenta usar la menor cantidad de colores posible, idealmente $\chi(G)$. Pero desafortunadamente el problema de coloreo es NP-Completo. Este hecho produjo el desarrollo tanto de técnicas para acelerar el cómputo de los algoritmos exactos, como también heurísticas y algoritmos 'greedy' para obtener coloreos subóptimos. En el segundo caso, el objetivo será el de buscar una manera veloz de colorear a G sin usar una cantidad de colores excesiva.

En esta sección se presentan algunos de los algoritmos más importantes para colorear grafos, tanto exactos como aproximados. El motivo por el cual existe una amplia variedad es que para cada algoritmo hay grafos para los cuales el método no se desempeña satisfactoriamente, ya sea usando demasiados colores o requiriendo más tiempo del disponible por el usuario.

5.1. Algoritmo de Conexión-Contracción para calcular $\chi(G)$

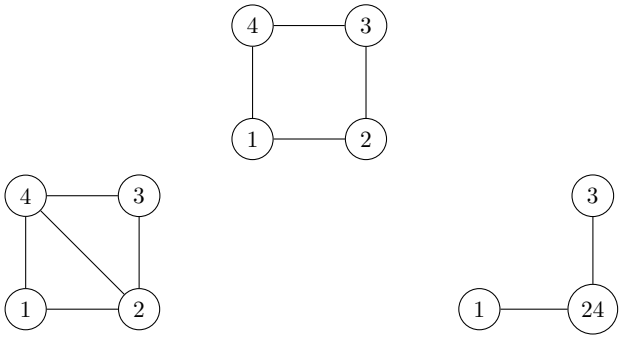
Presentamos primero un algoritmo **exacto** para calcular $\chi(G)$. Esta es una idea ingeniosa que produce un algoritmo muy simple y claro de entender, pero por su complejidad computacional es prácticamente imposible de usar, salvo para grafos muy pequeños.

El siguiente algoritmo es de tipo recursivo. Vamos a usar el hecho de que $\chi(K_n) = n$ como caso base (ver prop. 3.3.1).
Si G no es completo existen dos vértices v, w que no so adyacentes. Luego se presentan dos posibilidades o a v, w se les asigna el mismo color, o se les asignan colores diferentes. (Claramente son dos casos disjuntos). Si se les asignaron colores diferentes, de un punto de vista de coloreo, colorear a G es equivalente a colorear al grafo que es igual a G pero además tiene la arista (v, w), lo notamos G_1. Llamamos a este proceso **conexión**.

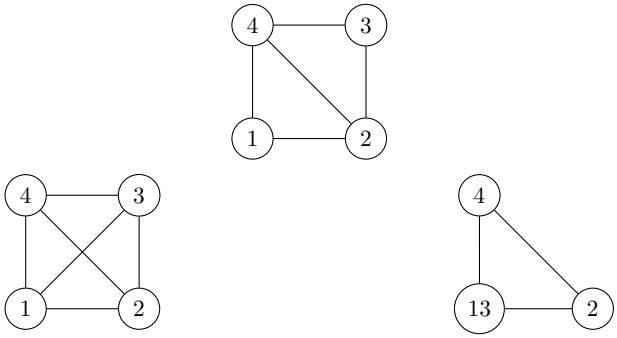
Por otra parte si a v, w se les asignó el mismo color, luego de un punto de vista exclusivamente de coloreo, es lo mismo colorear a G que colorear al grafo que es igual a G pero fusionando los vértices v, w en uno solo, lo notamos G_2. Llamamos a este proceso **contracción**. Claramente $\chi(G) = min\{\chi(G_1), \chi(G_2)\}$ (resultado conocido como el teorema de Zykov [35]). Se repite el proceso en los grafos G_1, G_2 recursivamente hasta obtener todos grafos completos.
A continuación se muestra un implementación en Python3 usando la librería open sorce networkx [32].

```
# Algoritmo de Conexion-Contraccion
#    es G completo?
#        si: caso base.
#            Return = numero de vertices de G
#        no: recursion.
#            Return = min{cc(G.Conexion), cc(G.Contraccion)}

import networkx as nx

def cc(G):
    n = len(G.nodes)
    (u, v) = (0, 0)
    for i in G.nodes:
        for j in G.nodes:
            if i != j and (i, j) not in G.edges:
                (u, v) = (i, j)
                break
        if (u, v) != (0, 0):
            break
    if (u, v) == (0, 0):
        return n
    H = nx.contracted_nodes(G, u, v, self_loops=True)
    G.add_edge(u, v)
    return min(cc(H), cc(G))
```

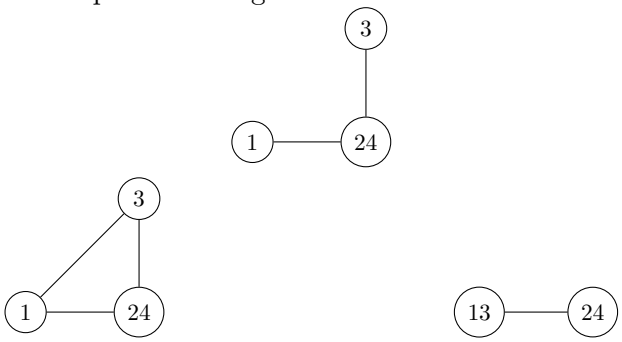
Ejemplo: La imagen siguiente muestra el caso de un ciclo C_4, el grafo obtenido por la conexión de los vértices $2, 4$ y el grafo obtenido por la contracción de los vértices $2, 4$.



Repetimos el procedimiento para cada uno de los grafos obtenidos. Esta vez conectamos y contraemos los vértices $1, 3$.



Ahora para el otro grafo:



Obtenemos así todos grafos completos: K_4, K_3, K_3, K_2. Luego $\chi(G) = 2$.

Observación: Podemos usar el algoritmo para obtener rápidamente una cota superior de $\chi(G)$, para esto en lugar de ramificar el proceso en cada paso analizando conexión y contracción, podemos directamente realizar solo la operación de contracción hasta obtener un grafo completo.

5.2. Polinomio Cromático

Dado un grafo G, definimos $P(k)$ como la cantidad de k-coloreos distintos que puede tener G. Por el momento llamemos a $P(k)$ función cromática pues a priori no es del todo claro que se trate efectivamente de un polinomio.

Observar que en el caso de $G = K_n$ es fácil calcular $P(k)$, en efecto

$$P(k) = k(k-1)(k-2)...(k-n+1)$$

pues para el primer vértice tenemos k posibles colores, para el segundo $k-1$ pues son adyacentes, y así sucesivamente. Entonces en el caso de un grafo completo $P(k)$ es un polinomio.

Notar que $P_G(k) = P_{G1}(k) + P_{G2}(k)$ con G_1, G_2 los grafos definidos en el algoritmo de Conexión-Contracción, pues estamos separando las diferentes maneras de colorear a G en casos disjuntos.

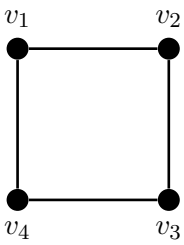
Como conclusión, para convencernos de que $P(k)$ es siempre un polinomio basta con aplicar el algoritmo de Conexión-Contracción a G. De esta manera obtenemos una colección de grafos completos, para cada uno de ellos sabemos que la función cromática es un polinomio y $\chi(G)$ es la suma de todos los números cromáticos de los grafos generados por el algoritmo.

Proposición 5.2.1. Sea T un árbol, entonces $P(k) = k(k-1)^{n-1}$.

Demostración. Claramente la proposición vale para $n = 1$. Podemos ver que vale en general por inducción. Supongamos que vale para n y consideremos un árbol de $n+1$ vértices. Si quitamos una hoja (todo árbol tiene una hoja) obtenemos un árbol de n vértices y por lo tanto puede ser coloreado de $k(k-1)^{n-1}$ formas. Ahora agregamos la hoja y como tiene un solo vecino, podemos colorear este vértice de $k-1$ maneras, luego en total tenemos $P(k) = k(k-1)^n$ como queríamos. □

5.3. Polinomio cromático de un grafo ciclo

Comenzamos por calcular el polinomio cromático del ciclo C_4.



Sea k la cantidad de colores disponible. Para v_1 tenemos k posibilidades. Ahora analizamos v_2 y v_4, ambos son vecinos de v_1 y por lo tanto deben usar un color diferente del elegido para v_1. Se presentan dos posibilidades:

(1) Usamos el mismo color para v_2 y para v_4, es decir que tenemos $k-1$ elecciones posibles para v_2 y luego el color de v_4 queda automáticamente prefijado. Entonces para v_3 quedan $k-1$ elecciones posibles. Luego hay $k(k-1)^2$ formas posibles de colorear el ciclo C_4 de manera que los vértices v_2 y v_4 tengan el mismo color.

(2) Los vértices v_2 y v_4 reciben colores diferentes. Entonces tenemos $k-1$ elecciones posibles para v_2 y $k-2$ elecciones posibles para v_4. Como v_3 es vecino de v_2 y v_4, le quedan $k-2$ colores posibles. Entonces en este caso tenemos: $k(k-1)(k-2)^2$ formas para colorear el ciclo.

Como los dos casos analizados anteriormente son claramente disjuntos, entonces tenemos que todos las formas posibles de colorear el ciclo C_4 son:

$$k(k-1)^2 + k(k-1)(k-2)^2$$

$$= (k-1)^4 + (k-1)$$

□

Veamos ahora una fórmula para C_n con n arbitrario.

