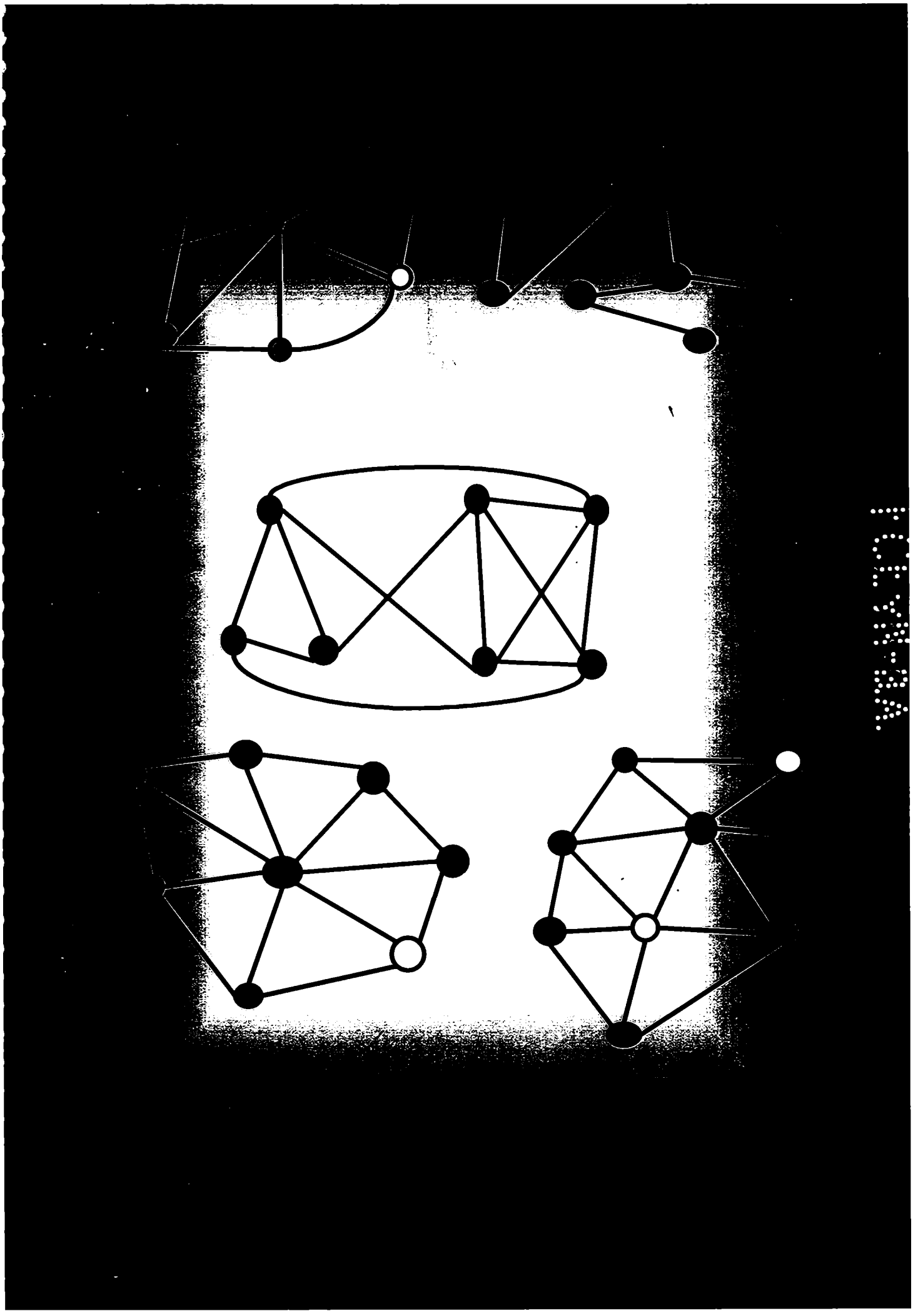
Tesis de Posgrado

Problema de coloreo de Grafos : un estudio poliedral y un algoritmo Branch-and-Cut

Méndez Díaz, Isabel

2003

Tesis presentada para obtener el grado de Doctor en Ciencias de la Computación de la Universidad de Buenos Aires



Tesis Doctoral

Problema de Coloreo de Grafos
Un Estudio Poliedral y un Algoritmo Branch-and-Cut

Isabel Méndez Díaz

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Director: Dr. Nelson Maculan

Abril de 2003

A mamá y papá
A Fabi, Pili y Nico

Desvarío laborioso y empobrecedor el de componer vastos libros;
el de explayar en quinientas páginas una idea
cuya perfecta exposición oral cabe en pocos minutos.
Mejor procedimiento es simular que esos libros ya existen
y ofrecer un resumen, un comentario...

J.L.B.

Resumen

El *problema de coloreo de grafos*, *PCG*, es uno de los problemas clásicos de la teoría de grafos y es estudiado desde el siglo XIX. Más allá del interés teórico, tiene una significativa importancia práctica debida a las numerosas situaciones de la vida real en las cuales surgen problemas que pueden ser modelados como un *problema de coloreo de grafos*.

PCG pertenece a la clase de problemas *NP-Hard*, es decir que no se conoce un algoritmo polinomial para resolverlo. Existe en la bibliografía gran cantidad de trabajos proponiendo algoritmos para su resolución, especialmente heurísticas y en menor medida algoritmos exactos.

Como muchos problemas de Optimización Combinatoria, *PCG* se puede modelar como un problema de programación lineal entera. Los algoritmos *Branch-and-Cut* son la herramienta más efectiva que se conoce para resolver un modelo de programación lineal entera. En particular, las implementaciones que usan desigualdades válidas del poliedro asociado al modelo han mostrado ser las más efectivas. Tal vez una de las mayores dificultades de este abordaje se presenta cuando el problema de programación lineal entera tiene la propiedad de simetría, es decir que existen múltiples soluciones con el mismo valor de la función objetivo. En estos casos, los algoritmos habituales disminuyen en gran medida su performance.

El objetivo de esta tesis es abordar la resolución del *problema de coloreo de grafos* mediante modelos de programación entera binaria que parcialmente eliminen soluciones simétricas. Con este fin, proponemos varios modelos que tratan la simetría del problema con diferentes criterios. Estudiamos los poliedros asociados a estos modelos y desarrollamos e implementamos un algoritmo *Branch-and-Cut* donde utilizamos este estudio poliedral y estrategias particulares que guían la búsqueda evitando explorar sobre soluciones simétricas. Se presentan resultados que demuestran que este algoritmo es capaz de competir exitosamente con los algoritmos exactos conocidos.

Abstract

The graph coloring problem, *PCG*, is perhaps one of the oldest and most well-known problems in graph theory. Nowadays, it arises in many applications such as scheduling, timetabling, electronic bandwidth allocation and sequencing.

PCG is known to be NP-hard for arbitrary graphs. The practical importance of the problem makes neccesary to devise algorithms capable of solving, in acceptable computational times, medium to moderate instances arising in real-world applications. A lot of work has been spent in an attempt to develop efficient algorithms for the problem, mainly by using heuristic techniques to deal with large instances. Relatively few methods for solving the problem exactly can be found in the literature.

Like most optimization problems on graphs, *PCG* can be formulated as a linear integer programming problem. LP-based Branch-and-Cut algorithms are currently the most successfull tool to deal with these models computationally. However, the amount of research effort spent in attempts to solve *PCG* by this method is not comparable with that devoted to other problems, like TSP or maximum stable set.

If the integer programming formulation exhibits symmetries, i.e. many solutions exist with the same optimal value, it turns out that Branch-and-Cut exhibts poor performance even for small instances. The classical models for *PCG* suffered from this problem.

In this thesis, we present new integer programming formulations that reduce the number of feasible symmetrical solutions. We develop a polyhedral study of the polytope associated with the proposed models in order to derive families of facet-defining inequalities.

Branch-and-Cut implementations that take advantage of the particular structure of the problem under consideration have proved to be the most successfull. In this sense, the use of cutting planes arising from a polyhedral study of the feasible solution set allowed many instances of hard combinatorial optimization problems to be solved to proven optimality for the first time. We develop a Branch-and-Cut algorithm based on our theoretical polyhedral results. We also take into account many others factors like preprocessing, search and branching strategies, lower and upper bounds and streghthening of the LP-relaxation. We present computational results showing that our algorithm performance is successful when compared to known exact algorithms.

Índice General

Capítulo 1

Introducción

1.1. Un Poco de Historia

Hay problemas de la física, química, genética e informática, que pueden ser modelados como problemas en *grafos*. Sin embargo, este concepto tiene su origen en una anécdota pueblerina. Debemos remontarnos al siglo XVIII para encontrar la noción de *grafo* como un modelo matemático que permite analizar y resolver muchos problemas de la vida real.

La ciudad de Königsberg, Prusia, está atravesada por el río Pregel. En su curso, se encuentra una isla a partir de la cual el río se ramifica en dos. Existían 7 puentes que permitían pasar de una parte a otra. Los habitantes de Königsberg se preguntaban si sería posible realizar un paseo atravesando cada uno de los puentes una única vez. Fallidos intentos hacían pensar que ésto no era posible pero no se tenía certeza que así fuera. Fue en 1736 cuando el famoso matemático Leonhard Euler pudo asegurar que no era posible tal recorrido. El modelo matemático usado asociaba a cada zona de tierra firme un punto (*vértice*) y a cada puente una línea (*arista*) que unía los correspondientes puntos. El problema se tradujo en encontrar un recorrido por los *vértices* que pase una y sólo una vez por cada *arista*. Cualquier configuración de *vértices* y *aristas* es lo que conocemos hoy como *grafo*. Euler fue más allá del problema de los puentes de Könisberg y dió condiciones sobre la estructura de un *grafo* para que exista tal tipo de recorrido.

Tal vez el problema que más contribuyó al crecimiento de la teoría de *grafos* fue *el problema de los 4 colores*. En el año 1852, Francis Guthrie observó al colorear un mapa de Inglaterra que le bastaban 4 colores aún en el caso que dos regiones vecinas no pudieran ser coloreadas con el mismo color. Intrigado, escribe una carta a su hermano Frederick, discípulo del matemático De Morgan, planteándole la inquietud de si cualquier mapa dibujado en una hoja de papel tendría esta propiedad. Fredrick y De Morgan intentaron dar una respuesta matemática al problema. Pero, si bien no pudieron dibujar un mapa que requiriera al menos 5 colores, tampoco pudieron demostrar que 4 colores bastaban. El 23 de Octubre de 1852, De Morgan escribe una carta a Hamilton explicando el problema que termina diciendo:

If you return with some very simple case which makes me out a stupid animal, I think I must do as the sphynx did...

De Morgan no logró despertar el interés de Hamilton quien respondió que no disponía de tiempo

para dedicarse al tema.

A fines del siglo XIX, la comunidad matemática comenzó a tratar el problema. En 1878 apareció la primera referencia escrita debida a Cayley en *The Royal Geographical Society*, quien también explicó ante la *London Mathematical Society* las dificultades encontradas hasta el momento para el tratamiento del problema.

En 1879, Alfred Kempe [52] publica en *American Journal of Mathematics* la primera demostración del problema que fue considerada correcta. Sin embargo, en 1890 Percy John Heawod [44] encuentra un error en los argumentos usados por Kempe y contribuye con el resultado que 5 colores son suficientes para colorear un mapa.

Durante muchos años diversos investigadores abordaron el tema. Hubo infructuosos intentos de demostración, pero muchos de ellos aportaron resultados parciales o formulaciones equivalentes que enriquecieron la teoría. Recién en 1976, Wolfgang Haken y Kenneth Appel [2, 3] logran demostrar la conjetura con la ayuda de una computadora. Su demostración está basada en conceptos que ya habían sido utilizados por Kempe, Heawod y Heesch en sus fallidas demostraciones: reducibilidad y descarte. Esencialmente, si la conjetura fuera falsa ninguna configuración entre 1478 posibles podría aparecer en el mapa contraejemplo (reducibilidad). Sin embargo, se demuestra que el mapa contiene alguna de las 1478 configuraciones (descarte) llegando a una contradicción. El procedimiento de reducibilidad y descarte de los 1478 casos fue efectuado con una computadora. Esta metodología no fue aceptada universalmente como una demostración ante la dificultad de verificar la validez del razonamiento de la computadora. En 1997, Robertson, Sanders, Seymour y Thomas [70] logran reducir a 633 las configuraciones a tener en cuenta y utilizan un procedimiento computacional más transparente que es aceptado como una demostración formal de la conjetura.

De la misma manera que el problema de los puentes de Könisberg, el *problema de los 4 colores* puede ser modelado como un problema de *grafos*. Basta asignar un *vértice* a cada país y una *arista* entre *vértices* que correspondan a países fronterizos.

Pero la historia no termina con la coloración de un mapa. A partir de la conjetura de los 4 colores que despertó durante muchos años el interés de los investigadores, se desarrolló el área de teoría de *grafos*. Entre los muchos problemas estudiados surgió una natural extensión del problema de los 4 colores: el coloreo de un *grafo*. Un *coloreo* de un *grafo* es cualquier asignación de colores a los *vértices* de tal manera que *vértices* conectados por una *arista* no comparten el mismo color. Es inmediato plantearse la pregunta: ¿cuál es la cantidad mínima de colores necesarias para colorear un *grafo* y cómo colorear el *grafo* con esa cantidad de colores? Este es el conocido *problema de coloreo de grafos*.

No se ha encontrado aún una respuesta teórica a esta pregunta para cualquier configuración de un *grafo*. Los resultados teóricos de la literatura podemos encuadrarlos en dos tipos. Por un lado aquellos que han caracterizado familias de *grafos* para las cuales se puede establecer la cantidad mínima de colores y, en algunos casos, cómo colorear con esa cantidad de colores. Por otro lado, resultados que brindan cotas superiores e inferiores para *grafos* generales o con algunas propiedades particulares. El libro de T. Jensen y B. Toft [48] es una excelente referencia para el estudio de los resultados teóricos sobre el *problema de coloreo de grafos*.

Más alla del interés teórico que puedan tener los problemas en *grafos*, muchos de ellos y en particular el *problema de coloreo de grafos* tienen una significativa importancia práctica. En la siguiente Sección describimos algunos problemas de la vida real que pueden ser modelados como un *problema de coloreo de grafos*.

1.2. Aplicaciones del *Problema de Coloreo de Grafos*

Son numerosas las situaciones de la vida real en las cuales surgen problemas que pueden ser modelados como un *problema de coloreo de grafos*. A continuación mencionaremos algunas.

- ***Asignación de Horarios***

  Supongamos que queremos asignar turnos para el dictado de un conjunto de materias. Claramente deberá considerarse que dos materias dictadas por el mismo profesor no podrán ser asignadas a turnos que se superpongan. El mismo argumento es aplicable a materias dictadas a un mismo grupo de estudiantes. El problema de determinar el mínimo número de turnos para dictar todas las materias es equivalente al *problema de coloreo* en el *grafo* que resulta de considerar un *vértice* por cada posible materia-turno y una *arista* por cada incompatibilidad horaria. Asociando los turnos a colores, es inmediata la equivalencia de los problemas.

  Este problema se encuadra dentro de una clase más general que es el de programación de actividades que deban ser ejecutadas por máquinas y/o personas. El objetivo es encontrar la cantidad mínima de máquinas y/o personas necesarias para realizar un conjunto de tareas. El caso más sencillo es cuando cualquier tarea puede ser realizada por cualquier máquina/persona y existen ciertas incompatibilidades entre las tareas que impiden que éstas sean realizadas en la misma máquina. Por ejemplo, si dos tareas deben ser realizadas en el mismo horario. Este problema ha sido tratado por muchos investigadores, entre los cuales podemos mencionar los trabajos de Leighton [57] y Werra [27].