Proposición 5.3.1. El polinomio cromático de un grafo ciclo C_n es:

$$P(k) = (k-1)^n + (-1)^n(k-1)$$

Demostración. Comenzamos presentando algunas notaciones.
Notamos $P(G,k)$ al polinomio cromático de G.
Sea e un arista de G, notamos $(G-e)$ el grafo que se obtiene al quitar la arista e de G, y notamos $(G\cdot e)$ al grafo obtenido al realizar la operación de contracción de los vértices de la arista e.
Luego de acuerdo a lo visto anteriormente tenemos que:

$$P(G-e,k) = P(G,k) + P(G\cdot e,k)$$

o equivalentemente:

$$P(G,k) = P(G-e,k) - P(G\cdot e,k)$$

Para probar que $P(C_n,k) = (k-1)^n + (-1)^n(k-1)$, usamos inducción en n. Para $n=3$ es claro que:

$$\begin{aligned} P(C_3,k) &= k(k-1)(k-2) \\ &= (k-1)(k^2-2k) \\ &= (k-1)^3 + (-1)^3(k-1). \end{aligned}$$

Probando así el caso base.
Supongamos que vale para C_n: $P(C_n,k) = (k-1)^n + (-1)^n(k-1)$.
Veamos que vale para C_{n+1}:

$$P(C_{n+1},k) = P(C_{n+1}-e,k) - P(C_{n+1}\cdot e,k)$$

Notar que el grafo $(C_{n+1}\cdot e)$ resulta isomorfo a un ciclo C_n pues estamos realizando una contracción de dos vértices vecinos. Luego por hipótesis inductiva:

$$P(C_{n+1}\cdot e,k) = P(C_n,k) = (k-1)^n + (-1)^n(k-1)$$

Notar que $(C_{n+1}-e)$ es el camino P_{n+1}, y como es un árbol sabemos que:

$$P(C_{n+1}-e,k) = k(k-1)^n$$

Combinando las identidades obtenidas, tenemos:

$$\begin{aligned} P(C_{n+1},k) &= P(C_{n+1}-e,k) - P(C_{n+1}\cdot e,k) \\ &= k(k-1)^n - (k-1)^n - (-1)^n(k-1) \\ &= (k-1)^{n+1} + (-1)^{n+1}(k-1) \end{aligned}$$

□

Definición 5.3.1. Dos grafos que tiene el mismo polinomio cromático se dicen **cromaticamente equivalentes**. En este caso ambos tiene la misma cantidad de vértices y aristas, además de tener el mismo número cromático. Decimos que dos grafos G y H son **cromaticamente únicos** si el hecho de que $P(G) = P(H)$ implica que H es isomorfo a G. No se conoce una manera de caracterizar ni a los grafos cromaticamente equivalentes ni a los grafos cromaticamente únicos.

5.4. Coloreo Exacto con Programación Lineal

A continuación se presenta una formulación de Isabel Méndez-Díaz y Paula Zabala [38], para tratar el problema de coloreo como un problema de programación lineal entera que luego puede ser resulto por un solver de PLE, por ejemplo CPLEX.

El objetivo es desarrollar modelos que eliminen soluciones simétricas producidas por la permutación de colores. De esta manera se reducen considerablemente las ramas que deben ser exploradas y como consecuencia aumenta la velocidad del algoritmo.

Se utiliza como herramienta fundamental el *método de planos de corte*. La idea es iterativamente refinar el conjunto factible agregando desigualdades lineales denominadas *cortes*.

Dado un problema P de programación lineal **entera**, se comienza por resolver el *problema relajado*, que es igual al problema P pero en lugar de pedir que las variables sean enteras, se toman continuas. De un punto de vista geométrico el problema relajado es un politopo convexo que contiene todas las soluciones factibles.

Notar que el problema relajado se puede resolver usando *simplex* pues es un problema de programación lineal.

A menos que la solución obtenida al resolver el problema relajado sea entera, se procede a agregar nuevas restricciones (cortes) al problema relajado. Estas nuevas restricciones son tales que todas las soluciones con valores enteros las satisfacen, pero son violadas por las soluciones con valores no enteros. De esta manera la solución no es más factible y al resolver el nuevo problema se obtiene una mejor aproximación. Ahora podemos repetir el proceso agregando nuevas restricciones.

Es interesante notar que si se requiere de varias iteraciones el problema puede volverse lento de resolver debido a la acumulación de las restricciones adicionales. Este inconveniente puede evitarse quitando *cortes* introducidos previamente y que ya no son relevantes para la solución actual.

Para evitar que el tamaño del problema lineal sea demasiado grande es importante tener cotas inferiores y superiores de la solución óptima. Para obtener cotas inferiores, recordando que $\omega(G) \leq \chi(G)$, se utilizan heurísticas para encontrar cliques máximas por ejemplo (5.11). Por otra parte, para proporcionar una cota superior usamos el algoritmo DSATUR (5.7.3) que veremos más adelante.

Comenzamos presentando un modelo lo más simple posible para fijar ideas. Las variables que usaremos para nuestro modelo son: x_{ij} binarias con $i \in V$, $j \in \{1, ..., n\}$ tales que $x_{ij} = 1$ si el vértice i es coloreado con el color j y $x_{ij} = 0$ en caso contrario. Además definimos w_j con $j \in \{1, ..., n\}$ binarias tales que $w_j = 1$ si el color j es usado para colorear algún vértice.

$$f.obj: \ \ min \ \sum_{j=1}^{n} w_j$$

subto:

$$\sum_{j=1}^{n} x_{ij} = 1 \ \ \forall i \in V \tag{5.1}$$

$$x_{ij} + x_{kj} \leq w_j \ \forall (i, k) \in E, \ \ j \in \{1, ..., n\} \tag{5.2}$$

$$x_{ij}, \, w_j \in \{0, 1\} \ \ \forall \, i, j$$

La primera restricción garantiza que cada vértice tenga asociado exactamente un color.
La segunda restricción cumple dos propósitos: por un lado nos asegura que vértices vecinos reciban colores distintos, y por otro lado hace que la variable w_j valga 1 si el color j fue usado.

Si bien la formulación anterior es correcta de un punto de vista teórico, el tiempo de ejecución es demasiado extenso para cualquier aplicación real. El principal problema es que en coloreo dada una solución factible, si se permutan los colores se obtiene otra solución, es decir hay muchas simetrías. Este hecho motiva el estudio de técnicas para eliminar las simetrías. A continuación se presentan algunas alternativas.

Restricciones adicionales para eliminar soluciones equivalentes

Alternativa 1: Dado un k-coloreo de G, cualquier subconjunto de k elementos del conjunto $\{1, ..., n\}$ es otro k-coloreo válido. Para reducir la cantidad de soluciones equivalentes se agrega una restricción para que el color j pueda ser utilizado solamente si el color $j-1$ ya está siendo utilizado.
Entonces al problema lineal anterior le agregamos:

$$w_j \leq \sum_{i \in V} x_{ij} \ \ \forall j \in [1, n]$$
$$w_{j+1} \leq w_j \ \ \forall j \in [1, n-1]$$

La primera restricción asegura que si el color j no es utilizado entonces la variable que indica si se usa o no, efectivamente dice que no. Notar que si no es agrega esta restricción de todas formas el problema lineal termina con todas las variables $w_j = 0$ para todos los colores j que no son utilizados, como de hecho sucede en el modelo anterior. Esto es así porque la función objetivo pide minimizar la suma de todas las variables w_j.
La segunda restricción claramente cumple con lo que queríamos.

Alternativa 2: Este es un modelo aún más restrictivo que el anterior. Se intenta eliminar algunas de las soluciones equivalentes que surgen de la permutación de los primeros k colores. Las nuevas restricciones aseguran que el número de vértices coloreados con j sea mayor o igual al número de vértices coloreados con el color $j+1$.

$$w_j \leq \sum_{i \in V} x_{ij} \ \ \forall j \in [1, n]$$
$$\sum_{i=1}^{n} x_{ij+1} \leq \sum_{i=1}^{n} x_{ij} \ \ \forall j \in [1, n-1]$$

Alternativa 3: Dada una partición del conjunto de vértices en conjuntos independientes, es claro que si se permutan los colores entre conjuntos independientes del mismo cardinal se obtienen soluciones simétricas para el modelo anterior.

Para eliminar estas soluciones se propone indexar cada conjunto independiente con el índice del vértice de menor índice dentro del conjunto y exigir que el conjunto independiente j-esimo deba recibir el color j-esimo.
De esta manera todas las otras permutaciones de colores no producen soluciones factibles. Las nuevas restricciones son:

$$x_{ij} = 0 \ \ \forall\, j \geq i+1$$

$$x_{ij} \leq \sum_{k=j-1}^{i-1} x_{kj-1} \ \ \forall\, i \in V - \{1\}, \forall\, 2 \leq j \leq i-1$$

La primer restricción asegura que un vértice no puede recibir un color de índice mayor a su propio índice. En nuestro caso dado el conjunto independiente j-esimo, todos los vértices reciben el color j y como el vértice de menor índice es por construcción v_j, todos los vértices reciben un color que es a lo sumo tan grande como el de su índice.
La segunda restricción asegura que para que el vértice v_i pueda recibir el color j es necesario que algún vértice de índice menor haya recibido el color $(j-1)$. Notar que por la primer restricción no es necesario sumar por debajo de $k = j-1$.

Observación: Los poliedros convexos de los modelos 2 y 3 se encuentran contenidos en el poliedro convexo del modelo 1.

¿Cuál elegir? Si bien la elección natural es tomar el modelo que posea la menor cantidad de soluciones equivalentes, en la práctica no siempre es simple decidir que modelo conviene usar pues depende también de otros factores como la cantidad de variables, la cantidad de restricciones y el desempeño del solver utilizado. En particular los poliedros asociados a los modelos 2 y 3 no son fáciles de caracterizar pues dependen de ciertas propiedades del grafo. Por ejemplo en el modelo-3 dependiendo del orden en le que se indexan los vértices puede cambiar la dimensión de la relajación de la capsula convexa y la cantidad de facetas que definen desigualdades.

Cómo elegir los planos de corte: Una manera de cuantificar la calidad de un plano de corte es tomar como referencia el incremento que se obtiene en el valor de la cota inferior al agregar un cierto corte. Como regla general tenemos que cuanto mayor es el aumento, mejor son las restricciones. Pero aun así es necesario tener más cuidado al determinar si un corte es mejor que otro, pues podría suceder que dado un corte denso, este requiera de un mayor uso de memoria. Otras veces sucede que el tiempo de cómputo para procesar las nuevas restricciones no justifica la mejora en la cota inferior.

Para determinar los *cortes* que convine incorporar, se realizaron tests con

grafos generados aleatoriamente, prefijando la probabilidad de que exista una arista entre dos vértices, y de esta manera obtener una buena estrategia que sea equilibrada en cuanto al uso de memoria, tiempo de ejecución y mejora en la cota inferior.

5.5. Un algoritmo 'greedy' para coloreo de vértices

Este algoritmo también conocido como RS (*random sequential*) pertenece a un familia de algoritmos dominados *secuenciales*.

El método consiste en:

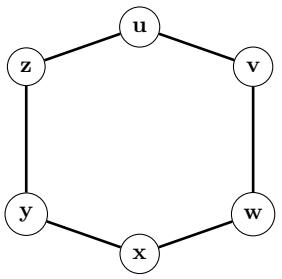
1. Numerar los vértices de G libremente, obteniendo así $v_1, ..., v_n$
2. Asignar el color 1 a v_1
3. Suponiendo que $v_1, ..., v_j$ se encuentran coloreados. Colorear a v_{j+1} con el menor color no utilizado por sus vecinos en $\{v_1, ..., v_j\}$.

Proposición 5.5.1. Con el algoritmo RS obtenemos una cota superior para $\chi(G)$.

$$\chi(G) \leq \Delta(G) + 1$$

Demostración. Una vez que ordenamos los vértices en una lista, cada vértice tiene a lo sumo $\Delta(G)$ predecesores vecinos. Luego el algoritmo no usará más de $\Delta(G) + 1$ colores. Entonces $\chi(G) \leq \Delta(G) + 1$. □

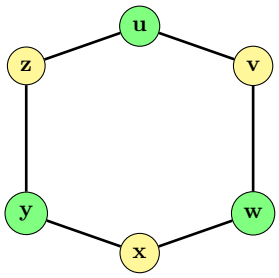
Ejemplo: A continuación se muestra una aplicación del algoritmo con el grafo $G = C_5$:

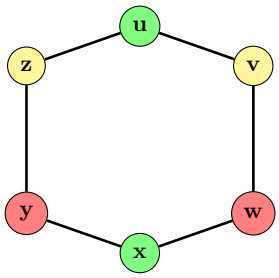


Si listamos los vértices de G en el orden u, w, v, y, z, x el algoritmo produce el coloreo: $c(u) = 1$, $c(w) = 1$, $c(v) = 2$, $c(y) = 1$, $c(z) = 2$, $c(x) = 2$. Por lo tanto se utilizan solamente dos colores. En este caso el algoritmo coincide con $\chi(G)$.

Si ahora en cambio listamos a los vértices de manera: u, x, v, w, z, y, el algoritmo produce el coloreo: $c(u) = 1, c(x) = 1, c(v) = 2, c(w) = 3, c(z) = 2, c(y) = 3$. Obteniéndose así un 3-coloreo del grafo.

En las dos imágenes siguientes se exhiben ambos coloreos. Los valores $1, 2, 3$ están representados por los colores: verde, amarillo, rojo respectivamente.





Como conclusión se observa que dependiendo de la manera en la que se indexan los vértices del grafo, se obtienen coloreos que utilizan distinta cantidad de colores.

Proposición 5.5.2. Para todo grafo G existe un orden de sus vértices para el cual el algoritmo RS produce un coloreo óptimo [24].

Complejidad

El algoritmo puede ser implementado en tiempo: $O(m+n)$. Pero encontrar un orden de los vértices para que el algoritmo RS produzca un coloreo óptimo en un grafo arbitrario es un problema **NP-Hard**, pues puede ser utilizado para resolver el problema de coloreo que es NP-Completo. Este hecho motivó el desarrollo de una familia de algoritmos basados en RS donde esencialmente lo que cambia es la manera de ordenar los vértices. El la sección 5.7 se presentan algunas de las variantes más conocidas.

En 1984 Chvátal probó que los cografos tienen una propiedad muy particular.

Teorema 5.5.3. (Chvátal) El algoritmo greedy aplicado a cografos produce un coloreo óptimo independientemente del modo en el que se indexan los vértices. [46]

Demostración. Más adelante, en la sección 5.9 □

Proposición 5.5.4. Sea G un grafo de intervalos (ver def. 2.1.18), luego $\chi(G) = \omega(G)$.

Demostración. Ordenamos los vértices según el extremo izquierdo de cada intervalo. Aplicamos el algoritmo RS y supongamos que el vértice x recibió el color de mayor número, digamos k. (Recordar que entonces $k \geq \chi(G)$

por como funciona el algoritmo). Como x no recibe un color menor, luego el extremo izquierdo de su intervalo (denotemos lo a) pertenece a intervalos ya coloreados con los colores $1, ..., k-1$. Todos estos intervalos contienen al punto a (por la manera en la que ordenamos los vértices). Luego tenemos una clique de cardinal k. Entonces $\omega(G) \geq k \geq \chi(G)$. Como $\chi(G) \geq \omega(G)$ para todo grafo, obtenemos la igualdad. $\square$

5.6. Teorema de Brooks

Teorema 5.6.1. Sea G un grafo conexo que no es ni completo ni un ciclo impar luego: $\chi(G) \leq \Delta(G)$.

Demostración. Sea $k = \Delta(G)$, podemos suponer que $k \geq 3$, pues si $k = 1$ G es completo y si $k = 2$ G es un grafo bipartito o un ciclo impar. En todos los casos el teorema vale.

El objetivo es el de indexar los vértices de G de modo tal que cada vértice tenga a lo sumo $k-1$ vecinos de menor índice. Luego por la prop. 5.5.1 aplicando el algoritmo 'greedy' RS se obtiene un coloreo que cumple con la cota.

Comenzamos por analizar el caso en el que G no es k-regular (ver def. 2.1.10). Tomamos un vértice con grado menor que k como v_n. Como G es conexo podemos construir un árbol generador de G con raíz en v_n. Asignamos los índices de manera decreciente a medida que avanzamos desde v_n. Obtenemos así un lista $v_1, ..., v_n$ donde cada vértice distinto de v_n tiene un vecino de índice mayor y por lo tanto a lo sumo $k-1$ vecinos de menor índice. Recordar que el grado de v_n es menor que k, luego todos los vértices tiene a lo sumo $k-1$ vecinos de menor índice y por lo tanto el algoritmo 'greedy' usa a lo sumo k colores.

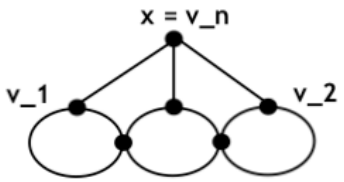
Supongamos de ahora en más que G es k-regular. Supongamos primero que G tiene un vértice de corte x (ver def. 2.3.4). Sea H una de las componentes de $G - x$, tomamos G' el grafo inducido por $V_H \cup x$. Notar que como el grado de x en G' es menor a k, pues por ser vértice de corte tiene vecinos en alguna otra componente, luego G' no es k-regular y puede ser coloreado usando el método anterior. De igual manera procedemos para las otras componentes que se obtuvieron al quitar el vértice de corte x, es decir a cada una se le agrega nuevamente el vértice x y todas las aristas que van en G desde esa componente a x. Luego con el método anterior coloreamos los subgrafos. Ahora permutando los colores asignados a los vértices de cada uno de estos subgrafos podemos hacer que el vértice x tenga el mismo color en cada uno de ellos. De este modo obtenemos un k-coloreo de todo G. Pues notar que por ser x vértice de corte, todo camino de una de las componentes a la otra tiene que pasar por x.