- ***Asignación de Frecuencias Radiales***

  Consideremos un conjunto de usuarios (radios o particulares) del espectro electromagnético a los cuales hay que asignar frecuencias. Para evitar interferencias, las zonas geográficamente cercanas deberán tener distintas frecuencias mientras que aquellas que se encuentren suficientemente lejos podrán compartirlas. Determinar cómo asignar el mínimo número de frecuencias necesarias para suplir los requerimientos de todos es equivalente al *problema de coloreo de grafos*. Basta considerar un *vértice* por cada usuario y una *arista* entre dos *vértices* cuando la distancia geográfica entre los usuarios es inferior a una distancia establecida para evitar interferencias. Si cada frecuencia es identificada con un color, encontrar la mínima cantidad de colores para colorear el *grafo* resuelve el problema.

  En la práctica pueden surgir algunos otros requerimientos y/u objetivos. Por ejemplo, los usuarios pueden tener restricciones sobre el uso de las frecuencias. Además, ante una situación en la que sea imposible evitar interferencias, es de utilidad saber cúal es aquella asignación que minimiza la probabilidad global de que se produzca una interferencia. La aplicación de modelos matemáticos basados en coloreo para resolver el problema fueron introducidos por Metzger [65].

En [43], Hale presenta un estudio del problema, sus extensiones y la relación con el problema de coloreo. Además introduce generalizaciones del problema de coloreo que han dado origen a una basta teoría y resultan ser modelos aplicables a muchas de las variantes del problema de asignación de frecuencias. Más recientemente, la tesis de Koster [54] brinda un excelente estado del arte del tema.

- *Asignación de Registros*

  Durante la ejecución de un programa, las variables que se encuentran en los registros de memoria pueden ser accedidas más rápidamente que aquellas que no se encuentran en los mismos. En la mayoría de los casos, la cantidad de variables supera el número de registros disponibles, por lo cual un registro deberá ser usado en distintos tiempos de ejecución por diferentes variables. Con el objetivo de minimizar el tiempo de ejecución, es fundamental que aquellas variables a las cuales deba accederse dentro de un mismo período corto de tiempo estén accesibles en los registros de memoria. Es decir, es necesario minimizar los accesos a memoria fuera de los registros. Se puede crear un *grafo* con un *vértice* por cada variable del programa y *aristas* entre *vértices* que correspondan a variables necesarias en un mismo período de tiempo. Si identificamos cada registro con un color, cualquier coloreo del *grafo* brinda una posible relación variable-registro que no asigna un mismo registro a variables requeridas en igual período. Si el mínimo número de colores necesario para colorear el *grafo* es inferior a la cantidad de registros disponibles, se optimiza el uso de la memoria. Debido al interés práctico de esta aplicación, hay numerosos trabajos así como variaciones del problema. (Chaitin [12], Chow [14], Briggs et al. [11]).

1.3. ¿Es Difícil Colorear un Grafo?

Hasta aquí hemos hecho un poco de historia sobre el origen del problema de coloreo y de las actuales aplicaciones prácticas del mismo. Como consecuencia de esta relevancia práctica, tiene especial interés la pregunta: ¿Qué podemos hacer para encontrar un coloreo de un *grafo* que utilice la cantidad mínima de colores?.

Todo coloreo de un *grafo* puede pensarse como una asignación entre V, el conjunto de *vértices*, y el conjunto $\{1,\ldots,n\}$ donde $n = |V|$. Entonces, el conjunto de todos los coloreos de un *grafo* tiene cardinal finito y el *problema de coloreo de grafos* se reduce a buscar en un conjunto finito de elementos, uno que utilice la cantidad mínima de colores. Esta propiedad puede llevarnos a la conclusión errónea que el problema puede ser resuelto por simple enumeración de las soluciones. Sin embargo, si bien el número de coloreos es finito, puede ser tan grande que enumerar todos los coloreos en busca del mejor resulte prácticamente inviable.

La Optimización Combinatoria es una área de la Investigación Operativa que estudia los problemas con la particularidad que el conjunto F de soluciones donde se busca el óptimo es finito pero grande. Más precisamente, sea $N = \{1,\ldots,n\}$ un conjunto finito, F una familia de subconjuntos de N y una función $f : F \rightarrow \mathbb{R}$. Un problema de Optimización Combinatoria busca el óptimo (máximo o mínimo) de $f(S)$ entre los elementos $S \in F$. Caracterizar los elementos de F de muchos problemas suele ser sencillo. Diferentes órdenes de un conjunto de elementos, apareamientos

entre elementos de dos conjuntos, circuitos de rutas entre ciudades, coloreos de un grafo, entre otros.

Para resolver un problema necesitamos de un *algoritmo*, es decir de un procedimiento sin ambigüedades que consiste de una sucesión finita de pasos a realizar en un orden específico. Para los problemas de Optimización Combinatoria, la enumeración de los elementos de F constituye un algoritmo. Sin embargo, si el cardinal de F es grande, es claro que éste no es un método práctico ni eficiente. Pero, ¿Cuándo consideramos a un algoritmo eficiente? La teoría de complejidad fue iniciada por Cook en 1971 [20] y estableció criterios para decidir si un algoritmo resuelve un problema de manera *eficiente*. Se considera que un algoritmo es eficiente si encuentra la solución de tal manera que, en el peor de los casos, el tiempo requerido (número de operaciones elementales) está acotado por un polinomio en la medida de los datos de entrada. Son los llamados algoritmos polinomiales. Hay problemas para los cuales se dispone de un algoritmo polinomial. Éstos conforman la clase P. Tal es el caso del problema de flujo máximo, el camino más corto entre dos ciudades, entre otros.

Muchos problemas de Optimización Combinatoria pertenecen a una clase de problemas aparentemente difíciles desde el punto de vista computacional. Esta clase es denominada *NP-Hard* en la teoría de la complejidad. Para estos problemas no se conoce un algoritmo que encuentre la solución en tiempo polinomial. Sin embargo, si para alguno de estos problemas se encontrara un algoritmo polinomial, esto implicaría que muchos otros problemas también podrían ser resueltos en tiempo polinomial. Entre los problemas de la clase *NP-Hard*, podemos mencionar el *problema de coloreo de grafos* [51], el *problema de máximo conjunto independiente* (encontrar la mayor cantidad de *vértices* de un *grafo* donde todo par de *vértices* no está conectado por una *arista*) y *el problema de camino hamiltoniano* (encontrar la manera de recorrer todos los *vértices* de un *grafo* pasando una y sólo una vez por cada uno). El libro de Garey y Johnson [34] y el de Papadimitriou y Steiglitz [69] son buenas referencias donde se pueden encontrar los conceptos de complejidad y una extensa clasificación de problemas.

Hemos señalado que el *problema de coloreo de grafos* pertenece a la clase *NP-Hard*. Sin embargo, como ocurre con muchos otros problemas, ciertas particularidades del *grafo* hacen que resulte más sencillo resolver el problema. Por ejemplo, si todo par de *vértices* de un *grafo* están conectados por una *arista*, se necesitan tantos colores como *vértices* tenga el *grafo* para colorearlo. No es éste el único caso, hay caracterizadas otras familias de *grafos* para las cuales se dispone de un algoritmo polinomial para resolver el problema. En [34] se presentan varias de estas familias.

Esencialmente, la resolución de los problemas que pertenecen a la clase *NP-Hard* está enfocada desde dos puntos de vista. Por un lado, con algoritmos que si bien no garantizan encontrar el óptimo brindan *buenas* soluciones en tiempos razonables. Estos son denominados algoritmos de aproximación y algoritmos heurísticos. Por otro lado, si bien la pertenencia a la clase *NP-Hard* indica que el tiempo requerido para encontrar la solución óptima puede resultar prohibitivo, los algoritmos exactos no son dejados de lado. La posibilidad de resolver en forma exacta instancias cada vez más grandes aumenta debido al desarrollo de mejores algoritmos exactos y a la aparición de nueva tecnología. El *problema de coloreo de grafos* ha sido abordado bajo los dos enfoques que hemos mencionado. En el próximo capítulo haremos una reseña de los mismos.

1.4. Objetivo de la Tesis

Muchos problemas de Optimización Combinatoria pueden ser modelados como problemas de programación lineal entera. En estos modelos el objetivo es buscar el óptimo de una función lineal donde algunas o todas las variables están restringidas a ser enteras y deben verificar un sistema de desigualdades lineales. Si en particular las variables deben tomar valores 0 ó 1, se denomima problema de programación lineal entera binaria.

Dado un problema de Optimización Combinatoria asociado a una conjunto F, para cada elemento $S \in F$ podemos considerar su vector característico X^S, es decir $X_i^S = 1$ si $i \in S$ y 0 en caso contrario. Si la condición de pertenencia a F puede ser expresada como inecuaciones lineales en las variables X^S y la función f a optimizar es lineal, la modelización del problema como un problema de programación lineal entera es directa.

Si bien el problema general de programación entera pertenece a la clase *NP-Hard*, la gran cantidad de trabajos en el área permite contar con herramientas que posibilitan día a día resolver eficientemente instancias correspondientes a problemas de la vida real. En particular, las técnicas *Branch-and-Cut* basadas en la teoría poliedral han demostrado ser una de las mejores para tratar este tipo de problemas. El *problema del viajante* [68], *conjunto estable* [67] y *orden lineal* [39] son algunos de los ejemplos exitosos de la aplicación de esta metodología.

Cuando en un problema de programación lineal entera existen múltiples soluciones con el mismo valor de la función objetivo, los algoritmos habituales disminuyen en gran medida su performance. En el Capítulo 3 analizamos con más detalle esta afirmación

En el caso del *problema de coloreo de grafos*, hay muchos coloreos que utilizan la misma cantidad de colores. Basta pensar que a partir de un coloreo que usa k colores, cualquier elección de un subconjunto de cardinal k del conjunto $\{1,\ldots,n\}$ es un coloreo con k colores. Es decir que, debido a la indistinguibilidad de los colores, para cada k hay por lo menos $\binom{n}{k}$ coloreos que pueden ser considerados *equivalentes* o *simétricos*. Por otro lado, también puede ocurrir que un *vértice* pueda tener más de una alternativa de asignación de color usando la misma cantidad total de colores. Dependiendo de la estructura particular del *grafo*, esto puede dar origen a un número muy grande de coloreos *simétricos*. Ahora bien, si todos los coloreos *simétricos* están representados por soluciones factibles del modelo de programación lineal entera y no se contempla una manera de diferenciarlas o eliminarlas, la simetría se transmite al modelo.

El objetivo de esta tesis es abordar la resolución del *problema de coloreo de grafos* mediante modelos de programación entera binaria que parcialmente eliminen soluciones simétricas. Con este fin, proponemos varios modelos que tratan la simetría del problema con diferentes criterios. Estudiamos los poliedros asociados a estos modelos y desarrollamos e implementamos un algoritmo *Branch-and-Cut* donde utilizamos este estudio poliedral y estrategias particulares que guian la búsqueda evitando explorar sobre soluciones simétricas. A este nuevo algorimto lo denominamos *BC-Col.*

El trabajo está dividido en 7 Capítulos. En el Capítulo 2 hacemos una reseña de algunos de los algoritmos para colorear un *grafo* que han sido propuestos en la literatura. Están clasificados

en dos grandes categorías: heurísticos y exactos. Como nuestro enfoque es modelar el problema de coloreo con un problema de programación lineal entera binaria, en el Capítulo 3 introducimos los conceptos básicos de la programación lineal entera binaria y describimos los principales algoritmos usados para su resolución. En el Capítulo 4 proponemos varios modelos de programación entera binaria para el *problema de coloreo de grafos*. Los modelos presentados difieren en la manera de eliminar soluciones *simétricas*. Hacemos un análisis de las ventajas y desventajas de cada uno de ellos. Algunos eliminan más soluciones que otros, sin embargo, observamos que esto no garantiza la supremacía de un modelo sobre el otro desde el punto de vista de un estudio poliedral ni algorítmico. En el Capítulo 5 se presenta un estudio poliedral de uno de los modelos del Capítulo 4. El conocimiento del poliedro asociado a un modelo de programación lineal entera juega un rol fundamental en el desarrollo de algoritmos eficientes. Los resultados de este Capítulo constituyen uno de los ejes principales de nuestro algoritmo *BC-Col*, un algoritmo *Branch-and-Cut* que presentamos en el Capítulo 6. Allí detallamos los procedimientos y las diferentes alternativas que fueron consideradas dentro del esquema general de un algoritmo *Branch-and-Cut*. El comportamiento del algoritmo es analizado en el Capítulo 7 sobre *grafos* generados al azar y sobre el conjunto de instancias de DIMACS [29] [19], librería de *grafos* provenientes de distintas fuentes utilizada por los investigadores para testear nuevos algoritmos. Finalmente, en el Capítulo 8 concluímos con futuras líneas de investigación.

En la próxima sección introducimos la notación y conceptos básicos de la teoría de *grafos* y poliedros que usaremos en esta tesis.

1.5. Definiciones Básicas y Notación

Un *grafo* $G = (V, E)$ consiste de un conjunto V de *vértices* y un conjunto E de *aristas*. Cada *arista* es un par no ordenado de *vértices*. El número de *vértices* de G lo notaremos n y m al número de *aristas*.

Si $[u, v] \in E$, llamamos a u y v los extremos de la *arista*. Un *vértice* v es adyacente a otro *vértice* u si la *arista* $[u, v] \in E$.

La vecindad de un *vértice* v es el conjunto $N(v)$ de todos los *vértices* adyacentes a v. El grado de v, $\delta(v)$, es el cardinal de $N(v)$. Si $\delta(v) = 0$, se dice que el vértice v es aislado. Si $\delta(v) = n - 1$, se dice que el vértice v es universal.

Un *grafo* G es completo si todo par de *vértices* es adyacente. La densidad de un *grafo* $G = (V, E)$, $dens(G)$, es el porcentaje de *aristas* que tiene G respecto de un *grafo* completo con igual cantidad de *vértices*, es decir $dens(G) = 100(2|E|/|V|(|V| - 1))$.

Dado $V' \subset V$, denotamos por $G[V']$ al *grafo* cuyo conjunto de *vértices* es V' y el conjunto de *aristas* es el subconjunto de E cuyos extremos son *vértices* de V'. A $G[V']$ se lo denomina *subgrafo inducido* por V'.

Sea $K \subset V$. K es una *clique* si $G[K]$ es completo. Diremos que la *clique* es maximal si no existe $v \in V \backslash K$ tal que v es adyacente a todos los *vértices* de K. Una *clique* es máxima si no existe una *clique* maximal de mayor cardinal. Denotamos por $\omega(G)$ al cardinal de una *clique* máxima de G.

Sea $I \subset V$. I es un *conjunto independiente* si $G[I]$ no tiene *aristas*. Diremos que el conjunto independiente es maximal si no existe $v \in V \backslash I$ tal que v no es adyacente a los *vértices* de I. El conjunto independiente es máximo si no existe una conjunto independiente maximal de mayor

cardinal. Denotamos por $\alpha(G)$ al cardinal de un conjunto independiente máximo de G.