Supongamos entonces que G es 2-conexo (ver def. 2.3.3). Tomemos un vértice v_n tal que existen vértices v_1, v_2 ambos vecinos de v_n pero que no son adyacentes entre sí. Esto siempre es posible pues de lo contrario el grafo G seria completo (y además recordar que es 2-conexo). Supongamos que $G - \{v_1, v_2\}$ es conexo. En este caso indexamos los vértices de un árbol generador de $G - \{v_1, v_2\}$ usando los números $3, ..., n$ de modo tal que los índice disminuyan a medida que los vértices se alejan a la raíz v_n. Como sucedía antes, cada vértice distinto de v_n tiene un vecino de índice mayor y por lo tanto a lo sumo $k-1$ vecinos de menor índice. Por lo tanto el algorit-

mo'greedy' usará a los sumo k-colores para estos vértices. En particular al aplicar el algoritmo 'greedy' con este orden de vértices, notar que los vértices v_1 y v_2 reciben el mismo color (el color 1) pues no son adyacentes. Entonces para colorear a los vecinos de v_n se utilizan a lo sumo $k-1$ colores. Por lo tanto el vértice v_n puede ser coloreado con un color $\leq k$, como queríamos.

Claramente si G es n-conexo con $n \geq 3$ podemos proceder del mismo modo pues la hipótesis de que $G - \{v_1, v_2\}$ es conexo se cumple trivialmente. Luego para terminar la demostración es suficiente probar que todo grafo 2-conexo, k-regular, con $k \geq 3$ tiene vértices v_1, v_2, v_n que cumplen las hipótesis supuestas anteriormente.

Tomemos un vértice x. Si $\kappa(G-x) \geq 2$ (ver def. 2.3.2) definimos $v_1 := x$ y v_2 un vértice con distancia 2 de x. Esto es posible pues G es regular y no es un grafo completo. Tomamos v_n como un vecino común de v_1 y v_2. Si $\kappa(G-x) = 1$, definimos $v_n := x$. Como G no tiene vértices de corte, x tiene un vecino en cada bloque-hoja de $G-x$ (ver def. 2.3.5). Dos vecinos de x, v_1 y v_2 pertenecientes a bloque-hoja distintos, son no adyacentes. Además $G - \{x, v_1, v_2\}$ es conexo pues los bloques no tienen vértices de corte. Como $k \geq 3$, el vértice x tiene otro vecino y entonces $G - \{v_1, v_2\}$ es conexo.



□

5.7. Variantes del Algoritmo greedy

5.7.1. Método LF

Una de las variantes más sencillas es LF (*Largest First*) que consiste en ordenar los vértices de manera decreciente según el grado i.e. $d(v_i) \geq d(v_{i+1})\ \forall i = 1, ..., n$. La idea intuitiva es que los vértices que tienen menos vecinos poseen una mayor flexibilidad para ser coloreados y por lo tanto conviene colorearlos al final. Evidencia estadística indica que en general esta es una buena estrategia, mientras que si se ordenan los vértices de manera que los primeros sean los que tienen menos vecinos, el algoritmo requeriría una mayor cantidad de colores.

Complejidad

El algoritmo LF puede ser implementado en tiempo: $O(m + n)$.
Teniendo en cuenta que los grafos densos tienen una cantidad de aristas aproximadamente cuadrática con respecto a la cantidad de vértices, nos queda que $m \propto n^2$. Luego en el peor caso el orden de complejidad de LF es $O(n^2)$ [42].

Proposición 5.7.1. Si se ordenan los vértices de un grafo G de manera decreciente en le grado $d(v_i) \geq d(v_{i+1})\ \forall i = 1, ..., n$, luego:

$$\chi(G) \leq 1 + max_i\ min\{d_i, i - 1\}$$

Demostración. Aplicamos el método LF. Cuando se tiene que colorear el vértice i-esimo v_i como tiene a lo sumo $min\{d_i, i - 1\}$ predecesores vecinos, el mayor color que se le puede asignar es $1 + min\{d_i, i - 1\}$. Como el razonamiento vale para todos los vértices y estamos buscando una cota superior, se obtiene el resultado. □

5.7.2. Método SL

El método SL *Smallest Last* se basa en el mismo principio que LF i.e. los vértices de menor grado deben ser coloreados al final, pero cambia el modo de generar la lista de vértices. El procedimiento es el siguiente:

1. Sea $H = \emptyset$
2. Agregar a H el vértice de menor grado en $(V \backslash H)$
3. Si $(V \backslash H) \neq \emptyset$ volver a paso 2
4. Devolver un lista de vértices en orden inverso de como fueron agregados en H.

Notar que a diferencia de LF, con SL los grados de los vértices no se calculan de manera absoluta en todo el grafo original, pero relativa al subgrafo obtenido en cada paso.

Complejidad

El algoritmo SL puede ser implementado en tiempo: $O(m + n)$ [42].

5.7.3. Método DSATUR

A diferencia de los métodos mencionados antes la característica distintiva de este algoritmo es que la lista de vértices se forma de manera dinámica durante el proceso de coloreo, en lugar de comenzar con una lista predefinida. Su nombre deriva de *degree of saturation*.

Comenzamos por introducir el concepto de grado de saturación de un vértice:

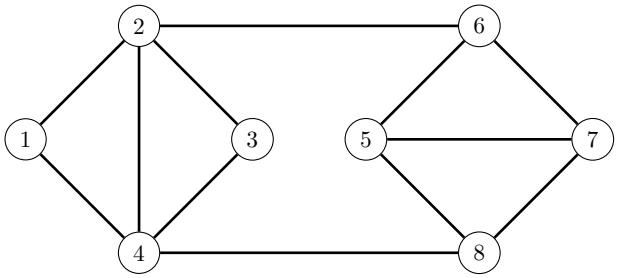
DSAT(v) := cantidad de colores diferentes usados por los vecinos de v.

El algoritmo consiste en:

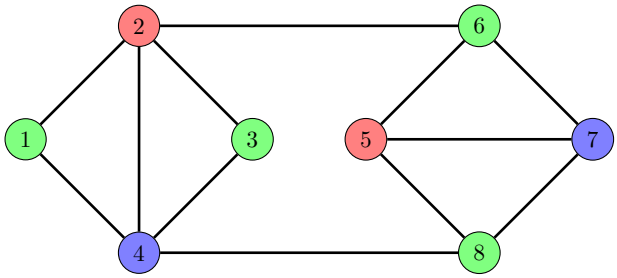
1. Ordenar los vértices de manera decreciente según el grado.
2. Colorear uno de los vértices de grado máximo con el color 1.
3. Elegir un vértice con DSAT máximo; en caso de igualdad, grado máximo.
4. Colorear el vértice obtenido en el paso anterior con el menor color posible.
5. Si todos los vértices están coloreados terminar, si no volver al paso 3.

Un ejemplo donde el algoritmo falla

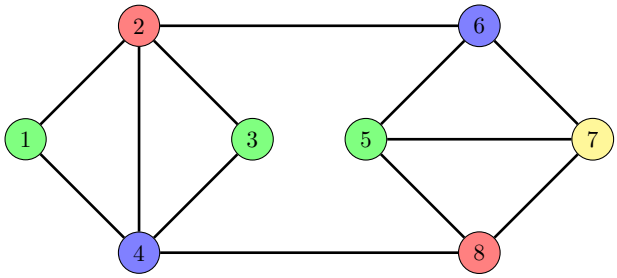
El siguiente grafo G tiene número cromático $\chi(G) = 3$. Pero DSATUR realiza un 4-coloreo. Este grafo es le más pequeño para el cual el algoritmo DSATUR falla. [36]



Un ejemplo de 3-coloreo.



Un ejemplo del resultado al aplicar DSATUR.



Para entender por qué el algoritmo requiere un color más, veamos que sucede paso a paso. Comenzamos por identificar los colores de la manera: rojo: 1, azul: 2, verde: 3, amarillo: 4.

- Calculamos el grado de cada vértice: $d(v_2) = d(v_4) = 4$, $d(v_5) = d(v_6) = d(v_7) = d(v_8) = 3$, $d(v_1) = d(v_3) = 2$

- Ordenamos a los vértices según su grado: $v_2, v_4, v_5, v_6, v_7, v_8, v_1, v_3$
- Le asignamos a v_2 el color 1 (rojo).
- Ahora el vértice con número DSAT mayor y a su vez entre ellos con mayor grado es v_4. Le asignamos a v_4 el color 2 (azul).
- Los dos vértices con mayor número DSAT son v_1, v_3 que tienen ambos DSAT = 2. Seleccionamos v_1, y le asignamos el menor color no usado, que es 3 (verde).
- Seleccionamos v_3 y también le asignamos el color 3 (verde).

Por ahora ambos coloreos coinciden. Veamos como prosigue:

- v_6, v_8 tienen mismo valor de DSAT y número de vecinos. Elegimos v_6 y le asignamos el color 2 (azul).
- ahora podemos elegir entre: v_5, v_7, v_8

Analizamos las distintas opciones:

Opción 1:

- Seleccionamos v_8 y le asignamos el color 1 (rojo)
- Seleccionamos v_5 y le asignamos el color 3 (verde)
- Seleccionamos v_7 y le asignamos el color 4 (amarillo)

Opción 2:

- Seleccionamos v_5 y le asignamos el color 1 (rojo)
- Seleccionamos v_7 y le asignamos el color 3 (verde)
- Seleccionamos v_8 y le asignamos el color 4 (amarillo)

Opción 3:

- Seleccionamos v_5 y le asignamos el color 1 (rojo)
- Seleccionamos v_8 y le asignamos el color 3 (verde)
- Seleccionamos v_7 y le asignamos el color 4 (amarillo)

Nota: Los caso que fueron obviados producen resultados similares debido a la simetría en el grafo.