Sea $P_k \subset V$, $P_k = \{v_1, \ldots, v_k\}$. P_k es un *camino* si $G[P_k]$ tiene las aristas $[v_i, v_{i+1}]$ para $i = 1, \ldots, k-1$ y ninguna otra arista entre *vértices* de P_k.

Sea $C_k \subset V$, $C_k = \{v_1, \ldots, v_k\}$. C_k es un *ciclo* si $G[C_k]$ tiene las aristas $[v_1, v_k], [v_i, v_{i+1}]$ para $i = 1, \ldots, k-1$. Si no existe ninguna otra arista entre *vértices* de C_k, entonces se denomina *agujero*.

Decimos que $\bar{G} = (V, E')$ es el *grafo* complemento de $G = (V, E)$ si $[u, v] \in E'$ si y sólo si $[u, v] \notin E$.

Un *grafo* G es conexo si para cualquier par de *vértices* u, v existe una secuencia $u, v_1, \ldots, v_k, v$ tal que $[u, v_1]$,$[v_i, v_{i+1}]$ para $i = 1, \ldots, k-1$ y $[v_k, u] \in E$.

Sean $I_i \subset V$ conjuntos independientes para todo $i = 1, \ldots, r$. $F = \{I_1, \ldots, I_r\}$ define una partición en conjuntos independientes de V si y sólo si $I_i \cap I_j = \emptyset$ para todo $i \neq j$ y además $\cup_{i=1}^{r} I_i = V$. Si para algún par de índices i, j se satisface que $I_i \cap I_j \neq \emptyset$ entonces F es un cubrimiento en conjuntos independientes de V.

Sean $K_i \subset V$ *cliques* para todo $i = 1, \ldots, r$. $F = \{K_1, \ldots, K_r\}$ define una partición en *cliques* de V si y sólo si $K_i \cap K_j = \emptyset$ para todo $i \neq j$ y además $\cup_{i=1}^{r} K_i = V$. Si para algún par de índices i, j se satisface que $K_i \cap K_j \neq \emptyset$ entonces F es un cubrimiento de *cliques* de V.

En esta tesis asumiremos que los *grafos* son conexos y no completos.

Un *coloreo* de G es una asignación entre V y el conjunto {1,...,n} (colores) de manera tal que dos *vértices* adyacentes no reciben el mismo color.

Un *(k)-coloreo* de G es un coloreo que utiliza k colores distintos. El número cromático de G, $\chi(G)$, es el menor k para el cual existe un *(k)-coloreo* de G. Un *(k)-coloreo* de G define una partición en conjuntos independientes de V que tiene cardinal k.

Un poliedro $P \subset \mathbb{R}^n$ es el conjunto de puntos que satisfacen un número finito de desigualdades lineales, es decir $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ con $A \in \mathbb{R}^{mxn}$ y $b \in \mathbb{R}^m$.

Sea $S \subset \mathbb{R}^n$. Un punto x es una combinación convexa de puntos de S si existen $x_1, x_2, \ldots, x_k \in S$ tal que $x = \sum_{i=1}^{k} \lambda_i x_i$ con $\sum_{i=1}^{k} \lambda_i = 1$ y $\lambda_i \geq 0$ para todo $i = 1, \ldots, k$. La cápsula convexa de S, $conv(S)$, es el conjunto de todas las combinaciones convexas de puntos de S. Equivalentemente, $conv(S)$ es el menor poliedro que contiene a S.

Un conjunto de puntos $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ es afinmente independiente si la única solución de $\sum_{i=1}^{k} \alpha_i x_i = 0$ y $\sum_{i=1}^{k} \alpha_i = 0$ es $\alpha_i = 0$ para todo $i = 1, \ldots, k$.

Un poliedro $P \subset \mathbb{R}^n$ tiene dimensión k si el número máximo de puntos afinmente independientes en P es $k+1$.

La desigualdad $\pi x \leq \pi_0$ es una desigualdad válida para P si $\pi x \leq \pi_0$ para todo $x \in P$. Una desigualdad válida define una *cara* de P si el poliedro $F = \{x \in P : \pi x = \pi_0\} \neq \emptyset$. Una cara se llama faceta si $dim(F) = dim(P) - 1$. La cantidad de caras de P es finita.

Capítulo 2

Algoritmos para el *Problema de Coloreo de Grafos*

El *problema de coloreo de grafos* ha sido abordado por muchos investigadores. Dependiendo de la aplicación y del objetivo buscado, han sido propuestos una gran variedad de modelos y técnicas. Esto es una natural consecuencia de la pertenencia del problema de coloreo a la clase *NP-Hard*. Como ya hemos mencionado en el Capítulo anterior, los métodos de resolución están encuadrados en dos grandes familias: algoritmos heurísticos y algoritmos exactos. En este Capítulo presentamos los principales enfoques algorítmicos del problema, dejando para el Capítulo 4 el abordaje del problema desde el punto de vista de programación entera que es el objetivo principal de esta tesis.

2.1. Algoritmos Heurísticos

En muchos problemas de la vida real se requiere tomar decisiones que no pueden esperar y que no siempre es imprescindible que sean óptimas. Si no se dispone de un algoritmo exacto eficiente o la instancia es lo suficientemente complicada, no puede encararse la resolución del problema esperando obtener el óptimo. Es en estos casos donde las heurísticas juegan un rol fundamental.

Los algoritmos heurísticos son técnicas que permiten obtener buenas soluciones, no necesariamente óptimas, en tiempos razonables. En general, poco puede decirse de la proximidad al óptimo de la solución obtenida. Si el algoritmo asegura una cota superior del error cometido, se denomina algoritmo aproximado. En estos casos se tiene cierta garantía de la calidad de la solución que permite evaluarla con mayor certeza. Al desarrollar un algoritmo heurístico es conveniente que garantice buenas cotas superiores. Sin embargo, Garey y Johnson [34] han demostrado que si existiera un algoritmo polinomial para el *problema de coloreo de grafos* que usara a lo sumo $c\chi(G)$ colores ($c > 1$), entonces existiría un algoritmo polinomial para determinar $\chi(G)$.

A pesar de la falla en conseguir el óptimo o de poder garantizar buenas cotas superiores, son algoritmos importantes por varias razones. Por un lado, proveen soluciones aceptables a una gran variedad de problemas difíciles para los cuales los algoritmos exactos conocidos son incapaces

de dar una solución en tiempo razonable. Por otro lado, el conocimiento de buenas soluciones permite reducir el conjunto de soluciones factibles descartando aquellas cuyo valor sea peor al obtenido. De esta manera, un algoritmo exacto puede buscar en un conjunto más chico y tener la posibilidad de encontrar el óptimo para instancias en las cuales inicialmente es imposible este abordaje. Cabe destacar que en la práctica muchos problemas de gran dificultad han sido resueltos no sólo en tiempos cortos sino que en muchos casos encontrando soluciones muy cercanas al óptimo. Dentro de las técnicas heurísticas aplicadas al *problema de coloreo de grafos*, podemos encontrar diversos métodos como: métodos constructivos, de descomposición, de reducción y de búsqueda local. Algunos de ellos son desarrollados específicamente para el problema en cuestión y otros encuadrándose dentro de un esquema general denominado Metaheurística.

2.1.1. Heurísticas

Los métodos más sencillos utilizan un procedimiento de coloreo secuencial que corresponde a un algoritmo de tipo *goloso*. Estos esquemas son constructivos y encuentran una solución que no es sometida posteriormente a ningún proceso de mejoramiento. Básicamente, la mayoría de estos procedimientos consisten en colorear un *vértice* por iteración siguiendo un orden establecido para los *vértices* o para los colores.

- ***Algoritmos de Coloreo Secuencial Siguiendo un Orden de Vértices***

  Estos algoritmos eligen en la iteración i el *vértice* que se encuentra en la posición i de la lista ordenada de *vértices* y le asignan el primer color factible entre los colores usados por los *vértices* ya coloreados. En el caso que esto no sea posible, un nuevo color es usado para colorear el *vértice*.

  Las implementaciones siguiendo esta línea, difieren en el criterio utilizado para ordenar los *vértices*. Las más usadas son:

  - **LF** (*Largest First*)
    Se consideran los *vértices* ordenados en forma decreciente por grado. Fue propuesto por Welsh y Powell en [77].
  - **SL** (*Smallest Last*)
    Se comienza eligiendo el *vértice* de menor grado y se lo coloca último en la lista. En el paso k, se elige el *vértice* de menor grado en el *subgrafo inducido* resultante de la eliminacion de los *vértices* ya ordenados. Este *vértice* es colocado en el lugar k de la lista comenzando del final de la lista. Este orden fue propuesto por Matula, Marble e Isaacson en [60].

  Ambos métodos son sencillos de implementar, pero no es difícil construir *grafos* donde el coloreo obtenido está muy lejos de ser el óptimo.

  Los colores también pueden estar ordenados siguiendo varios criterios. Un orden fijo inicial o un orden dinámico considerando la cantidad de *vértices* ya coloreados por cada uno de ellos. Cualquiera de estos órdenes pueden ser tomado en forma creciente o decreciente.

  Dentro de este esquema heurístico de coloreo secuencial de *vértices*, el algoritmo más difundido es DSATUR, desarrollado por Brélaz [10]. La diferencia fundamental con los mencionados

antes es que considera un ordenamiento dinámico de los *vértices*. En cada iteración, se elige el *vértice* que tenga la mayor cantidad de colores distintos usados en su vecindad y se lo colorea con el primer color factible. Este algoritmo ha servido como base a muchas variaciones, fundamentalmente en el criterio para elegir el *vértice* en el caso de empate en el ordenamiento de los *vértices*. Brélaz desempata elegiendo el *vértice* de mayor grado. En [73], Sewell propone otro criterio. Si v es un *vértice* del empate, para cada color c factible para v se calcula la la cantidad de vecinos que podrían utilizar el color c. Se calcula la suma de estas cantidades y el *vértice* con mayor suma es elegido. La idea de este criterio es elegir el *vértice* que más reduce el número de colores factibles para los *vértices* aún no coloreados.

Para mejorar la performance de estos métodos, puede considerarse la posibilidad de realizar intercambios de colores. Supongamos que estamos en la iteración k y que ninguno de los colores utilizados en los $k-1$ *vértices* ya coloreados es factible para el *vértice* actual. Si cambiar el color de algún *vértice* permite que al actual se lo coloree sin necesidad de utilizar un nuevo color, se realiza este intercambio. Esta modificación fue sugerida por Matula, Marble e Isaacson en [60] que aplicada a los ordenes descriptos anteriomente son los algoritmos *Largest First with Interchange* (**LFI**) y *Smallest Last with Interchange* (**SLI**) respectivamente. Si bien los intercambios consumen mayor tiempo, mejoran los resultados obtenidos por **LF** y **SL**.

- ***Algoritmos de Coloreo Secuencial Siguiendo un Orden de Colores***

  Los métodos de coloreo secuencial siguiendo un orden en los colores colorean la mayor cantidad posible de *vértices* con el color i antes de proseguir con el color $i+1$. Los *vértices* que pueden ser coloreados con un mismo color corresponden a *conjuntos independientes* del *grafo*. Por lo tanto, estos métodos pueden ser pensados como un procedimiento que va construyendo conjuntos independientes maximales y asignándoles un color. El criterio de selección de los *vértices* para formar los conjuntos independientes da origen a las distintas implementaciones de este esquema básico. En el método **RLF** (*Recursive Largest First*) desarrollado por Leighton [57], los *vértices* a ser coloreados con el color i son elegidos en orden creciente de acuerdo a la cantidad de vecinos no coloreados que no puedan usar el color i. El objetivo de este criterio es colorear la mayor cantidad de *vértices* con el color i y que el *subgrafo inducido* por los *vértices* no coloreados tenga densidad baja.

  A cualquiera de los algoritmos mencionados se le puede añadir un proceso aleatorio. Por ejemplo, Johnson et al [49] desarrollaron el método **XRLF** basado en **RLF**. En este caso, en la iteración i se construyen varios conjuntos de *vértices* candidatos a ser coloreados con el color i. De todos estos conjuntos, se elige el que induzca el *grafo* de *vértices* no coloreados con menor densidad. Para construir los conjuntos de *vértices* candidatos, se aplica varias veces el procedimiento **RLF**. Cada una de las aplicaciones de **RLF** se hace sobre un subconjunto de tamaño fijo de *vértices* elegido al azar entre los *vértices* no coloreados.

2.1.2. Metaheurísticas

Las heurísticas que describimos hasta ahora son constructivas. Es decir, finalizan una vez que logran encontrar una solución factible del problema. Para mejorar el proceso se puede añadir una etapa de mejoramiento. El esquema de mejoramiento más aplicado es el de búsqueda local. Básicamente un procedimiento de búsqueda local trabaja de la siguiente manera: Sea S el conjunto de

soluciones factibles del problema y f la función objetivo. A cada solución $x \in S$ se le asocia una vecindad $N(x)$ de soluciones factibles que pueden ser obtenidas a partir de x mediante una operación llamada *movimiento*. El proceso parte de una solución inicial x_0, calcula $N(x_0)$ y elige x_1 en $N(x_0)$. Bajo este esquema, repitiendo el proceso, en cada iteración del algoritmo se *visita* una solución factible hasta satisfacer alguna condición de parada. Para definir completamente el algoritmo hay que especificar la vecindad de una solución y el criterio para elegir una nueva solución en la vecindad. Esto depende en gran medida de la particularidad del problema a resolver. El criterio más simple de selección de una solución en la vecindad consiste en tomar una solución con mejor valor de f (criterio *goloso*). En este caso, el algoritmo se detendrá en un óptimo local que debido a la *miopía* del proceso de selección puede encontrarse lejos del óptimo global. Para evitar estacionarse en óptimos locales puede resultar beneficioso admitir movimientos en la vecindad hacia soluciones que empeoren el valor de la función objetivo. Este es el principio básico de las Metaheurísticas. Entre las más difundidas se encuentran:

- ***Búsqueda Tabú***

  La técnica de Búsqueda Tabú fue introducida por Glover [36]. Esencialmente el método trata de usar la información de lo que ha sucedido hasta el momento para actuar en consecuencia. Es decir, la vecindad de una solución se verá afectada por la eliminación de aquellas soluciones que ya hayan sido visitadas por el algoritmo en un pasado reciente (*soluciones tabú*) para evitar que el proceso cicle o que tengan atributos que inducen a una estructura poco beneficiosa según se desprenda de las soluciones visitadas.

  El principio subyacente en este método es que es preferible tomar una decisión que localmente sea mala (moverse a una solución con peor valor de f) si está fundamentada en la información que puede extraerse de las soluciones ya visitadas. En este sentido puede decirse que hay un cierto aprendizaje durante el proceso y que la búsqueda es inteligente evitando quedar atrapados en óptimos locales.

  Hertz y de Werra [45] fueron los primeros en aplicar esta técnica al *problema de coloreo de grafos* conocida bajo el nombre de **TABUCOL**. En [30], Dorne y Hao, presentan una implementacion más reciente de esta técnica. Para el caso de *grafos* poco densos, el trabajo reciente de Gonzalez-Velarde y Laguna [38] presenta un procedimiento Tabú que resulta competitivo frente a otras metaheurísticas.

- ***Simulated Annealing***

  Este procedimiento fue propuesto por Kirkpatrick, Gelatt y Vecchi [53]. Siguiendo los lineamientos básicos de un algoritmo de búsqueda local, la propuesta es aceptar todo movimiento de mejora y permitir movimientos de no mejora de acuerdo con una función de probabilidad. La estrategia es comenzar con una probabilidad alta de aceptar movimientos de no mejora y en las sucesivas iteraciones ir disminuyendo la probabilidad de aceptarlos ante la esperable cercanía al óptimo. Así se amplía el espacio de búsqueda en las iteraciones iniciales. Para establecer una función de probabilidad con las características mencionadas, se utiliza la analogía al proceso físico de enfriamiento de un sistema.

  Chams, Hertz y de Werra [13] aplicaron esta técnica al *problema de coloreo de grafos*. Han desarrollado dos tipos de implementaciones. Una en la cual utilizan el esquema descripto y otra híbrida resultado de combinarla con la heurística **XRLF**. En [49], Johnson et al,

realizan una gran cantidad de experimentos usando diferentes variantes del algoritmo sobre un conjunto de *grafos* que varían en densidad y tamaño.

Hasta el momento, en la comparación con otras técnicas metaheurísticas no resulta competitiva para la resolución del *problema de coloreo de grafos*. En general, la performance de Tabú es superior a Simulated Annealing, en particular para *grafos* densos.

- ***Algoritmos Evolutivos***

  La diferencia de los Algoritmos Evolutivos respecto de las otras metaheurísticas es que trabajan con un conjunto de soluciones (*población*) que cambia (*evoluciona*) de una iteración a otra. Fueron propuestos por Holland [47] a partir de la observación del proceso de supervivencia en la evolución natural de los seres vivos.

  A través de las sucesivas generaciones, las distintas especies van evolucionando y ganando *conocimiento* para sobrevivir, que es codificado a nivel de los cromosomas. En el proceso de reproducción se combinan los cromosonas de los progenitores y aquellos con buenas estructuras se reproducen más a menudo que el resto. Se puede realizar una analogía entre el conjunto de soluciones de un problema y el conjunto de individuos de una poblacion natural. La información de cada solución se codifica en un vector binario que juega el rol de cromosonas y se usa la función objetivo del problema para calificar a cada cromosoma. Aquellos cromosomas con mejor evaluación serán elegidos para reproducirse en nuevas soluciones. El algoritmo parte de una población inicial de soluciones, evalúa los cromosomas de las mismas, selecciona y/o elimina cromosonas para realizar combinaciones y mutaciones que darán origen a una nueva población de soluciones.

  Dentro de las aplicaciones de las metaheurísticas al *problema de coloreo de grafos*, los algoritmos evolutivos han resultado los del mejor performance. Podemos destacar el trabajo de Costa, Hertz y Dubuis [22] y el de Fleurent y Ferland [33] cuya implementación combina un algoritmo evolutivo con un proceso de búsqueda local basada en técnicas tabú, a los que son sometidas las soluciones de la población.

- ***GRASP: Greedy Randomized Adaptive Search Procedure***

  Los métodos **GRASP** surgieron en la década del 80 (Feo y Resende [31, 32]). **GRASP** es un proceso iterativo compuesto de dos fases: la fase constructiva y la fase de mejoramiento. Durante la fase constructiva se generan soluciones factibles que en la fase de mejoramiento son sometidas a un proceso de búsqueda local.

  La construcción de soluciones es un proceso iterativo que añade en cada paso un elemento a la solución parcial con criterio goloso (*Greedy*). La evaluación de un elemento no es una propiedad exclusiva del elemento sino que se adapta (*Adaptive*) a la presente solución parcial. Por otro lado, el criterio goloso es aleatorio (*Randomized*) ya que no necesariamente selecciona el mejor candidato, sino que éste es elegido en forma aleatoria de un conjunto de elementos considerados los mejores por el criterio goloso. De esta manera se intenta evitar la repetición de soluciones. A las soluciones construídas en la primera fase, se intenta mejorarlas con búsqueda local, pero en general mediante procesos sencillos y rápidos.

  La buena performance del algoritmo está basada principalmente en realizar muchas iteraciones de fases constructivas que otorgarán un muestreo significativo del conjunto de soluciones del problema y por lo tanto con buena chance de que el óptimo se encuentre entre ellas.

No hay muchos trabajos sobre la aplicacion de GRASP al *problema de coloreo de grafos*. El más reciente es de Laguna y Marti [55] que reportan resultados medianamente satisfactorios para *grafos* generales y con mejor performance en *grafos* de baja densidad. De acuerdo a su experiencia computacional, la *Búsqueda Tabú* y *Simulated Annealing* tienen un comportamiento inferior al GRASP y el algoritmo evolutivo de Fleurent [33] es su mejor competidor. Por otro lado, concluyen que la fase constructiva es de mayor importancia que la de mejoramiento, aún cuando en esta última fase se apliquen heurísticas tales como *Tabú* o *Simulated Annealing*.

2.2. Algoritmos Exactos

Los algoritmos exactos son aquellos que garantizan la solución óptima. Como ya señalamos, el *problema de coloreo de grafos* es *NP-Hard* y por lo tanto, hasta el momento, no ha sido desarrollado un algoritmo que resuelva el problema en tiempo polinomial respecto de alguna medida del *grafo* (cantidad de *vértices* y/o de *aristas*). Sin embargo, gracias a los avances tecnológicos de las computadoras y al perfeccionamiento de los algoritmos, es cada vez mayor la posibilidad de resolver instancias de tamaño hasta no hace mucho tiempo impensable.

La estructura o concepto básico sobre la que están basados muchos algoritmos exactos para problemas de optimización combinatoria es la realización de una *búsqueda inteligente* en el conjunto de las soluciones factibles. Como este conjunto puede tener un cardinal muy grande, saber dónde buscar o saber dónde no es necesario buscar, facilita el proceso de búsqueda y optimiza el tiempo requerido por la misma. Con esta idea, para el *problema de coloreo de grafos* se han desarrollado los Algoritmos Enumerativos y los Algoritmos *Branch-and-Bound*.

2.2.1. Algoritmos Enumerativos

Los coloreos factibles de un *grafo* pueden ser enumerados mediante una estructura de árbol donde cada nodo del árbol representa un coloreo parcial del *grafo*. Recorriendo en forma completa las ramas del árbol podrá encontrarse el coloreo óptimo. Sin embargo, dado que el número de coloreos de un *grafo* es exponencial en el número de *vértices*, el proceso es impracticable. Esto podría evitarse parcialmente si durante el proceso de generación del árbol se dispusiera de reglas que permitan determinar que no es necesario generar ciertas ramas del árbol. Este es el principio básico de los algoritmos de enumeración implícita. Por ejemplo, si ya se encontró un coloreo del *grafo* con k colores, no será necesario recorrer las ramas correspondientes a coloreos que usen k o más colores. De esta manera estaríamos reduciendo el espacio de búsqueda.

A continuación detallamos algunos de los esquemas más conocidos para la generación del árbol.

- *Coloreo Secuencial de Vértices*

  Para generar el árbol se eligen secuencialmente los *vértices* y se considera para cada uno las posibles asignaciones de colores. De esta manera, cada nivel del árbol tiene asociado un *vértice* y a cada uno de los nodos de un mismo nivel le corresponde un color factible para el *vértice*. Más precisamente, sea C_{pq} un coloreo de los *vértices* $v_1, \ldots, v_p$ que usa q colores. Esto representa un nodo del árbol de nivel p. A partir de C_{pq} se generan todos los coloreos de los *vértices* $v_1, \ldots, v_p, v_{p+1}$ dejando fijo los colores de $v_1, \ldots, v_p$ y asignándole sucesivamente

a v_{p+1} los colores posibles de 1 a $q+1$. Cada coloreo obtenido corresponde a un *nodo hijo* del nodo actual. El proceso se repite hasta terminar con el coloreo del *vértice* v_n en cada una de las ramas. Si durante el proceso de enumeración se encuentra un *(k)-coloreo*, los nodos que se generan a partir de la asignación del color k no necesitan ser explorados, reduciéndose de esta manera el espacio de búsqueda.

La elección de que nodo del árbol considerar para generar sus hijos determina la estrategia para recorrer el árbol. Por otro lado, el orden en el cual son considerados los *vértices* del *grafo* para ser coloreados, dará origen a distintas configuraciones de árboles que pueden diferir en tamaño.

Una lista L es una estructura básica que suele utilizarse para implementar la generación y el recorrido de un árbol. En L se guardan los nodos a los cuales aún no se le han generado los hijos. Se los denomina *nodos abiertos*. Una vez que los hijos de un nodo son generados, éste es eliminado de la lista (*nodo cerrado*) y sus hijos son agregados al final de la misma. Inicialmente L esta vacía y al comenzar en la primera iteración contiene el nodo correspondiente al coloreo de v_1 con el color 1.

Las estrategias clásicas de recorrido del árbol son tres:

- **DFS**-*En profundidad:* Se elige el último nodo ingresado a la lista.
- **BFS**-*A lo ancho:* Se elige el primer nodo de la lista
- **BestF**-*Mejor cota:* Se elige el nodo de la lista que corresponda a un coloreo parcial con el menor número de colores.

No hay a priori manera de saber cual es la más conveniente para un problema en particular. En términos generales, una búsqueda **BFS** mantiene mayor cantidad de nodos abiertos en la lista, lo que puede traer problemas de memoria en una implementación. La estrategia **BestF** utiliza en criterio goloso que puede conducir la búsqueda por lugares lejanos al óptimo. Finalmente, **DFS** da prioridad el encuentro de soluciones factibles sin tener en cuenta lo que ocurre en otros nodos.

Respecto al orden en que pueden considerarse los *vértices*, puede utilizarse cualquiera de los mencionados en los algoritmos heurísticos secuenciales: **LF**, **SL**, **DSATUR** y **Sewell**.

Para cualquiera de las estrategias menciondas hay *grafos* que necesitan de una enumeración completa de los nodos del árbol.

En la literatura pueden encontrarse diferentes algoritmos enumerativos, producto de las distintas estrategias consideradas para el recorrido del árbol y para el orden de los *vértices*. Los trabajos de Brélaz [10], Corneil [21], Christopher [16], Sager [72], Sewell [73] se encuadran dentro de esta línea. Si bien el trabajo de Brélaz [10] ya tiene varias décadas, es aún considerado el algoritmo exacto estándar para el *problema de coloreo de grafos* y se lo toma como patrón para comparar los nuevos algoritmos que surgen.

- *Conjuntos Independientes*

  Dado un coloreo, el conjunto de *vértices* coloreados con el mismo color es un conjunto independiente y por lo tanto, un coloreo del *grafo* no es más que una partición del conjunto de *vértices* en conjuntos independientes. Basado en esta propiedad, Christofides [16] propone un

algoritmo que identifica cada nivel del árbol con un color y a cada nodo de un mismo nivel le asocia un conjunto independiente maximal.

Consideremos C_p el coloreo de los conjuntos independientes maximales $I_1, \ldots, I_p$ que usan los colores $1, \ldots, p$ respectivamente. Esto representa un nodo del árbol de nivel p. Sean $G' = G[V \setminus \cup_{i=1}^{p} I_i]$ y $M(G')$ la familia de conjuntos independientes maximales de G'. A partir de C_p, para cada $I \in M(G')$, se genera el coloreo que asigna a I el color $p+1$ y deja fijos los colores de $I_1, \ldots, I_p$. Cada coloreo obtenido corresponde a un *nodo hijo* del nodo actual. El proceso se repite hasta terminar con el coloreo de todos los *vértices* en cada una de las ramas. La estrategia de recorrido puede ser cualquiera de las menciondas antes: **BFS**, **DFS** ó **BestF**. Si ya fue encontrado durante el proceso de enumeración un coloreo con k colores, los nodos de nivel k o superior no necesitan ser explorados reduciéndose de esta manera el espacio de búsqueda.

El mayor inconveniente de este algoritmo es que el cardinal de $M(G')$ puede ser muy grande, obteniéndose un árbol cuya medida haga prácticamente imposible realizar una enumeración. Pueden realizarse una serie de simplificaciones para disminuir el tamaño del árbol. Por ejemplo, Wang [76] propone que en lugar de considerar $M(G')$, se elija un *vértice* $v \in V \setminus \cup_{i=1}^{p} I_i$ y se generen los conjuntos independientes maximales de $G[V \setminus \cup_{i=1}^{p} I_i]$ que contienen a v. Si el *vértice* v es convenientemente elegido, la cantidad de nodos hijos puede reducirse sustancialmente. Esta modificación al esquema básico del algoritmo mejora significativamente su performance.

Existen otras variaciones del algoritmo propuestas por diversos autores [76] pero, en general, la enumeración usando coloreo secuencial de *vértices* es más efectiva.

- *Dicotomía*

  En este caso el árbol de enumeración está asociado a un proceso de división del problema a dos posibilidades mutuamente excluyentes: si v_1 y v_2 son *vértices* no adyacentes, todo coloreo asigna a v_1 y v_2 el mismo color ó distinto color.

  Los coloreos que asignan a v_1 y v_2 distinto color pueden pensarse como los colores del *grafo* G^+ que resulta de agregar al *grafo* original una *arista* entre v_1 y v_2. Los coloreos que asignan igual color a v_1 y v_2 son los coloreos del *grafo* G^- resultante de reemplazar a v_1 y v_2 por un único vértice v cuya vecindad es la unión de las vecindades de v_1 y v_2. Entonces $\chi(G) = min(\chi(G^-), \chi(G^+))$. Esto permite generar recursivamente un árbol binario donde cada nodo hijo representa una de las dos alternativas. El proceso por una rama termina cuando no quedan *vértices* no adyacentes.

  Debido a las operaciones de agregado de aristas y reducción de *vértices*, al final de una rama se obtiene un *grafo* completo que es sencillo de colorear. El coloreo óptimo de G corresponde al *grafo* completo de menor cantidad de *vértices* obtenido por el proceso.

  El algoritmo fue propuesto por Zykov [78] y es bastante costoso en tiempo y memoria. Si bien se pueden hacer algunas mejoras [21], no es un algoritmo competitivo en la práctica.

2.2.2. Algoritmos *Branch-and-Bound*

Los algoritmos *Branch-and-Bound* están asociados al concepto *divide y conquista*: Si resulta difícil buscar el óptimo en un conjunto S, resultará más fácil hacerlo por separado en partes de S

y luego quedarse con la mejor solución.

Aplicado este concepto recursivamente podemos pensar en un *árbol* cuya raíz corresponde al problema original y sus ramas resultan de la división en partes del espacio de búsqueda. A cada nodo del árbol le corresponde un subproblema que consiste en buscar el óptimo en una parte del espacio de soluciones. El tamaño del árbol puede ser muy grande y la clave está en utilizar argumentos de factibilidad u optimalidad que permitan asegurar que el óptimo buscado no se encuentra en ciertas ramas del árbol y por lo tanto no hay necesidad de explorar en ellas. El término *Branch* se refiere al procedimiento de división del espacio de búsqueda en subconjuntos menores. El proceso de *Bound* es el que permite eliminar ramas del árbol.

Una forma de llevar a cabo el proceso de *Bound* es calcular en cada nodo cotas inferiores del óptimo del subproblema. Si la cota es peor que la mejor solución encontrada hasta el momento, no será necesario resolver el subproblema asociado a dicho nodo y las ramas del árbol correspondientes a ese nodo no necesitan ser exploradas. Para lograr una implementación eficiente, el cálculo de las cotas deberá ser hecho con un procedimiento que encuentre rápidamente buenas cotas. Una cota muy holgada hará que se exploren innecesariamente ramas del árbol y por otro lado, un procedimiento que garantice mejores cotas pero que insuma mucho tiempo no se justifica en este tipo de esquema.