Ahora podemos ver cual es el problema. El algoritmo en todos los casos le asignó colores distintos a los vértices v_6, v_8 porque debe asignar el menor color no usado. En particular notar que DSATUR nunca le asignaría el color 3 (verde) a los vértices v_6, v_8 como en cambio se hizo en el 3-coloreo anterior. De esta manera nos vemos forzados a usar un cuarto color para v_7.

Complejidad

El método DSATUR puede ser implementado en tiempo: $O(m \log n)$ [41].

Proposición 5.7.2. El algoritmo DSATUR colorea de manera óptima todos los grafos bipartitos conexos.

Demostración. Sea V_1, V_2 la partición de los conjuntos de vértices del grafo bipartito. Tenemos que mostrar que si se le asignó el color 1 a un vértice de V_1 luego el algoritmo nunca le asignará el color 1 a un vértice de V_2, pues de lo contrario necesitaríamos de por lo menos tres colores. Esto es así pues supongamos que primero se coloreó el vértice $v_1 \in V_1$ luego el algoritmo selecciona un vértice con número DSAT máximo que por definición es un vecino de v_1 y por lo tanto no puede recibir el mismo color. Análogamente en los pasos siguientes, cada vértice seleccionado por el algoritmo es vecino de otro vértice ya coloreado y como por ser bipartito sus vecinos están todos en el otro conjunto obtenemos un 2-coloreo del grafo. □

5.7.4. Otros métodos para seleccionar los vértices

Volvamos al algoritmo de DSATUR presentado antes y fijemos nuestra atención en el **paso-3** *"Elegir un vértice con DSAT máximo; en caso de igualdad, grado máximo"*. Existen otros criterios para hacer la selección del vértice en este paso para intentar mejorar la eficiencia del algoritmo.
Dos variantes interesantes pueden encontrares en:

- *A new DSATUR-based algorithm for exact vertex coloring* de San Segundo [40]
- *An improved algorithm for exact graph coloring* de Sewell. [39]

5.8. Comparación entre los algoritmos de tipo RS

Se implementaron los algoritmos: RS (viz. greedy), LF y DSATUR en Python y se usaron para colorear grafos conexos generados aleatoriamente con densidad de aristas predefinida. A continuación se reporta el promedio de los resultados obtenidos:

500 vértices y densidad de arista 0.5 (1000 iteraciones)

Algoritmo	Colores	Tiempo(s)
RS	72.53	0.02
LF	69.50	0.02
DSATUR	65.07	0.65

500 vértices y densidad de arista 0.8 (1000 iteraciones)

Algoritmo	Colores	Tiempo(s)
RS	135.87	0.03
LF	131.4	0.03
DSATUR	125.11	0.68

1000 vértices y densidad de arista 0.75 (100 iteraciones)

Algoritmo	Colores	Tiempo(s)
RS	216.87	0.15
LF	211.05	0.15
DSATUR	201.32	2.69

2000 vértices y densidad de arista 0.75 (100 iteraciones)

Algoritmo	Colores	Tiempo(s)
RS	389.9	0.82
LF	383.45	0.81
DSATUR	367.05	12.11

Conclusión: Definitivamente no conviene usar el algoritmo RS pues tiene el mismo tiempo de ejecución que LF pero requiere una mayor cantidad de colores. Por otra parte el algoritmo DSATUR es el que produce un mejor coloreo pero el tiempo de cómputo es considerablemente mayor que LF. Por lo tanto si el grafo que se quiere colorear no es demasiado grande conviene usar DSATUR y solo en caso de tener un grafo "grande" conviene sacrificar la calidad del coloreo para ahorrar tiempo de cómputo.

Un análisis para árboles: Se repitió el análisis anterior tomando G un árbol generado aleatoriamente. A continuación se reportan los resultados.

Árboles aleatorios de 500 vértices (1000 iteraciones)

Algoritmo	Colores	Tiempo(s)
RS	3.9	< 0.001
LF	3.01	< 0.001
DSATUR	2	0.6

Árboles aleatorios de 5000 vértices (100 iteraciones)

Algoritmo	Colores	Tiempo(s)
RS	4.02	0.02
LF	3.08	0.03
DSATUR	2	62.49

Conclusión: Vemos que DSATUR se desempeña de manera exacta, justamente como lo asegura la proposición (5.7.2). RS nuevamente es el peor requiriendo en promedio el doble de los colores necesarios. El desempeño de LF de un punto de vista de coloreo se encuentra justo en la mitad ente RS y DSATUR pero de un punto de vista temporal tiene un desempeño excelente pues es casi tan rápido como RS y **mucho** más rápido que DSATUR.

5.9. Algoritmo greedy para μ-coloreo de cografos

En esta sección veremos como se puede aplicar el algoritmo greedy para realizar un μ-coloreo de cografos.

Teorema 5.9.1. Sean G un cografo μ-coloreable, con $\mu : V \to \mathbb{N}$. Ordenamos los vértices de G de manera no decreciente según el valor de $\mu(v)$, i.e. $\mu(v_i) \leq \mu(v_{i+1})$. Ahora al aplicar el algoritmo greedy se obtiene un μ-coloreo de G. Más aún este coloreo es minimal (ver def. 4.2.4) y utiliza solamente los primeros $\chi(G)$ colores. [18]

Demostración. La propiedad vale para un grafo con un solo vértice. Supongamos que vale para cualquier cografo con menos de n vértices. Sea G un cografo con n vértices, sea μ tal que G es μ-coloreable, sea $v_1, ..., v_n$ una secuencia de vértices no decrecientes en μ. Por hipótesis inductiva aplicando el algoritmo greedy podemos obtener un μ-coloreo minimal de $G - v_n$. Lo notamos c_m. Supongamos que no es posible extender a c_m de modo tal que $c_m(v_n) \leq \mu(v_n)$. Luego por el lema 4.2.2 existe un grafo completo K de tamaño $\mu(v_n)$ en $N(v_n)$. Claramente $K \cup v_n$ es completo (y de tamaño $\mu(v_n) + 1$) pues v_n es adyacente a todos los vértices de K porque son sus vecinos, además por la manera en la que están indexados los vértices a ningún vértice de $K \cup v_n$ se le puede asignar un color mayor a $\mu(v_n)$. Entonces este grafo no es μ-coloreable y por lo tanto obtenemos una contradicción.
Notar que por la manera en la que el algoritmo greedy elige los colores, el coloreo obtenido es minimal.
Solo nos falta ver que no utiliza colores mayores a $\chi(G)$. Pero esto es consecuencia de lo que vimos antes. Si el algoritmo usa el color k entonces el grafo contiene un subgrafo completo de tamaño k y por lo tanto no es $k-1$ coloreable. □

Complejidad: Este algoritmo tiene orden de complejidad $O(n \log(n)+m)$.

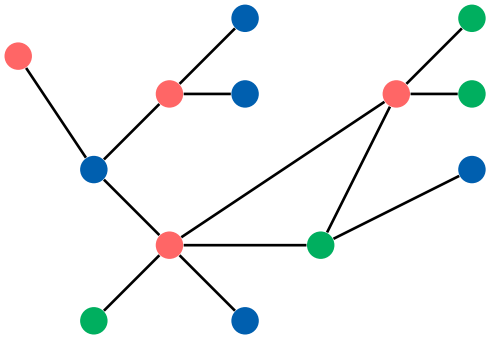
Demostración. Si usamos el algoritmo *Mergesort* podemos ordenar la lista de vértices según μ en tiempo $O(n \log n)$. Luego tenemos que colorear a cada vértice v con el menor color no utilizado. Esta operación tiene orden $O(d(v))$ si se tiene representado al grafo por lista de vecinos (2.5). Aplicando el teorema del apretón de manos (2.1.1) tenemos que: $\sum_{v \in V} d(V) = 2m_G$. Entonces el orden de complejidad resulta: $O(n \log(n) + m)$. □

Teorema de Chvátal: Usando el teorema anterior podemos probar el teorema de Chvátal mencionado anteriormente:
El algoritmo greedy aplicado a cografos produce un coloreo óptimo independientemente del modo en el que se indexan los vértices.
Para esto basta con tomar la función $\mu(v) \equiv \chi(G)$, entonces no solo estamos

realizado un k-coloreo en lugar de un μ-coloreo pero además como la función μ es constante cualquier orden de sus vértices es no decreciente en μ.

5.10. Coloreo Equitativo exacto con DSATUR

Esta sección está basada en el paper *A DSATUR-based algorithm for the Equitable Coloring Problem* de Isabel Méndez-Díaz, Graciela Nasini y Daniel Severín [37].



La idea es adaptar el algoritmo DSATUR para obtener un **coloreo equitativo** óptimo. Comenzamos por introducir la definición.

Definición 5.10.1. Un k coloreo equitativo (**k-eqcol**) es un coloreo de un grafo G donde el cardinal de las clases de colores difiere a lo sumo en una unidad entre sí. Es decir, sean $C_1, ..., C_n$ las clases de colores. Luego se cumple que:

$$||C_i| - |C_j|| \leq 1 \ \ \forall i, j \in \{1, ..., n\}$$

El **número cromático equitativo** denotado por $\chi_{eq}(G)$ es el menor k para el cual G posee un k-eqcol.