Los algoritmos enumerativos que mencionamos anteriormente son casos particulares y simplificados de este esquema general. Los mismos no utilizan procedimientos específicos para el cálculo de cotas inferiores que le permitan eliminar ramas del árbol. Tal vez una de las aplicaciones más difundidas de las técnicas *Branch-and-Bound* es la utilizada para resolver problemas de programación lineal entera. Constituyen un pilar básico para los algoritmos más eficientes que existen para resolver este tipo de problemas. Ya hemos mencionado que muchos problemas de optimización combinatoria, en particular el *problema de coloreo de grafos*, pueden ser modelados usando una formulación de programación lineal entera binaria. Debido a que hemos elegido este enfoque para resolver el problema, en el próximo Capítulo haremos una explicación más detallada de estos modelos y de los algoritmos para resolverlos. Hay trabajos en la literatura para el *problema de coloreo de grafos* que analizamos en el Capítulo 4, donde también proponemos nuevos modelos de programación lineal entera binaria.

En este capítulo hemos presentado lo que creemos son los aportes algorítmicos más importantes que se encuentran en la literatura. La relevancia práctica y complejidad del *problema de coloreo de grafos* capta el interés de muchos investigadores. Esto hace que sea un área donde surgen constantes aportes teóricos y algorítmicos. Sin embargo, muchos de los métodos propuestos son pequeñas variaciones de los esquemas generales básicos encontrados en los trabajos seminales que hemos presentado con más detalle. Consideramos que la clasificación de los métodos y las referencias a los trabajos primarios de los mismos, dan una marco suficiente para tener una idea clara de las metodologías existentes para la resolución del problema.

El sitio mantenido por Joseph Culberson, www.http://web.cs.ualberta.ca/~joe/Coloring, es un excelente lugar donde no sólo se encuentran referencias bibliográficas sino también programas fuente de algunos algoritmos y archivos de datos para testear nuevos desarrollos.

Capítulo 3

Programación Lineal Entera Binaria

Muchos problemas de optimización combinatoria pueden ser modelados con formulaciones de programación lineal entera y, particularmente, binaria. Generalmente el modelo de programación lineal entera es fácil de formular y en muchos casos existen varias alternativas. El objetivo de este Capítulo es describir los conceptos fundamentales y algoritmos para un problema de programación lineal entera binaria. Asumimos que el lector tiene los conocimientos básicos de programación lineal.

3.1. ¿Qué es un Problema de Programación Entera Binaria?

Un problema de programación lineal entera binaria sirve para modelar situaciones donde deban tomarse decisiones por *Si* ó por *No* en forma óptima. Por ejemplo, la inversión en un portfolio de acciones, la localización de recursos (hospitales, central de bomberos, etc), la asignación de colores a *vértices* de un *grafo*, etc. Resulta muy natural en estos casos asociar a cada decisión una variable x restringida a tomar valores 1 ó 0 que representan la respuesta por *Si* ó por *No* respectivamente. Si las variables del problema deben respetar condiciones que pueden modelarse mediante funciones lineales (por ejemplo, opciones x e y incompatibles entre sí deben verificar que $x+y \leq 1$) y además la función a optimizar es lineal, estamos frente a un problema de programación lineal entera binaria.

Formalmente, un problema de programación lineal entera binaria busca el óptimo de una función lineal entre los vectores binarios del espacio que se encuentran en un poliedro caracterizado por inecuaciones y ecuaciones lineales. La formulación responde entonces a la siguiente estructura:

$$
\begin{aligned}
&\textit{Minimizar} \sum_{j=1}^{n} c_j x_j \\
&\textit{sujeto a} \\
&\qquad Ax \leq b \\
&\qquad x_j \in \{0,1\} \qquad \forall\, j = 1,\ldots,n
\end{aligned}
$$

donde $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^m$ y $x \in \{0,1\}^n$.

Este problema en su forma general pertenece a la clase *NP-Hard*, es decir no se conoce un algoritmo polinomial que lo resuelva. Si las variables no estuvieran restringidas a tomar valores enteros (relajación lineal del problema), se tiene un problema de programación lineal. La diferencia esencial es que en este último caso se conoce un algoritmo polinomial para resolverlo [50].

Sea S al conjunto de soluciones factibles binarias del problema, es decir

$$S = \{x \in \{0,1\}^n : Ax \leq b\}$$

y P la cápsula convexa de S (menor poliedro que contiene a S). Existen instancias de problemas de programación entera que tienen la particularidad que pueden ser resueltas en tiempo polinomial. Por ejemplo, si P coincide con el poliedro asociado a la relajación lineal, $RLP = \{x \in \mathbb{R}^n : Ax \leq b \ 0 \leq x \leq 1\}$, basta resolver la relajación lineal del problema para obtener el óptimo buscado. Éste es el caso del conocido *problema de transporte*. Por otro lado, si se conociera la caracterización de la cápsula convexa con un número polinomial de *facetas*, también podríamos resolver el problema con técnicas de programación lineal. Es más, aún en el caso que esta caracterización no fuese polinomial, bajo ciertas circunstancias, existe un algoritmo polinomial. Más adelante mencionaremos un resultado que establece estas condiciones. Sin embargo, para muchos problemas la caracterización completa de la cápsula convexa no ha podido ser lograda.

En aquellos casos en que no se conoce la caracterización de P mediante restricciones lineales, la relajación lineal resulta muy útil. Si ésta resulta un problema no factible, también lo es el problema binario. En el caso que sea factible, el valor óptimo de la relajación es una cota inferior del valor óptimo binario. De aquí la importancia de disponer de una relajación lineal lo más *ajustada* posible para que este valor resulte de utilidad. Si además tiene la particularidad de tener un óptimo binario, éste es la solución del problema de programación binaria. La diferencia entre el valor óptimo del problema lineal entero y de la relajación lineal (*gap*) es una medida que suele usarse para medir la calidad de la relajación. La diferencia entre el valor de una solución factible del problema y el valor de la relajación lineal da una cota superior del *gap*. Esto permite conocer la calidad de una solución y en algunos casos certificar la optimalidad de la misma.

Además de la relajación lineal hay otro tipo de relajaciones que pueden usarse. Algunos problemas tienen un subconjunto de restricciones con cierta estructura que si fueran las únicas, el problema resulta fácil de resolver. Si se elimina el conjunto de restricciones *complicadas* dejando sólo las primeras se obtiene una relajación del problema. El valor óptimo de la relajación es una cota inferior del valor óptimo del problema lineal entero. La relajación lineal es una caso particular donde las restricciones de integrabilidad son las eliminadas. Puede ocurrir que este tipo de relajación otorgue una cota inferior muy débil. Para mejorar esta situación, una posibilidad es adicionar a la función objetivo las restricciones complicadas con un coeficiente de penalidad. Esto es lo que se conoce como *Relajación Lagrangeana*. Un problema puede tener más de una Relajación Lagrangeana, dependiendo de cúal sea la elección de las restricciones complicadas. Es una herramienta muy útil para aplicar en modelos para los cuales se identifica alguna *buena* estructura. En [8], Beasley hace una muy buena reseña de la aplicación de esta técnica.

Muchos de los algoritmos para resolver problemas de programación entera binaria se basan en la relación entre el problema binario y la relajación lineal. En la siguiente sección describimos algunos de los más usados en la práctica.

3.2. Algoritmos para Problemas de Programación Entera Binaria

La mayoría de los métodos de resolución exacta de un modelo de programación lineal entera se encuadra en alguno de los siguientes esquemas:

1. Métodos de Planos de corte
2. Métodos *Branch-and-Bound*
3. Métodos *Branch-and-Cut*
4. Métodos *Branch-and-Price*

3.2.1. Algoritmos de Planos de Corte

La idea fundamental de un algoritmo de planos de corte es considerar la relajación lineal del problema e ir mejorándola con el agregado de desigualdades lineales válidas para P que eliminen (*corten*) soluciones de la relajación RLP. El esquema general del algoritmo comienza relajando el problema omitiendo las condiciones de integralidad de las variables. Si al menos una variable que debe ser entera resultó fraccionaria, se busca identificar una desigualdad lineal válida para P que *separe* la actual solución del conjunto de soluciones factibles enteras. Al agregar esta desigualdad a la formulación se obtiene una nueva relajación del problema más ajustada, sobre la cual puede reiterarse el procedimiento. El éxito de la metodología depende en gran medida de la posibilidad y la eficiencia de encontrar desigualdades violadas (*planos de corte*) que puedan ser agregadas a la formulación para *separar* las soluciones fraccionarias. Es decir de disponer de un buen *algoritmo de separación.*

El esquema básico de un algoritmo de planos de corte es el siguiente:

- **Paso 1**-*Inicialización*: Considerar la relajación lineal del problema
- **Paso 2**-*Resolución de la relajación lineal*: Sea x^* la solución óptima de la relajación. Si x^* tiene sus coordenadas binarias, finalizar. Si no, ir al siguiente paso.
- **Paso 3**-*Separación*: Buscar desigualdades válidas violadas por x^*. Si se encuentran, agregarlas a la formulación e ir al paso 2. Si no, finalizar. El problema no ha podido ser resuelto.

Los planos de corte pueden ser generados bajo dos enfoques:

- *Con herramientas generales aplicables a cualquier problema de programación lineal entera.*

  A comienzos de los 60, Gomory [37] desarrolló un algoritmo general para generar desigualdades válidas que cortan la solución de la relajación. En cada iteración, la desigualdad es

obtenida a partir de la expresión de las variables básicas en función de las no básicas utilizando exclusivamente argumentos de integrabilidad. Es un algoritmo de aplicación general que no usa ninguna otra información adicional de problema y bajo ciertas condiciones es convergente. Este resultado es muy importante desde el punto de vista teórico pero no es una herramienta eficiente para resolver problemas en tiempos razonables. Por otro lado, la implementación computacional es numéricamente inestable.

Otros algoritmos utilizados para generar planos de corte están basados en procedimientos *disyuntivos*. La idea básica es que si $P = P^1 \cup P^2$ y $\pi^i x \leq \pi_0^i$ es una desigualdad válida para P^i para $i = 1, 2$ entonces $\pi x \leq \pi_0$ es válida para P con $\pi_j \leq min(\pi_j^1 \pi_j^2)$ para $j = 1, \ldots, n$ y $\pi_0 \geq \pi_0^1, \pi_0^2$.

En el caso particular que $P^1 = P \cap \{x \in \mathbb{R}^n : x_j = 0\}$ y $P^2 = P \cap \{x \in \mathbb{R}^n : x_j = 1\}$ para algún $j = 1, \ldots, n$, estos procedimientos se conocen con el nombre de *lift and project*. Los trabajos de Balas et al [6] y Sherali et al [74] son buenas referencias de este enfoque.

Si bien estos algoritmos tienen propiedades teóricas de mucho interés, su éxito en la práctica es discutible. Cualquiera de las técnicas menciondas tienen la ventaja de poder ser utilizadas para cualquier problema de programación entera, independientemente de su estructura. Si bien esto es una propiedad deseable en un algoritmo, no siempre brinda la herramienta más adecuada para casos particulares. Un estudio más específico del problema ayuda a obtener mejores procedimientos. Este es el sentido del próximo enfoque.

- *Explotando la estructura particular del problema.*

  Hay propiedades inherentes a cada problema que pueden ayudar a identificar mejores planos de corte. Un trabajo pionero en esta dirección fue el de Dantzig et al [26] en 1954 para el problema del viajante de comercio. La técnica usada permitió resolver una instancia de 49 ciudades (grande para la época) y sentó las bases de lo que hoy en día es una de las herramientas más efectivas: las técnicas poliedrales.

  Una propiedad deseada para un plano de corte es que elimine la mayor cantidad posible de soluciones no enteras. Como los planos de corte son desigualdades válidas de P, es de esperar que resulten de mayor utilidad aquellas desigualdades que resulten ser facetas de P. Un estudio poliedral de P permite disponer de *buenos* planos de corte. Los primeros estudios poliedrales fueron realizados a principios de la década de los 70, para el *problema de conjunto independiente* [67] y el *problema del viajante* [42]. Estos trabajos significaron un importante progreso en la resolución de estos problemas.

  Con fines algorítmicos, el estudio poliedral debe estar acompañado de algoritmos de separación eficientes. En este sentido, hay un resultado muy importante debido a Grötschel, Lovász y Schrijver [40] que relaciona la complejidad del problema de separación con la complejidad del problema de optimización. Se establece que el problema de optimización $max\{cx : x \in conv(S)\}$ puede resolverse polinomialmente si y sólo si el problema de separación ($x \in conv(S)$ *ó encontrar una desigualdad válida violada*) es polinomial. Es decir que si el problema que estamos tratando no es polinomial, existe al menos alguna familia de facetas que no puede separarse en tiempo polinomial. Esto de alguna manera implica el grado de dificultad de encontrar la descripción de todas las facetas de la cápsula convexa.

Por supuesto, la aplicación de esta clase de cortes está limitada al problema particular. Sin embargo, algunas desigualdades que han sido obtenidas a partir de problemas particulares, pueden ser usadas para problemas generales. Desigualdades válidas para el *problema de la mochila* [66] y generalizaciones de las mismas están incluídas como planos de corte en varios de los más importantes paquetes generales, como CPLEX [23].

3.2.2. Algoritmos *Branch-and-Bound*

En el Capítulo 2 describimos al algoritmo *Branch-and-Bound* como un algoritmo que particiona el espacio de soluciones con el fin de facilitar la búsqueda del óptimo. Señalamos que al aplicar este concepto recursivamente se genera un *árbol* cuya raíz corresponde al problema original y sus nodos tienen asociados subproblemas que resultan de la división en partes del espacio de búsqueda. Debido al tamaño que puede alcanzar el árbol, es esencial disponer de herramientas eficientes que permitan eliminar ramas del árbol.

Es el algoritmo tradicional para resolver problemas de programación lineal entera. La implementación más difundida y que se utiliza en la mayoría de los paquetes comerciales utiliza la relajación lineal para el proceso de *Branching* y fundamentalmente de *Bound*. A cada nodo del árbol se le asocia la relajación lineal del problema en el subespacio de búsqueda correspondiente al nodo. En el caso que la solución del problema relajado satisfaga los requerimientos originales de integralidad, ésta es la solución buscada en esa región del espacio. Si el subproblema relajado no tiene solución o su valor óptimo es peor que la mejor solución entera conocida hasta el momento, no hay necesidad de seguir explorando el subconjunto de soluciones asociado al nodo. Por lo tanto, en cualquiera de los casos mencionados la rama del árbol que se genera a partir del mismo puede ser *podada* (proceso de *Bound*). Por el contrario, si al menos una variable de la solución óptima relajada que debe ser entera es fraccionaria y el valor óptimo de la relajación es mejor que la mejor solución binaria que se dispone, no hay razón para detener la búsqueda en esa región del espacio. Para continuar con la misma, se deben generar nuevos nodos (proceso de *Branching*).

Los primeros trabajos con esta técnica se deben a Land y Doig [56], Dakin [25] y Balas [5]. El esquema básico del algoritmo es:

- **Paso 1**-*Inicialización:* Crear una lista L con el nodo raíz y la relajación lineal del problema. Sea $Z_{sup} = \infty$.
- **Paso 2**-*Elección de nodo:* Si L esta vacía, el algoritmo termina. Si no, elegir un nodo de L y eliminarlo de la lista.
- **Paso 3**-*Bound:* Resolver la relajación lineal asociada al nodo. Si no es factible, volver al paso 2. Sea x^* la solución óptima y Z^* el valor de la función objetivo.
  - Si x^* es solución factible del problema, sea $Z_{sup} = min(Z_{sup}, Z^*)$. Volver al paso 2.
  - Si $Z^* \geq Z_{sup}$, en esa rama no existe solución factible mejor que la actual. Volver al paso 2.