Luego el objetivo es modificar el algoritmo DSATUR para obtener $\chi_{eq}(G)$.

Notar que todo grafo admite un coloreo equitativo, de hecho si se le asigna un color diferente a cada vértice se obtiene trivialmente un coloreo equitativo.

Motivación: Un ejemplo donde puede ser útil el coloreo equitativo es en el caso donde se quiere distribuir una serie de tareas entre un conjunto de trabajadores. Si bien este problema puede ser resulto con las alternativas de coloreo mencionadas anteriormente, se podría producir la situación correcta de un punto de vista teórico pero un tanto injusta donde a una misma persona se le asignan muchos trabajos y a otras pocos. Con coloreo equitativo se logra una distribución más uniforme.

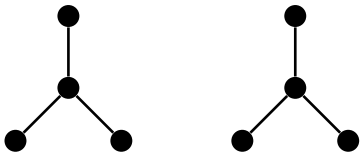
Proposición 5.10.1. La diferencia entre $\chi(G)$ y $\chi_{eq}(G)$ puede ser arbitrariamente grande.

Demostración. Tomamos como ejemplo los grafos estrella, es decir $K_{1,n}$. Claramente $\chi(G) = 2$, se colorea el centro de un color y todos los demás vértices de otro. Pero para un coloreo equitativo no se puede colorear más de dos vértices con el mismo color, pues como el vértice del centro es adyacente a todos, el color que se le asigna no puede volver a ser usado y por lo tanto tenemos una clase de cardinal 1. Si se usar el mismo color para 3 vértices, esta clase y la del vértice central diferirían en 2 unidades. Absurdo.

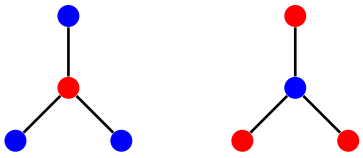
□

Observación: Es interesante notar que el problema del coloreo equitativo no permite utilizar libremente todas las técnicas que existen para coloreo clásico. Por ejemplo es fácil ver que para coloreo clásico el número cromático de un grafo disconexo es el mayor número cromático de cada componente. Este no es necesariamente el caso para coloreo equitativo como lo muestra el siguiente ejemplo.

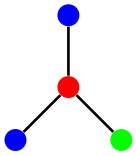
Ejemplo: Grafo disconexo.



Para el siguiente grafo disconexo se tiene que el número cromático equitativo es $\chi_{eq}(G) = 2$.

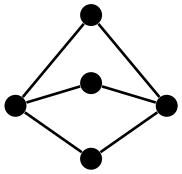


Pero el número cromático equitativo de cada componente conexa por separado es 3.

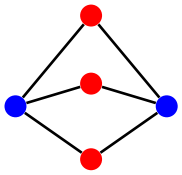


Otra diferencia se presenta cuando tratamos con vértices gemelos. Decimos que dos vértices v, w son **gemelos** si no son adyacentes y tienen exactamente los mismos vecinos. Para coloreo clásico una manera de aumentar la velocidad de cómputo de un algoritmo exacto es reduciendo el número de vértices del grafo. Para esto lo que se hace es notar que a dos vértices gemelos se les puede asignar el mismo color. Luego podemos reemplazar el grafo G por el grafo $G' = G - w$, con w gemelo de v. Esto procedimiento tampoco puede ser usado para coloreo equitativo como lo muestra el siguiente ejemplo.

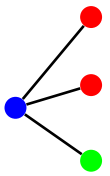
Ejemplo Vértices gemelos. No se puede hacer un coloreo equitativo del grafo G sin uno de sus vértices gemelos y luego agregarlo. En el grafo siguiente los vértices del extremo derecho e izquierdo son gemelos.



Coloreo equitativo. $\chi_{eq} = 2$.



Coloreo equitativo quitando uno de los vértices gemelos. $\chi_{eq} = 3$.



De los ejemplos anteriores podemos concluir que el coloreo equitativo es de hecho un problema diferente del coloreo clásico y requiere el desarrollo de nuevas técnicas para ser resuelto eficientemente.

Definición 5.10.2. Una **k-partición parcial** es una familia de conjuntos disjuntos $\Pi = (C_1, ..., C_n)$ tal que $\cup_{i=1}^{k} C_i \subseteq V$ y $C_i = \emptyset$ si $i \geq k+1$. Además notamos con $k(\Pi)$ a la cantidad de conjuntos no vacíos en Π. Llamamos $U(\Pi)$ al conjunto de los vértices que no están cubiertos por la partición parcial, i.e. $V \setminus \cup_{i=1}^{k} C_i$. Si $U(\Pi) = \emptyset$ decimos que Π es un k-partición. Por ultimo, dado $v \in V \setminus U(\Pi)$ notamos con $\Pi(v)$ al índice del conjunto al cual v pertenece, i.e. $v \in C_{\Pi(v)}$.

Definición 5.10.3. Un **k-coloreo parcial** es una k-partición parcial donde cada uno de los conjuntos C_i es un conjunto de vértices independientes. Los conjuntos C_i se denominan las clases de los colores. Notamos con $D_\Pi(v)$ al conjunto de los colores asignados a los vecinos de v. Con esta notación el grado de saturación de un vértice que llamamos anteriormente DSAT es el cardinal del conjunto $D_\Pi(v)$. Por ultimo definimos el conjunto de los colores disponibles para v como $F_\Pi(v) = \{1, ..., n\} \setminus D_\Pi(v)$.

Definición 5.10.4. Dada una k-partición parcial Π y $u \in U(\Pi)$ definimos la operación suma $\Pi + \langle u, i \rangle$ con $i \in \{1, ..., k+1\}$ como la nueva partición parcial que se obtiene al agregar u al conjunto C_i.
Decimos que una k_1-partición parcial Π_1 puede ser extendida a otra k_2-partición parcial Π_2, si esta última puede ser obtenida a partir de la primera aplicando reiteradamente el operador suma. Notar que necesariamente $k_1 \leq k_2$ y $C_i^1 \subseteq C_i^2$.

Objetivo: La idea para reducir el tiempo de cómputo es poder determinar si un coloreo parcial puede ser extendido a todo el grafo. De esta forma podemos evitar explorar ciertas ramas en lugar de perder tiempo intentando colorearlas. A continuación introducimos un teorema que nos provee de condiciones necesarias y suficientes para determinar si un coloreo parcial puede ser extendido a todo el grafo.

Teorema 5.10.2. MNS [37]. Dado $G(V, E)$ un grafo con $|V| = n$. Sea Π una k-partición parcial de V, sea $M = max\{|C_i| : i \leq k\}$ y sea t la cantidad de conjuntos en Π con cardinal M. Entonces Π puede ser extendida a una partición equitativa **si y solo si**:

$$n \geq (M-1)k + t$$

Demostración. ($\Rightarrow$) Si Π puede ser extendida a un partición equitativa Π' luego por lo observado anteriormente todos los conjuntos que tienen cardinal M en Π deben tener cardinal por lo menos M en Π'. Como consecuencia todos los otros conjuntos de la partición Π' deben tener por lo menos cardinal

$M-1$ pues de lo contrario diferirían en más de una unidad, y esto no es posible pues Π' es equitativo. Luego nos queda:

$$n \geq Mt + (M-1)(k-t)$$

que es equivalente a la desigualdad que queríamos ver.
($\Leftarrow$) Si vale la desigualdad, luego se tienen suficientes vértices para realizar el siguiente procedimiento: *agregar uno por uno los vértices restantes a la clase no vacía que posea cardinal menor.* Al finalizar se obtiene una partición equitativa. □

Corolario 5.10.3. Sea Π un k-coloreo de G, entonces Π es un coloreo equitativo **si y solo si** se cumple la desigualdad anterior.

Corolario 5.10.4. Sean Π, M, t como antes, LB una cota inferior para $\chi_{eq}(G)$ y Π_p un coloreo parcial de G. Entonces si Π_p puede ser extendido a un coloreo equitativo se tiene:

$$n \geq (M-1)max\{LB, k\} + t$$

Demostración. Si $k \geq LB$ por el teorema anterior vale la desigualdad.
Supongamos $k < LB$. Si Π_p puede ser extendido a un k_e-coloreo equitativo Π_e luego $k_e \geq \chi_{eq}(G) \geq LB$. Es necesario que las clases de colores en Π_p que tenían cardinal M, tengan por lo menos cardinal M en Π_e y todas las demás cardinal $M-1$. Por lo tanto obtenemos:

$$n \geq Mt + (M-1)(LB-t)$$

de donde se concluye el resultado que buscábamos. □

5.11. Algoritmo EqDSATUR

Entrada:

- Un grafo G
- Π^* un coloreo equitativo inicial
- LB una cota inferior de $\chi_{eq}(G)$

Salida:

- Π^* un coloreo equitativo óptimo
- $UB = \chi_{eq}(G)$.