- Paso 4-*Branching:* Generar subproblemas del nodo actual y agregarlos a la lista L. Volver al paso 2.

En este esquema básico no está especificada la regla a seguir para la elección de un nodo de la lista ni el proceso de generación de los subproblemas. Respecto a la elección de un nodo, las opciones más usuales son algunas de las que ya mencionamos para los algoritmos enumerativos: **DFS**-*Profundidad* (último nodo de la lista), **BFS**-*Ancho* (primer nodo de la lista) o **BestF**-*Mejor cota* (el nodo con menor valor óptimo de la relajación).

Para generar los subproblemas suele usarse la clásica dicotomía en una variable x_i. Del conjunto de variables fraccionarias de la solución óptima de la relajación, puede elegirse x_i como la variable tal que:

- x_i^* *es el valor más cercano a* 1/2.
- x_i^* *es el valor más cercano a* 0.
- x_i^* *es el valor más cercano a* 1.
- *tiene el menor coeficiente en la función objetivo.*
- *cumple alguna propiedad específica del problema.*

Los dos nuevos nodos que se generan tienen asociada la región del espacio del nodo padre con el agregado de $x_i = 0$ ó $x_i = 1$ respectivamente. Cualquier combinación de estas reglas da origen a un árbol distinto. En [58] se realiza un análisis muy detallado sobre la performance de las distintas estrategias, pero no se llega a una respuesta concluyente sobre la supremacía de una sobre la otra aplicable a cualquier problema.

3.2.3. Algoritmos *Branch-and-Cut*

A comienzos de los 80, Crowder et al [24] tuvieron un gran éxito al aplicar una metodología mixta para resolver problemas binarios. Trabajaron con un algoritmo *Branch-and-Bound* pero, antes de comenzar la primera etapa de *Branching*, aplicaron un algoritmo de planos de corte a la relajación lineal asociada a la raíz del árbol. De esta manera lograron mejorar la cota inferior brindada por la relajación lineal y eso disminuyó el tamaño del árbol explorado. A mediados de la misma década, aparecieron los primeros trabajos que ampliaron la aplicación de planos de corte a otros nodos del árbol. Grötschel et al [39] presentan este enfoque en el *problema de ordenamiento lineal* y Padberg et al [68] en el *problema de viajante de comercio* donde fue introducido el término *Branch-and-Cut.*

El esquema básico del algoritmo es:

- Paso 1-*Inicialización:* Crear una lista L con el nodo raíz y la relajación lineal del problema. Sea $Z_{sup} = \infty$.
- Paso 2-*Elección de nodo:* Si L esta vacía, el algoritmo termina. Si no, elegir un nodo de L y eliminarlo de la lista.

- **Paso 3-*Bound:*** Resolver la relajación lineal asociada al nodo. Si no es factible, volver al paso 2. Sea x^* la solución óptima y Z^* el valor de la función objetivo.
  - Si x^* es solución factible del problema, sea $Z_{sup} = min(Z_{sup}, Z^*)$. Volver al paso 2.
  - Si $Z^* \geq Z_{sup}$, no existe solución factible mejor que la actual. Volver al paso 2.
- **Paso 4-*Branching vs Cutting:*** Decidir si se buscarán planos de corte.

  No: ir al paso 6 (*Branching*). Si: ir al paso 5 (*Separación*).
- **Paso 5-*Separación:*** Buscar desigualdades válidas violadas por x^*. Si no se encuentran, ir al paso 6 (*Branch*). Si se encuentran, agregarlas a la formulación e ir a paso 3 (*Bound*).
- **Paso 6-*Branching:*** Generar subproblemas del nodo actual y agregarlos a la lista L. Volver al paso 2.

En la descripción del algoritmo quedan muchos puntos sin especificar. Por ejemplo, ¿Cuántas iteraciones realizar del algoritmo de planos de corte?, ¿Cuántos cortes agregar por iteración?, ¿Qué hacer con los cortes generados en los distintos nodos del árbol?. La performance del algoritmo depende de estos factores y de muchos otros. En la práctica, lograr un equilibrio entre ellos no es tarea fácil y depende en gran medida del problema particular que se quiere resolver.

En términos generales, podemos señalar dos factores fundamentales de los buenos resultados que pueden obtenerse con esta técnica:

- Utilizar planos de corte que a pesar de ser generados en un nodo del árbol son válidos para todo el árbol. Esto permite aprovechar el trabajo de identificación de cortes en un nodo para cualquier otro y disminuye el requerimiento de memoria necesario para cada nodo del árbol. En la práctica, se dispone de un espacio de memoria común (*pool de cortes*) donde se almacenan las desigualdades y para cada subproblema basta referenciar las desigualdes que correspondan a la formulación asociada al mismo.
- Utilizar desigualdes válidas específicas del problema, resultado del estudio de su estructura poliedral.

El éxito computacional para resolver problemas de programación lineal entera basado en los puntos que señalamos, explican el creciente uso de las técnicas *Branch-and-Cut* conjugadas con la teoría poliedral en el desarrollo de algoritmos. La identificación de desigualdades válidas y en lo posible facetas del poliedro de soluciones factibles, usadas en el contexto de un algoritmo *Branch-and-Cut*, permite resolver instancias de problemas de programación entera que no resulta posible con otro procedimiento. Los trabajos [46], [71], [54] y [9] son una muestra de esta afirmación.

3.2.4. Algoritmos *Branch-and-Price*

Los algoritmos *Branch-and-Price* son una generalización de los algoritmos *Branch-and-Bound* y surgieron en los años 80. Están diseñados especialmente para formulaciones que tienen un gran

número de variables que no puede ser considerado explícitamente. Estos modelos no son una rareza y son usados en la práctica por diversas razones. Puede ocurrir que sean los únicos modelos conocidos para un problema y no exista otra alternativa. Puede suceder también que modelos con menor cantidad de variables tengan relajaciones más débiles y no resulten buenos para aplicarles los algoritmos conocidos.

Debido al tamaño del modelo, la resolución de las relajaciones asociadas a cada nodo del árbol es muy costosa. La estrategia en estos casos es resolver la relajación lineal restringiéndose a un subconjunto de variables. La solución obtenida es factible de la relajación original pero no necesariamente óptima. Para chequear la optimalidad hay que buscar entre las variables no consideradas si hay alguna candidata a entrar a la base. Es decir, si y^* es la solución del dual del problema restringido, hay que buscar una columna a del modelo tal que $c_a - y^*a < 0$ donde c_a es el coeficiente de costo de la variable correspondiente a la columna a. Si existe una columna en tales condiciones, se reoptimiza la relajación. Si no, se continúa con el esquema general del algoritmo *Branch-and-Bound*. Esta técnica para resolver las relajaciones se denomina *generación de columnas*.

Como no se disponde de todas las columnas en forma explícita, el procedimiento usado para chequear la optimalidad de la solución es clave para la aplicación de esta técnica. Muchas veces, las columnas de la matriz de restricciones pueden describirse en forma implícita como vectores característicos de subconjuntos de un conjunto. Por ejemplo, dado un conjunto de cajas de diferentes pesos, subconjuntos de cajas con peso total inferior a un valor fijo. En estos casos, el problema de generación de columna puede ser formulado mediante un modelo de programación lineal entera que puede ser resuelto mediante alguna heurística o algoritmo exacto. En el caso del ejemplo considerado, obtenemos un problema de mochila: entre todos los posibles subconjuntos de cajas que no superan cierto peso, elegir el de menor costo reducido. Uno de los primeros trabajos utilizando esta técnica de generación de columnas es de Gilmore y Gomory [35] para el *problema de cutting stock*.

En el contexto de un algoritmo *Branch-and-Price*, se necesita de un buen algoritmo de generación de columnas que permita resolver las relajaciones sin consumir mucho tiempo. Esto depende esencialmente de la estructura del problema de generación y de que las estrategias de *Branching* no modifiquen o compliquen esta estructura. A pesar de estas dificultades, es una técnica utiliza con éxito para muchos problemas entre los cuales se encuentra el *problema de ruteo de vehículos* y el *problema de planificación* [28].

Aún en los casos que no es posible encontrar la solución de un problema de programación lineal entera binaria, los procedimientos descriptos resultan de utilidad. El valor óptimo de la relajación lineal es una cota inferior del valor óptimo entero. Lo algoritmos tienen la particularidad de ir mejorando iterativamente esta cota inferior y en el caso de *Branch-and-Bound* y *Branch-and-Cut* pueden ir surgiendo mejores soluciones enteras que permiten ir actualizando la cota superior. En la práctica, generalmente, no resulta imprescindible contar con el óptimo y basta con conocer una medida del error que se comete con soluciones aproximadas. El conocimiento de una buena cota inferior y superior permite estimar el porcentaje de error respecto al valor óptimo de la solución aproximada e incluso llegar a probar su optimalidad.

En los algoritmos que hemos descripto, el mejoramiento de la cota inferior es logrado por la aplicación sucesiva de planos de corte y/o por el proceso de *Branching*. Sin embargo, si el modelo tiene múltiples soluciones equivalentes (desde el punto de vista de la función objetivo) puede ocurrir que se requieran muchas iteraciones para lograr un incremento de la cota inferior o determinar que ya se encuentró el óptimo. Por eso, es importante evitar trabajar con modelos que tengan esta característica. Esta situación puede ser una de las causantes de la baja performance de estos algoritmos. Con el fin de evitar esta clase de simetría, resulta también muy útil implementar estrategias de *Branching* que no consideren regiones del espacio simétricas desde el punto de vista del valor de la función objetivo. Se evita así una búsqueda innecesaria en lugares del espacio donde se puede afirmar que no existen mejores soluciones.

Ninguno de los métodos presentados deriva en algoritmos polinomiales y es muy difícil estimar el tiempo que requerirá obtener la solución de una instancia de un problema de programación entera binaria. El tamaño, cantidad de variables y/o restricciones, puede darnos una idea muy global de la dificultad, pero no es suficiente. Esta última observación puede llevarnos a tener una visión pesimista de la situación. Sin embargo no es así. Hay un gran esfuerzo de muchos investigadores en lograr implementaciones cada vez más eficientes y en desarrollar nuevas estrategias. Esto sumado a los avances tecnológicos, posibilita que hoy en día dispongamos de herramientas sólidas que resuelven instancias de problemas de programación entera intratables hasta no hace mucho tiempo.

Capítulo 4

Modelos de PLEB para el *Problema de Coloreo de Grafos*

En el Capítulo anterior explicamos los conceptos básicos y algoritmos de los problemas de programación entera. Señalamos algunas de las dificultades que aparecen en la práctica y en particular mencionamos que la presencia en el modelo de soluciones simétricas perjudica la performance de los algoritmos. El objetivo de este Capítulo es presentar y analizar modelos de programación lineal entera para el *problema de coloreo de grafos*. Comenzamos describiendo dos modelos clásicos que aparecen en la literatura y finalmente proponemos tres nuevos modelos que, con diferentes criterios, eliminan soluciones simétricas. Hacemos un análisis de las ventajas y desventajas de cada uno de ellos.

4.1. Modelo Clásico

La formulación clásica para el *problema de coloreo de grafos* fue propuesta por Christofides [17]. Consideremos w_j y x_{ij} variables binarias, para todo $i \in V$ y $1 \leq j \leq n$, donde

$$x_{ij} = \begin{cases} 0 & \text{si el } \textit{vértice } i \text{ no es coloreado con el color } j \\ 1 & \text{si el } \textit{vértice } i \text{ es coloreado con el color } j \end{cases}$$

$$w_j = \begin{cases} 0 & \text{si el color } j \text{ no es usado por ningún } \textit{vértice} \\ 1 & \text{si el color } j \text{ es usado por algún } \textit{vértice} \end{cases}$$

A partir de estas variables, la formulación clásica de programación entera es:

$$\textit{Minimizar} \quad \sum_{j=1}^{n} w_j$$

sujeto a

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\, i \in V \tag{4.1}$$

$$x_{ij} + x_{kj} \leq w_j \qquad \text{si } [i,k] \in E \;\; \forall\, j = 1,\ldots,n \tag{4.2}$$

$$x_{ij}, w_j \in \{0,1\} \qquad \forall\, i \in V \;\; \forall\, j = 1,\ldots,n$$

donde las restricciones de *asignación* (4.1) aseguran que cada *vértice* sea coloreado con un color y las restricciones de *adyacencia* (4.2) imponen que dos *vértices* adyacentes no compartan el mismo color y que $w_j = 1$ cuando algún *vértice* es coloreado con el color j. El modelo tiene $n^2 + n$ variables y $n + nm$ restricciones. Al poliedro de soluciones factibles asociado a esta formulación lo denomimamos $\mathcal{SCP}$.

Cuando describimos el *problema de coloreo de grafos*, mencionamos la propiedad de simetría. Esto se ve reflejado en este modelo. Todo coloreo está representado por una solución factible de la formulación. Dada una solución entera de $\mathcal{SCP}$ cualquier permutación de los colores es una solución factible con el mismo valor de la función objetivo. Es decir que cualquier solución tiene un número exponencial de soluciones simétricas o equivalentes. Lo mismo ocurre con las soluciones fraccionarias. Por ejemplo, sea $k = 2,\ldots,n$ y $N_k \subset \{1,\ldots,n\}$ tal que $|N_k| = k$. La solución $w_j = 2/k$ y $x_{ij} = 1/k$ para todo $i \in V$ y $j \in N_k$ tiene valor objetivo 2. Hay un número exponencial de soluciones fraccionarias simétricas ó equivalentes.

Si pensamos en aplicar un algoritmo *Branch-and-Bound* o *Branch-and-Cut* para resolver el problema, esta característica de simetría del modelo puede conducir a una exploración innecesaria de ramas del árbol. Además, la existencia de múltiples óptimos, entorpece el mejoramiento de las cotas inferiores. Los planos de corte eliminan la solución de la relajación, sin embargo la existencia de óptimos alternativos fraccionarios determina que el algoritmo no pueda mejorar el valor de la relajación en sucesivas iteraciones.

No hemos encontrado en la literatura un algoritmo *Branch-and-Cut* basado en un estudio poliedral de $\mathcal{SCP}$ ni con estrategias que intenten eliminar simetrías. En [54] y [1], la formulación es utilizada para modelar un problema de asignación de frecuencias, si bien con el agregado que cada *vértice* tiene un conjunto de colores (frecuencias) posible. El modelo es resuelto con un algoritmo *Branch-and-Cut* si bien los planos de corte no son específicos del problema. Usan desigualdes válidas del poliedro del *problema de conjunto independiente máximo* [67], que es una relajación de $\mathcal{SCP}$. Proponen dos estrategias de *Branching*, pero ninguna de ellas pensadas para evitar soluciones simétricas.

En el comienzo de nuestro trabajo, realizamos un estudio del poliedro $\mathcal{SPC}$ [18] e implementamos un algoritmo *Branch-and-Cut* con la desigualdades válidas obtenidas. Si bien los resultados no fueron muy satisfactorios, nos sirvieron como base para lograr una mejora del modelo y de

los algoritmos a partir del análisis de las dificultades que se presentaron. Al final del Capítulo 5, hacemos un resumen de nuestros resultados poliedrales sobre $\mathcal{SCP}$.

En el Capítulo anterior mencionamos que el tamaño de una formulación de programación lineal entera da una idea de la dificultad de la resolución, si bien no es un factor concluyente. Muchas veces, cierto pre-procesamiento de los datos permite reducir este tamaño y marca la diferencia entre la posibilidad de poder o no resolver el problema. En el caso particular del *problema de coloreo de grafos*, el conocimiento de la estructura del *grafo* permite tomar decisiones acerca de la coloración de ciertos *vértices*. Por ejemplo, si K es una *clique* de G, todos sus *vértices* usan colores distintos y por lo tanto $|K| \leq \chi(G)$. Si coloreamos los *vértices* de K con los coloreos $1, \ldots, k$, bastará buscar un coloreo en $G[V \backslash K]$ restringiendo los colores de los *vértices* de acuerdo a sus adyacencias con los *vértices* de la clique. De esta manera se reduce en $|K|n$ el número de variables del modelo y se eliminan las restricciones de asignación y adyacencia correspondientes a los *vértices* de K. Además las variables x_{ij} para $[ij] \in E$ y $j \in K$ también son eliminadas. Por otro lado, mediante alguna heurística rápida y eficiente puede obtenerse una cota superior de $\chi(G)$. Así, las variables x_{ij} y w_j con j mayor que la cota superior del número cromático pueden ser eliminadas del modelo.

4.2. Modelo de Cubrimiento

Hemos encontrado en la literatura muy pocos modelos que hayan sido propuestos con el fin de mejorar alguna de las observaciones que hemos hecho al modelo anterior. Uno de ellos está basado en considerar al *problema de coloreo en grafos* como un *problema de cubrimiento* de *vértices* por conjuntos. Sabemos que el conjunto de *vértices* coloreados con un mismo color es un conjunto independiente. Entonces, un coloreo no es más que una partición de V por medio de conjuntos independientes. A partir de esto puede obtenerse una formulación del *problema de coloreo de grafos* usando los conjuntos independientes y buscando una partición de mínimo cardinal.

Sea $S = \{S_1, S_2, \ldots, S_t\}$ el conjunto de todos los conjuntos independientes de G. Sea x_{S_i} una variable binaria tal que $x_{S_i} = 1$ si todos los *vértices* de S_i son coloreados con un mismo color y $x_{S_i} = 0$ en caso contrario. Con estas variables el problema de coloreo puede ser formulado como:

$$
\begin{aligned}
&\textit{Minimizar} \quad \sum_{i=1}^{t} x_{S_i} \\
&\textit{sujeto a} \\
&\qquad \sum_{i:v \in S_i} x_{S_i} = 1 \qquad \forall\ v \in V \qquad (4.3) \\
&\qquad x_{S_i} \in \{0,1\} \qquad \forall\, i = 1, \ldots, t
\end{aligned}
$$

Las restricciones (4.3) imponen que exista un conjunto independiente (y sólo uno) al que pertenezca cada uno de los *vértices* del *grafo*. Esta formulación tiene n restricciones y un número muy grande de variables. Una forma de reducir el número de variables es considerar conjuntos independientes maximales. En este caso, no puede imponerse que un *vértice* pertenezca a un único conjunto independiente maximal pues el problema podría resultar no factible. Pero a los fines de colorear, basta buscar un cubrimiento en lugar de una partición. Para los *vértices* pertenecientes a

más de un conjunto independiente que forme parte de la solución, puede elegirse un color entre los posibles.

Si $M = \{M_1, M_2, \ldots, M_p\}$ es el conjunto de todos los conjuntos independientes maximales, la formulación es:

$$\begin{aligned} &\textit{Minimizar} \quad \sum_{i=1}^{p} x_{M_i} \\ &\textit{sujeto a} \\ &\qquad \sum_{i:v\in S_i} x_{M_i} \geq 1 \qquad \forall\ v \in V \\ &\qquad\quad x_{M_i} \in \{0,1\} \qquad \forall\, i = 1,\ldots,p \end{aligned}$$

Aún con esta reducción, el número de variables puede ser muy grande. No es necesario que el *grafo* tenga un gran número de *vértices* para que sea imposible contar con la formulación explícita del modelo. En [61], Mehrotra y Trick proponen un algoritmo *Branch-and-Price* para resolver el problema. Los autores reportan buenos resultados para *grafos* generados al azar pequeños a medianos (no más de 70 *vértices*) y con mayor dificultad para densidades medias. También experimentan con *grafos* de la literatura donde los resultados son más satisfactorios. De acuerdo a su experiencia, el valor de la relajación lineal es una buena cota inferior del número cromático.

Este modelo elimina un cierto tipo de simetría. Al no considerar los colores en la formulación, los coloreos equivalentes resultantes de la permutación de colores no están presentes en la formulación. Hay una simetría que sí se conserva. En una solución factible, las variables x_{S_i} con valor 1 determinan una partición de conjunto de *vértices*. El cardinal de dicha partición es la suma de estas variables. Dependiendo de la estructura del *grafo*, pueden existir diversas maneras de obtener una partición en conjuntos independientes con el mismo cardinal. Todas ellas tienen asociadas soluciones factibles del modelo que resultan ser simétricas.

4.3. Nuevos Modelos que Rompen Simetría

La simetría de las soluciones responde básicamente a la indistinguibilidad de los colores. La existencia de particiones en conjuntos independientes de igual cardinal también representa soluciones simétricas pero influye en menor medida.

A continuación, proponemos tres modelos que eliminan parcialmente soluciones simétricas.

- *Modelo 1*

  Dada $S_1, S_2, \ldots S_k$ una partición en conjuntos independientes de G, cualquier asignación de k colores elegidos de los n posibles, tiene asociada una solución factible de $\mathcal{SCP}$. Si consideramos en orden los colores de 1 a n y exigimos que el color $j+1$ no puede ser usado si los colores $1, 2, \ldots, j$ no fueron usados en algún *vértice*, habremos eliminado un gran número de coloreos simétricos. Si buscamos que estos coloreos no estén presentes en la formulación de programación lineal, basta con agregar a las restricciones del modelo clásico las desigualdades

$$w_j \leq \sum_{i \in V} x_{ij} \qquad \forall\, j = 1, \ldots, n \tag{4.4}$$

$$w_j \geq w_{j+1} \qquad \forall\, j = 1, \ldots, n-1 \tag{4.5}$$

Llamamos $\mathcal{CP}$ a la cápsula convexa de las soluciones factibles enteras de este modelo.

$$\mathcal{CP} = \mathcal{SCP} \cap \{(x, w) \in \mathbb{R}^{n^2+n} : (x, w) \text{ verifica 4.4 y 4.5}\}.$$

Resulta significativa la reducción de soluciones factibles. Por ejemplo, si $G = C_{2k+1}$ (un agujero de $2k+1$ *vértices*), en $\mathcal{CP}$ hay $6(2k+1)$ soluciones óptimas, mientras que en $\mathcal{SCP}$ hay $k(2k-1)(2k+1)^2/3$.

Un modelo similar a éste fue propuesto por Baybars [7] para minimizar la máxima frecuencia asignada en un problema de asignación de frecuencias. Sin embargo, no hemos encontrado en la literatura ninguna experiencia de un algoritmo *Branch-and-Cut* con el modelo ni un estudio poliedral del mismo.

La eliminación de variables mediante la coloración de una *clique* y del conocimiento de una cota superior de $\chi(G)$ puede hacerse como ya indicamos en el modelo clásico.

- *Modelo 2*

  En el Modelo 1, para una partición de cardinal k en conjuntos independientes, aún permanecen todas las soluciones factibles resultantes de permutar los k primeros colores. Con el fin de eliminar algunas, podemos exigir que el número de *vértices* coloreados con color j sea mayor o igual a los coloreados con color $j+1$. De esta manera, si en la partición intervienen conjuntos de distinto cardinal, habremos eliminado algunas de las permutaciones.

  Esto se logra agregando las siguientes restricciones al modelo clásico:

$$w_j \leq \sum_{i \in V} x_{ij} \qquad \forall\, j = 1, \ldots, n \tag{4.6}$$

$$\sum_{i=1}^{n} x_{ij} \geq \sum_{i=1}^{n} x_{ij+1} \qquad \forall\, j = 1, \ldots, n-1 \tag{4.7}$$

  Llamamos $\mathcal{CP}'$ a la cápsula convexa de las soluciones factibles enteras de este modelo.

$$\mathcal{CP}' = \mathcal{SCP} \cap \{(x, w) \in \mathbb{R}^{n^2+n} : (x, w) \text{ verifica (4.6) y (4.7)}\}.$$

  El poliedro $\mathcal{CP}'$ esta incluído en $\mathcal{CP}$.

  Dada una partición en conjuntos independientes, a mayor cantidad de conjuntos con distinto cardinal, mayor la eliminación de soluciones simétricas. En el caso sencillo del ejemplo anterior, $G = C_{2k+1}$, en $\mathcal{CP}'$ hay $1/3$ menos de soluciones óptimas que en $\mathcal{CP}$.

La disminución en la cantidad de soluciones simétricas está basada en la distinción entre los colores y no tiene en cuenta la simetría resultante de las particiones equivalentes. Sin embargo, cuando intentamos realizar un estudio poliedral de $\mathcal{CP}'$, encontramos que ciertas propiedades del *grafo* influyen en las características del poliedro.

Al tratar de caracterizar la dimensión del poliedro, observamos que ésta depende de la estructura de los coloreos. Por ejemplo, si en cualquier coloreo óptimo del *grafo* a lo sumo 2 *vértices* utilizan un mismo color, entonces toda solución factible entera de $\mathcal{CP}'$ satisface $\sum_{i=1}^{n} x_{i2} = 2 - w_{n-2}$. Sin embargo, esta igualdad no resulta válida si existe un conjunto independiente de cardinal mayor a 2 en un coloreo con $\chi(G)$ colores. Esto dificulta determinar el conjunto minimal de ecuaciones asociado con el poliedro $\mathcal{CP}'$ y su dimensión.

Desde el punto de vista algorítmico también tuvimos problemas con este modelo. La reducción del tamaño del modelo presenta alguna dificultad. Por la forma en que se eliminan soluciones simétricas, no puede fijarse la coloración de una *clique*. Dependiendo de la estructura del *grafo* puede resultar no factible que el conjunto de *vértices* coloreados con el color i sea el $i-esimo$ de mayor cardinal. Una forma de evitar esta situación es restringir las desigualdades (4.7) a índices mayores al tamaño de la clique.

- *Modelo 3*

El modelo anterior no distingue los colores que le pueden ser asignados a conjuntos independientes con igual cardinal. Es decir, cada partición puede tener asociada más de una solución factible en $\mathcal{CP}'$. Para evitar esto, en lugar de ordenarlos por cardinal, ordenamos a los conjuntos independientes considerando el menor índice de los *vértices* pertenecientes a cada uno de ellos. Este orden es único y permite eliminar en forma total las permutaciones de los colores, considerando para cada partición una única asignación de colores: al conjunto independiente en lugar j se le asigna el color j. De esta manera, a un *vértice* i nunca se le asigna un color cuyo índice sea superior a i. Para modelar estos coloreos se eliminan del modelo clásico las variables x_{ij} con $j > i$ y se agregan las restricciones:

$$x_{ij} \leq \sum_{k=j-1}^{i-1} x_{kj-1} \quad \forall i \in V\backslash\{1\} \ \forall j = 2, \ldots, i-1 \tag{4.8}$$

Estas restricciones aseguran que el *vértice* i puede usar un color j, con $j < i$ únicamente si algún vértice de índice menor usa el color $j-1$. Si no hay ningún *vértice* en estas condiciones, quiere decir que el *vértice* i está obligado a usar un color que a lo sumo tenga índice $j-1$.

Llamamos $\mathcal{CP}$" a la cápsula convexa de las soluciones factibles enteras de este modelo.

$$\mathcal{CP}" = \mathcal{SCP} \cap \{(x,w) \in \mathbb{R}^{n^2+n} : (x,w) \text{ verifica (4.8) y } x_{ij} = 0 \text{ para } j > i\}.$$

El poliedro $\mathcal{CP}$" esta incluído en $\mathcal{CP}$ y la reducción en el número de soluciones simétricas es significativa. Siguiendo con el ejemplo sencillo de $G = C_{2k+1}$, en $\mathcal{CP}$" hay $2k+1$ soluciones óptimas.

El número de variables es $n(n+1)/2$, inferior a la cantidad de variables de los otros modelos. El número de restricciones es mayor. Si bien se redujo un poco el número de desigualdes entre *vértices* adyacentes, las desigualdades (4.8) son $(n^2-n)/2$. La resolución de las relajaciones lineales insume bastante tiempo debido a esta cantidad de restricciones. De la misma manera que en modelos anteriores, se pueden eliminar variables fijando el coloreo de una *clique*, siempre que se enumeren los *vértices* del *grafo* de manera tal que los primeros *vértices* pertenezcan a la *clique*. Un cota superior de $\chi(G)$ se puede utilizar para eliminar las variables con índice j mayor o igual a dicha cota.

El modelo tiene la particularidad que las propiedades del poliedro dependen de la manera en que se enumeren los *vértices* del *grafo*. A distintas enumeraciones corresponden distintas formulaciones que cambian completamente al poliedro de soluciones factibles. Por ejemplo, si los dos primeros *vértices* del *grafo* son adyacentes, las soluciones del poliedro satisfacen $x_{22} = 1$. Si cambiamos la enumeración de los *vértices* para que no sean adyacentes esta condición no es satisfecha por todo coloreo. En el caso que el primer *vértice* sea universal, las variables x_{i1} se anulan para todo *vértice* i con $i = 2, \ldots, n$.

Para caracterizar la estructura facial del poliedro $\mathcal{CP}$", este comportamiento hace difícil o imposible su estudio.

Los nuevos modelos que proponemos cumplen con el objetivo de eliminar soluciones simétricas. Es natural la preferencia por aquel modelo donde la eliminación sea mayor. Sin embargo, cuando para un mismo problema existen diferentes alternativas, no hay un criterio determinante para elegir una formulación sobre otra. Hay factores que pueden ayudar a tomar una decisión. Por ejemplo: cantidad de variables, cantidad de restricciones, calidad de la relajación lineal, performance de los algoritmos, entre otros. En nuestro caso, los dos últimos modelos presentan dificultades para caracterizar la cápsula convexa de la soluciones factibles. El poliedro $\mathcal{CP}$ es más independiente de las particularidades del *grafo* y se puede realizar un estudio de su estructura facial. Como $\mathcal{CP}" \subset \mathcal{CP}$ y $\mathcal{CP}' \subset \mathcal{CP}$, las desigualdades válidas obtenidas para $\mathcal{CP}$ pueden ser usadas como planos de corte en las relajaciones lineales de $\mathcal{CP}'$ y $\mathcal{CP}"$. La experimentación con un algoritmo *Branch-and-Cut* utilizando los Modelos 2 y 3 también fundamenta nuestra elección de trabajar con $\mathcal{CP}$. Algunas dificultades ya las señalamos y otras son analizadas en el Capítulo 7.

En el próximo Capítulo presentamos nuestro estudio poliedral de $\mathcal{CP}$.