Algoritmo:

1. UB $\leftarrow k(\Pi^*)$
2. Crear un coloreo parcial Π tal que:
   $C_i \leftarrow \{v_i\}$ $\forall v_i \in Q$ con Q una clique máxima.
3. $U(\Pi) \leftarrow V \backslash Q$
4. $k \leftarrow q$
5. ejecutar **EqDsatur(1,q)**

EqDsatur:

if $U(\Pi) = \emptyset$ **then**
  $UB \leftarrow k$
  $\Pi^* \leftarrow \Pi$
  return
end if

sellect $u \in U(\Pi)$

for all $i \in [1, min\{k+1, UB-1\}] \cap F_\Pi(u)$ **do**
  size $\leftarrow |C_i|$
  if $i \leq k$ **then**
    if size $= M$ **then**
      $t' \leftarrow 1$
      $M' \leftarrow M + 1$
    else if size $= M - 1$ **then**
      $t' \leftarrow t + 1$
      $M' \leftarrow M$
    else if size $\leq M - 2$ **then**
      $t' \leftarrow t$
      $M' \leftarrow M$
    end if
  else if $i = k + 1$ **then**
    if $M = 1$ **then**
      $t' \leftarrow t + 1$
      $M' \leftarrow M$
    else if $M \geq 2$ **then**
      $t' \leftarrow t$
      $M' \leftarrow M$
    end if
  end if
  $k' \leftarrow k$
  $k \leftarrow max\{i, k\}$
  if $n \geq (M' - 1)max\{k, LB\} + t'$ **then**
    $C_i \leftarrow C_i \cup \{u\}$
    $U_\Pi \leftarrow U_\Pi \backslash \{u\}$
    call: EqDsatur(M', t')
    $U_\Pi \leftarrow U_\Pi \cup \{u\}$
    $C_i \leftarrow C_i \backslash \{u\}$
  end if
  $k \leftarrow k'$
end for

Observación 1: Notar que le algoritmo requiere de una cota inferior de $\chi_{eq}(G)$. Es importante que esta cota sea lo mejor posible pues no es actualizada durante el resto de la ejecución. Méndez-Díaz, Nasini y Severín [37] proponen un método para encontrar la cota de manera eficiente.

Observación 2: En el segundo paso del algoritmo se utiliza una clique de tamaño máximo. Existen varias maneras de obtenerla, por ejemplo dado un vértice v, construimos una clique que contiene a v agregando entre todos sus vecinos uno de grado máximo, luego si existen otros vértices vecinos de ambos se agrega uno siguiendo el mismo criterio y así sucesivamente. El método puede ser aplicado a distintos vértices iniciales y luego se elige la clique máxima.

Conclusión

Se realizaron tests con grafos generados aleatoriamente con número de vértices entre 60 y 120, y densidad de aristas con entre 10 % y 90 %.
Este algoritmo produce excelentes resultados y se desempeña mejor que otros algoritmos implementados con el método de Branch-and-Cut o de Programación Entera (usando CPLEX). En particular con EqDSatur se puedo resolver más instancias que con programación lineal entera, y en las instancias resueltas por ambos EqDSatur fue más rápido.
Para una análisis detallado donde se reportan los distintos resultados obtenidos ver [37].

5.12. (γ, μ)-coloreo en grafos bipartitos completos

Complejidad: El siguiente algoritmo resuelve el problema en tiempo polinomial. [19]

Entrada:

- G un grafo bipartito completo con bipartición V_1, V_2
- Funciones $\mu : V \to \mathbb{N}$, $\gamma : V \to \mathbb{N}$ tales que $\gamma(v) \leq \mu(v)\ \forall v \in V$

Salida: un (γ, μ)-coloreo de G

Algoritmo:

Existen dos posibilidades:

- **caso-1:** $\exists v \in V$ tal que $\gamma(v) = \mu(v)$
- **caso-2:** $\forall v \in V\ \ \gamma(v) < \mu(v)$

caso-1: Sea v tal que $\gamma(v) = \mu(v)$, luego v debe ser coloreado con el color $\mu(v)$. Supongamos sin pérdida de generalidad que $v \in V_2$. Luego como G es bipartito completo ningún vértice de V_1 puede usar el color $\mu(v)$. Entonces podemos colorear todos los vértices $w \in V_2$ tales que $\gamma(w) \leq \mu(v) \leq \mu(w)$ con el color $\mu(v)$.
Ahora quitamos los vértices ya coloreados de V_2 y quitamos el color $\mu(v)$ de la lista de colores posibles. Para poder seguir iterando en instancias de (γ, μ)-coloreo, renumeramos los colores libres restantes de manera que sigan siendo valores consecutivos.
Si terminado este proceso algún vértice de V_1 quedó sin colores posibles para serle asignados, luego podemos concluir que el grafo G no era (γ, μ)-coloreable. De lo contrario podemos repetir el proceso con el subgrafo obtenido, y así sucesivamente hasta que se produce una de la tres alternativas siguientes:

- (γ, μ)-coloreamos todo el grafo
- concluimos que no es posible colorearlo
- llegamos al **caso-2**.

caso-2: Notar que entonces todo vértice admite por lo menos un color de índice impar y uno de índice par. Luego podemos colorear todos los vértices de V_1 con colores de índice impar y todos los vértices de V_2 con colores de índice par.

5.13. Complejidad

El problema de coloreo puede ser expresado como un problema de decisión: Dado un grafo G y un entero $k > 1$, ¿existe un k-coloreo de G?

El problema de coloreo es NP-Completo [7], por lo tanto en lugar de buscar un algoritmo polinomial que pueda colorear un grafo de manera exacta, uno puede relajar las exigencias.
Dado una algoritmo A que colorea los vértices de cualquier grafo G, notamos por $A(G)$ la cantidad de colores usados por A para colorear a G. Nuestro objetivo puede ser en lugar de buscar un algoritmo tal que $A(G) = \chi(G)\ \forall G$, buscar un algoritmo A tal que $A(G)/\chi(G)$ sea cercano a 1. Si bien existen numerosos algoritmos que se desempeñan de manera casi óptima para ciertos grafos, siempre es posible encontrar grafos donde el algoritmo requiere una cantidad de colores ampliamente superior a la de su número cromático.

5.14. Análisis del Peor Caso

El siguiente resultado juega un rol muy importante para intentar entender qué limitaciones tienen los algoritmos que pueden ser construidos y cuánto es razonable pretender de ellos.

Teorema 5.14.1. Si no existe un algoritmo para colorear los vértices de un grafo en tiempo polinomial (i.e. si $P \neq NP$) luego para todo algoritmo polinomial A existe un grafo G tal que $A(G) \geq 2\chi(G)$.

Notar que podemos volver a enunciar el teorema de manera más conveniente. De hecho este último puede expresarse como:

Teorema 5.14.2. Si para alguna constante $r < 2$ y una constante d existe un algoritmo A polinomial tal que $A(G) \leq r\chi(G) + d$, luego existe un algoritmo polinomial A' tal que $A'(G) = \chi(G)$. [25]

Teorema 5.14.3. Si para alguna constante $r < 2$ y una constante d existe un algoritmo A polinomial tal que $A(G) \leq r\chi(G) + d$, luego existe un algoritmo polinomial para determinar si un grafo G cumple $\chi(G) \leq 3$. [25]

Teorema 5.14.4. (Stockmeyer) Si existe un algoritmo A que puede determinar en tiempo polinomial si $\chi(G) \leq 3$, entonces existe un algoritmo A para colorear los vértices de un grafo en tiempo polinomial tal que $A(G) = \chi(G)$. [28]

Capítulo 6

Conclusiones y problemas abiertos

En este trabajo se intentó realizar una recopilación de los resultados más importantes sobre coloreo de grafos, de modo tal que resulte útil no solo para un lector que comienza a dar los primeros pasos en esta área, pero también para que sirva como referencia, donde se pueda encontrar de modo ordenado desde los resultados más fundamentales hasta propiedades y cuestiones un tanto más especificas.

A lo largo del trabajo se puede apreciar cuan vasta y variada resulta la teoría de coloreo. Incluyendo desde aspectos de matemática pura a cuestiones de ciencias de la computación. Es curioso como una formulación tan simple como puede ser la de colorear los vértices de un grafo, nos provee de un inmenso poder de expresión para atacar problemas prácticos y construir modelos para obtener soluciones.

El estudio sobre coloreo y sus aplicaciones prácticas ha sido extremadamente fructífero en las últimas décadas, y lejos de estar llegando a una terminación, aún queden muchas cuestiones por resolver. Por ejemplo, recordamos los problemas abiertos mencionados anteriormente:

- Sería interesante tener una demostración del teorema de los 4 colores puramente matemática, es decir sin necesitad de complementar la demostración con el uso de una computadora.
- Conjetura de coloreo total. (Vizing):
  $\forall G \ \chi''(G) \leq 2 + \Delta(G)$ (3.14)
- Conjetura de coloreo por listas:
  $\forall G \ ch'(G) = \chi'(G)$ (4.11)
- Determinar la complejidad de (γ,μ)-coloreo de cografos. (4.8)

Otros problemas abiertos de interés son:

- Par el algoritmo de reconocimiento de grafos Berge (3.7) sería interesante tener una cota inferior para la complejidad, dado que el algoritmo actual tiene orden: $O(n^9)$.

- Conjetura de Hadwiger (1943) [51]
  Este es un de los problemas abiertos más importantes en teoría de coloreo. Sugiere una generalización del teorema de los 4 colores.
  Dada un grafo G definimos el **número de Hadwiger** $\eta(G)$ como el máximo número k tal que el grafo completo K_k es un **menor** de G, es decir existen k subconjuntos de $V(G)$, $X_1, ..., X_k$ disjuntos dos a dos tales que $G[X_i]$ es conexo para todo $1 \leq i \leq k$ y $E_G(X_i, X_j) \neq \emptyset$ para todo $1 \leq i < j \leq k$.
  La conjetura de Hadwiger asegura que: $\chi(G) \leq \eta(G) \ \ \forall\, G$ simple.

- Conjetura de Erdős–Faber–Lovász (1972):
  Dados k grafos completos, cada uno con k vértices y tales que cada par de grafos tiene a lo sumo un vértice en común. Entonces el grafo que resulta de la unión de todos los grafos puede ser coloreado con k colores. [52]

- Conjetura de Gyárfás–Sumner (1975):
  Para todo árbol T y todo completo K, los grafos que no tiene ni a T ni a K como subgrafo inducido pueden ser coloreados con una cantidad constante de colores. [53] [54]

- Problema de Hadwiger–Nelson:
  Encontrar la mínima cantidad de colores para colorear el plano de modo tal que no haya dos puntos a distancia 1 entre si que tengan el mismo color.
  Si bien no se conoce la respuesta, se sabe que la cantidad es 5, 6 o 7. Más aún el valor puede depender de los axiomas de teoría de conjuntos que se eligen.

Bibliografía

[1] D.B. West
Introduction to graph theory
Prentice Hall, Upper Saddle River, NJ, 2001

[2] Reinhard Diestel
Graph Theory
Springer, Graduate Texts in Mathematics

[3] V.G. Vizing
Coloring the vertices of a graph in prescribed colors
Metody Diskret. Anal. v Teorii Kodov i Schem 29 (1976)

[4] P. Erdős, A.L. Rubin, H. Taylor
Choosability in graphs
Proc. West Coast Conference on Combinatorics, Graph Theory and Computing, Arcata, California, XXVI (1979) 125–157

[5] Norman Biggs
Algebraic Graph Theory
Cambridge University Press 1974

[6] Scheinerman, Edward R.
Mathematics: A Discrete Introduction (3rd ed.)
Cengage Learning (2012)

[7] Grötschel, M., Lovász, L., Schrijver, A. (1981)
The ellipsoid method and its consequences in
combinatorial optimization, Combinatorica, 1, 169–197

[8] Gary Chartrand, Linda Lesniak, Ping Zhang
Graphs & Digraphs
Chapman and Hall CRC; 6 edition

[9] Robert J. Wilson
Introduction to Graph Theory, Fifth Edition

[10] J. Mycielski
Sur le coloriage des graphes
Colloq. Math. 3 (1955) 161- 162

[11] C. Berge
Some classes of perfect graphs. Six Papers on Graph Theory
Indian Statistical Institute, Calcutta (1963) 1-21

[12] Zsolt Tuza
Graph colorings with local constraints — a survey
Discussiones Mathematicae Graph Theory 17 (1997) 161–228

[13] M. Biro, M. Hujter, Zs. Tuza
Cross fertilisation of graph theory and aircraft maintenance scheduling
Thirty-Second Annual Symposium (G. Davison, ed.), AGIFORS
(Airline Group of the International Federation of
Operational Research Societies), 1993, 307–317

[14] M. Biro, I. Simon, C. Tdnczos
Aircraft and Maintenance Scheduling Support,
Mathematical Insights and a Proposed Interactive System
Journal of Advanced Transportation,Vol.26,No.2,pp. 121-130

[15] M. Biro, M. Hujter, Zs. Tuza
Precoloring extension. I. Interval graphs
Discrete Mathematics 100 (1992) 267-279

[16] M. Hujter and Zs. Tuza
Precoloring extension. II. Graphs Classes Related to Bipartite Graphs
Acta Mathematica Universitatis Comenianae, 62(1), 1–11

[17] M. Hujter and Zs. Tuza
Precoloring Extension. III. Classes of Perfect Graphs

[18] Bonomo F. and Cecowski M.
Between coloring and list-coloring: μ-coloring (2005)
Electronic Notes in Discrete Mathematics, 19, 117–123

[19] Bonomo F., Durán G., Marenco J.
Exploring the complexity boundary between coloring and list-coloring
Springer Science Business Media, LLC 2008

[20] Bertossi, A.
Dominating sets for split and bipartite graphs (1984)
Information Processing Letters, 19, 37– 40

[21] Jansen, K. and Scheffler, P.
Generalized coloring for tree-like graphs (1997)
Discrete Applied Mathematics, 75, 135–155

[22] Lekkerkerker, C.G.; Boland, J.C.
Representation of a finite graph by a set of intervals on the real line (1962). Fund. Math, 51: 45–64

[23] Marx, D.
Precoloring extension on unit interval graphs (2006)
Discrete Applied Mathematics, 154, 995– 1002

[24] Matula, D.W., Marble, G. and Isaacson, J.D.
Graph Coloring Algorithms,
Graph Theory ami Computing, Ronald C. Read, editor, (Academic Press, New York, 1972), pp. 109-122

[25] M.R. Garey and D.S. Johnson
The Complexity of Near-Optimal GraphColoring
Bell Laboratorzes, Murray Hdl, New Jersey
Journal of the Assomanon for Computing Machinery, Vol 23, No 1, January 1976,pp 43-49

[26] König, D.
Über graphen und ihre anwendung auf determinantentheorie
und mengenlehre, Mathematische Annalen (1916)

[27] Holyer, I.
The NP-completeness of edge-coloring (1981)
SIAM Journal on Computing, 10, 718–720

[28] Stockmeyer, L.
Planar 3-colorabdlty Is polynomial complete
ACM SIGACT News 5, 3 (1973), 19-25

[29] Walter Klotz
Graph Coloring Algorithms
Mathematik-Bericht 5 (2002), 1-9, TU Clausthal

[30] Frank Thomson Leighton
A Graph Coloring Algorithm for Large Scheduling Problems
Center for Applied Mathematics,
National Bureau of Standards Washington, DC (1979)

[31] B. Reed
ω, Δ and χ
Graph Theory 27 (1998) 177-212

[32] NetworkX
Software for complex networks
https://networkx.github.io

[33] Noga Alon
Restricted colorings of graphs
Tel Aviv University, Tel Aviv, Israel
Surveys in Combinatorics, Proc.
14th British Combinatorial Conference, London Mathematical Society
Lecture Notes Series 187, edited by K. Walker,
Cambridge University Press, 1993, 1-33

[34] Misra, J.; Gries, David
A constructive proof of Vizing's Theorem (1992)
Information Processing Letters

[35] Zykov, A.A.
On some properties of linear complexes
Amer. Math. Soc. Translations 79 (1952), p. 81

[36] R. Janczewski, M. Kubale, K. Manuszewski, K. Piwakowski
The smallest hard-to-color graph for algorithm DSATUR

[37] Isabel Méndez-Díaz, Graciela Nasini y Daniel Severínb
A DSATUR-based algorithm for the Equitable Coloring Problem

[38] Isabel Méndez-Díaz, Paula Zabala
A cutting plane algorithm for graph coloring
Discrete Applied Mathematics 156 (2008) 159 – 179

[39] Sewell
An improved algorithm for exact graph coloring
Cliques, coloring and satisfiability
American Mathematical Society; 1996, p. 359–73

[40] San Segundo
A new DSATUR-based algorithm for exact vertex coloring

[41] J.S. Turner
Almost all k-colorable graphs are easy to color
J. Algorithms 9 (1988) 63-82

[42] Adrian Kosowski, Krzysztof Manuszewski
Classical Coloring of Graphs

[43] Erdős, Paul; Wilson, Robin J. (1977)
Note on the chromatic index of almost all graphs
Journal of Combinatorial Theory

[44] Ewa Kubicka and Allen J. Schwenk
An Introduction to Chromatic Sums
Department of Mathematics and Statistics
Western Michigan University

[45] V.G. Vizing
Critical graphs with a given chromatic class (Russian)
Diskret. Analiz 5 (1965), 9-17

[46] V. Chvátal
Perfectly ordered graphs
Discrete Math 21 (1984)

[47] László Lovász
A characterization of perfect graphs
Journal of Combinatorial Theory, Series B 1: 95–98, 1972.

[48] M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas
The Strong Perfect Graph Theorem
Annals of Mathematics, 164 (1): 51–229, 2006.

[49] M. Chudnovsky, A. Scott, P. Seymour, S. Spirkl
Detecting an odd hole
https://arxiv.org/abs/1903.00208 (1 Mar 2019)

[50] G. Cornuejols, M. Chudnovsky, X. Liu, P. Seymour and K. Vuskovic
Recognizing Berge Graphs
Combinatorica 25, 143-186, 2005.

[51] Hadwiger, H. Über eine Klassifikation der Streckenkomplexe (1943)
Viertel/sehr. Naturforsch. Ges. Zürich, 88:133-142.

[52] Erdős, Paul
On the combinatorial problems which I would most like to see solved
(1981) Combinatorica, 1: 25–42,

[53] Gyárfás, A.
On Ramsey covering-numbers
Infinite and finite sets (1975)
Vol. II, Colloq. Math. Soc. János Bolyai, 10, Amsterdamd, pp. 801–816

[54] Sumner, D. P.
Subtrees of a graph and the chromatic number
The theory and applications of graphs
(Kalamazoo, Mich., 1980), Wiley, New York, pp. 557–576