Capítulo 5

Estudio Poliedral del *Problema de Coloreo de Grafos*

La caracterización total o parcial de un poliedro asociado a un modelo de programación lineal entera, tiene como objetivo identificar desigualdades válidas para ser usadas en un algoritmo *Branch-and-Cut*. La efectividad de una desigualdad en este contexto está relacionada en gran medida con la dimensión de la cara que defina la misma. De ahí la importancia de caracterizar caras con dimensión alta, en particular facetas del poliedro. Con este fin, en este Capítulo presentamos un detallado estudio poliedral de $\mathcal{CP}$, el poliedro asociado a uno de los modelos de programación lineal entera que propusimos en el Capítulo anterior para el *problema de coloreo de grafos*. Resultados parciales de este estudio están publicados en [63]. Asumimos que el lector tiene los conocimientos básicos de la teoría poliedral.

5.1. El Poliedro $\mathcal{CP}$

Sean w_j y x_{ij} variables binarias $\forall i \in V$ y $j = 1, \ldots, n$ tal que:

$$x_{ij} = \begin{cases} 0 & \text{si el } \textit{vértice } i \text{ no es coloreado con el color } j \\ 1 & \text{si el } \textit{vértice } i \text{ es coloreado con el color } j \end{cases}$$

$$w_j = \begin{cases} 0 & \text{si el color } j \text{ no es usado por ningún } \textit{vértice} \\ 1 & \text{si el color } j \text{ es usado por algún } \textit{vértice} \end{cases}$$

Como ya definimos en el Capítulo 4, el poliedro $\mathcal{CP}$ está asociado con la siguiente formulación de programación lineal entera:

$$\textit{Minimizar} \quad \sum_{j=1}^{n} w_j$$

sujeto a

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

$$x_{ij} + x_{kj} \leq w_j \qquad \text{si } [i,k] \in E\ \forall\ j = 1,\ldots,n \tag{5.2}$$

$$w_j \leq \sum_{i \in V} x_{ij} \qquad \forall\ j = 1,\ldots,n \tag{5.3}$$

$$w_j \geq w_{j+1} \qquad \forall\ j = 1,\ldots,n-1 \tag{5.4}$$

$$x_{ij}, w_j \in \{0,1\} \qquad \forall\ i \in V\ \forall\ j = 1,\ldots,n \tag{5.5}$$

El poliedro $\mathcal{CP}$ es la cápsula convexa de las soluciones factibles del modelo, es decir

$$\mathcal{CP} = conv\{(X,W) \in \{0,1\}^n : (X,W) \text{ satisface las restricciones 5.1 a 5.5}\}$$

Dicho en otros términos, el poliedro $\mathcal{CP}$ está asociado con los coloreos C de G que tienen la propiedad que si usan un color entonces también usan los de índice menor. Para simplificar la notación, usaremos indistintamente C ó (X^C, W^C) para referirnos a las soluciones factibles de $\mathcal{CP}$. Dado un coloreo C de G, $C(v) = j$ significa que el *vértice* v es coloreado con j. Entenderemos que x_{ij}^C vale 1 si el *vértice* i es coloreado con j por el coloreo C y 0 en caso contrario. La variable w_j^C toma valor 1 cuando el color fue usado por algún *vértice* por el coloreo C y 0 en caso contrario.

La teoría poliedral establece que todo poliedro $P \subset \mathbb{R}^n$ puede ser descripto por ecuaciones e inecuaciones lineales. Es decir,

$$P = \{x \in \mathbb{R}^n : A^{=}x = b^{=}, A^{\leq}x \leq b^{\leq}\}$$

El sistema $A^{=}x = b^{=}$ (*sistema minimal de ecuaciones*) determina la dimensión del poliedro, siendo $dim(P) = n - rango(A^{=})$.

Toda desigualdad válida (π, π_0) de un poliedro P, $\pi x \leq \pi_0 \forall x \in P$, define una cara F siendo $F = P \cap \{x \in \mathbb{R}^n : \pi x = \pi_0\}$. Si $dim(F) = dim(P) - 1$, entonces F es una *faceta*. Las únicas desigualdades necesarias y suficientes para la descripción de P son las que definen facetas. Es decir, para caracterizar la cápsula convexa de un conjunto de puntos se necesita y es suficiente encontrar un sistema minimal de ecuaciones y desigualdades válidas que definan facetas.

Para determinar si una desigualdad válida (π, π_0) es faceta hay que:

- exhibir algún punto que satisfaga la desigualdad en forma estricta.
- encontrar $dim(P)$ vectores afínmente independientes en F.

En algunos casos no es fácil obtener los puntos afínmente independientes. Un alternativa para reconocer si una cara es una faceta es usar el siguiente resultado:

Lema: *Sea $A^=x = b^=$ el sistema minimal de ecuaciones del poliedro P y (π, π_0) una cara F de P. (π, π_0) define faceta de P si y sólo si para cualquier (λ, λ_0) desigualdad válida para P tal que $\lambda x = \lambda_0$ para todo $x \in F$ existe un escalar $\alpha > 0$ y un vector u tal que $(\lambda, \lambda_0) = (\alpha\pi + uA^=, \alpha\pi_0 + ub^=)$.*

La metodología para utilizar el Lema es la siguiente. Se asume la existencia de una desigualdad válida (λ, λ_0) tal que $F \subset \{x \in P : \lambda x = \lambda_0\}$. Conociendo las propiedades que tienen los coeficientes de una combinación lineal de π y de las filas de $A^=$, se trata de verificar que son cumplidas por los coeficientes de λ. En general esta deducción se hace considerando puntos en F o propiedades de los mismos que implican las propiedades deseadas. Si $X_1, X_2 \in F$, entonces $\lambda X_1 = \lambda X_2$. Eligiendo convenientemente los valores de las coordenadas de X_1 y X_2, se deducen condiciones sobre los coeficientes de λ. Esto se repite hasta completar la descripción.

En este Capítulo hacemos uso de los dos procedimientos. En algunos casos resulta sencillo generar los puntos afínmente independientes y en otros el análisis se hace utilizando la metodología descripta. Comenzamos por estudiar la dimensión de $\mathcal{CP}$.

5.2. Dimensión de $\mathcal{CP}$

Teorema 5.1 *Un sistema minimal de ecuaciones para $\mathcal{CP}$ es*

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

$$\sum_{i=1}^{n} x_{in} = w_n$$

$$w_j = 1 \quad \forall\, j = 1, \ldots, \chi(G)$$

Entonces, la dimensión de $\mathcal{CP}$ es $n^2 - \chi(G) - 1$.

Demostración:

Toda solución factible satisface las primeras n ecuaciones del sistema ya que se econtraban en la formulación del modelo. Debido a la restricción de orden entre los colores, las soluciones factibles de CP no usan el color n ó usan exactamente una vez cada color. En ambos casos, las soluciones verifican la última ecuación del sistema.

En toda ecuación interviene una variable que no se encuentra en el resto de las ecuaciones, lo que nos permite afirmar que la matriz tiene rango completo. Entonces, $dim(\mathcal{CP}) \leq n^2 - \chi(G) - 1$.

Para determinar la dimensión de $\mathcal{CP}$, construimos $n^2 - \chi(G)$ soluciones afínmente independientes. A continuación detallamos los coloreos factibles que consideramos.

- Sea C^1 un *(n)-coloreo.* Llamamos v_i al *vértice* coloreado con color i para $i = 1, \ldots, n$.

- Sean $j, i \in \{1, \ldots, n-1\}$ con $i \neq j$. A partir de C^1, construimos un *(n)-coloreo* intercambiando los colores de v_i, v_j y v_n, asignando color j a v_n, color i a v_j y color n a v_i. De esta manera obtenemos $(n-1)(n-2)$ coloreos.

- Para $j = 1, \ldots, n-1$, nuevamente basados en C^1, obtenemos los *(n)-coloreo* que asignan color j a v_n, color n a v_j y color i a v_i para todo $i \neq j, n$. Con este procedimiento tenemos $n-1$ coloreos.

- Sin pérdida de generalidad podemos suponer que v_{n-1} y v_n no son adyacentes.
  Sea el *(n-1)-coloreo* que asigna color i a v_i para $i = 1, \ldots, n-2$, y el color $n-1$ a los *vértices* v_{n-1} y v_n.

- Considerando el *(n-1)-coloreo* definido arriba, podemos obtener un *(n-1)-coloreo* intercambiando los colores i y $n-1$. Haciendo esto para $i = 1, \ldots, n-2$, construimos $n-2$ coloreos.

- Finalmente, tomamos un *(j)-coloreo* para cada $j = \chi(G), \ldots, n-2$.

Tenemos $n^2 - \chi(G)$ coloreos. Notamos con C^i los coloreos construidos para $i = 1, \ldots, n^2 - \chi(G)$. Consideremos una combinación lineal afín

$$\sum_{i=1}^{n^2-\chi(G)} \alpha_i (X^{C^i}, W^{C^i}) = 0 \text{ tal que } \sum_{i=1}^{n^2-\chi(G)} \alpha_i = 0.$$

Para verificar la independencia afín de las soluciones tenemos que ver que todos los α_i resultan nulos.

El *(n)-coloreo* inicial es el único que colorea a v_n con color n, es decir la única solución con la variable $x_{v_n n}$ no nula. Por lo tanto $\alpha_1 = 0$.

Para $j = \chi(G), \ldots, n$, las soluciones con la variable w_j no nula son los *(k)-coloreo* para $k = j, \ldots, n$. Sea S_j el conjunto de los *(k)-coloreo* para $k \geq j$. Entonces se debe satisfacer que

$$\sum_{i: C^i \in S_j} \alpha_i = 0 \text{ para todo } j = \chi(G), \ldots, n$$

Pero $S_{j+1} \subset S_j$ y $|S_j| = 1$ para $j = \chi(G), \ldots, n-2$. Entonces $\alpha_i = 0$ para las soluciones asociadas a *(j)-coloreo* con $j = \chi(G), \ldots, n-2$.

Para analizar el resto de las soluciones, construimos una tabla. Las filas corresponden a cada uno de los *(n)-coloreo* y *(n-1)-coloreo* que restan analizar. Las n columnas corresponden a los n colores posibles e indicamos en cada una los *vértices* coloreados por cada uno de ellos.

	color 1	color 2	color 3	....	color i	....	color n-1	color n
Bloque 1	v_n	v_1	v_3	....	v_i	....	v_{n-1}	v_2
	v_n	v_2	v_1	....	v_i	....	v_{n-1}	v_3
	.							
	.							
	v_n	v_2	v_3	....	v_1	....	v_{n-1}	v_i
	.							
	.							
	v_n	v_2	v_3	....	v_i	....	v_1	v_{n-1}
	v_n	v_2	v_3	....	v_i	....	v_{n-1}	v_1
Bloque 2	v_2	v_n	v_3	....	v_i	....	v_{n-1}	v_1
	v_1	v_n	v_2	....	v_i	....	v_{n-1}	v_3
	.							
	.							
	v_1	v_n	v_3	....	v_2	....	v_{n-1}	v_i
	.							
	.							
	v_1	v_n	v_3	....	v_i	....	v_2	v_{n-1}
	v_1	v_n	v_3	....	v_i	....	v_{n-1}	v_2
Bloque i	v_i	v_2	v_3	....	v_n	....	v_{n-1}	v_1
	v_1	v_i	v_3	....	v_n	....	v_{n-1}	v_2
	.							
	.							
	v_1	v_2	v_3	....	v_n	....	v_i	v_{n-1}
	v_1	v_2	v_3	....	v_n	....	v_{n-1}	v_i
Bloque n-1	v_{n-1}	v_2	v_3	....	v_i	....	v_n	v_1
	v_1	v_{n-1}	v_3	....	v_i	....	v_n	v_2
	.							
	.							
	v_1	v_2	v_3	....	v_{n-1}	....	v_n	v_i
	.							
	.							
	v_1	v_2	v_3	....	v_i	....	v_n	v_{n-1}
Bloque n	v_{n-1}, v_n	v_2	v_3	....	v_i	....	v_1	-
	v_1	v_{n-1}, v_n	v_3	....	v_i	....	v_2	-
	.							
	.							
	v_1	v_2	v_3	....	v_{n-1}, v_n	....	v_i	-
	v_1	v_2	v_3	....	v_i	....	v_{n-1}, v_n	-

Si observamos los $n-2$ primeros bloques en que hemos dividido a la tabla, podemos afirmar que los primeros $n-3$ coloreos del bloque j son únicos en la coloración del *vértice* v_j. Claramente los coeficientes que corresponden a estas soluciones resultan nulos.

Para $i = 1, \ldots, n-2$, llamemos δ_i y β_i a los multiplicadores de las dos últimas soluciones de cada bloque . Para $i = 1, \ldots, n-1$, sea γ_i el correspondiente al i-esimo *(n)-coloreo* del bloque $n-1$ de la tabla y μ_i al i-esimo *(n-1)-coloreo* del último bloque.

Eliminado los coloreos para los cuales ya sabemos que los coeficientes son nulos, podemos afirmar que:

1. $\sum_{i=1}^{n-1} \mu_i = 0$ (únicos *(n-1)-coloreo*)
2. $\sum_{i=1}^{n-2} \delta_i + \gamma_{n-1} = 0$ (únicos que colorean a v_{n-1} con n)

3. $\gamma_i + \mu_i = 0 \quad \forall\, i = 1, \ldots, n-2$ (únicos que colorean a v_{n-1} con i)
4. $\delta_i + \mu_i = 0 \quad \forall\, i = 1, \ldots, n-2$ (únicos que colorean a v_i con $n-1$)
5. $\sum_{i=1}^{n-2} \beta_i + \mu_{n-1} = 0$ (únicos que colorean a v_{n-1} con $n-1$)
6. $\delta_i + \beta_i + \mu_i = 0 \quad \forall\, i = 1, \ldots, n-2$ (únicos que colorean a v_n con i)
7. $\beta_i + \gamma_i = 0 \quad \forall\, i = 1, \ldots, n-2$ (únicos que colorean a i con n)

De 4 y 6, se deduce que $\beta_i = 0$ para todo $i = 1, \ldots, n-2$. Pero entonces, de 7 resulta $\gamma_i = 0$ para todo $i = 1, \ldots, n-2$ y de 5, $\mu_{n-1} = 0$. Como consecuencia de 3, tenemos que $\mu_i = 0$ y por 4, $\delta_i = 0$ para todo $i = 1, \ldots, n-2$. Por último, de 2 obtenemos $\gamma_{n-1} = 0$.

Construimos $n^2 - \chi(G)$ soluciones factibles afínmente independentes. Resulta entonces que $dim(\mathcal{CP}) = n^2 - \chi(G) - 1$.

□

5.2.1. Proyección de $\mathcal{CP}$

La reducción de la cantidad de variables y restricciones en un modelo es importante para poder resolver instancias lo más grande posible. Si $\hat{\chi}(G)$ es una cota superior de $\chi(G)$, podemos afirmar que la solución óptima del *problema de coloreo de grafos* se encuentra en

$$Proy(\mathcal{CP}) = \{(X, W) : w_j = 0 \ \forall\, j = \hat{\chi}(G) + 1, \ldots, n\}.$$

Si bien $Proy(\mathcal{CP}) \subset \mathcal{CP}$, el conocimiento de las características propias de $Proy(\mathcal{CP})$ resulta de interés para tener otro criterio de valoración de las desigualdes válidas de $\mathcal{CP}$.

En el caso particular que $\chi(G) = \hat{\chi}(G)$, $Proy(\mathcal{CP})$ tiene propiedades que dependen de las características que tengan las particiones de V en $\chi(G)$ conjuntos independientes. Por ejemplo, si v_1 y v_2 son *vértices* no adyacentes que en cualquier *$(\chi(G))$-coloreo* pertenecen al mismo conjunto independiente, toda solución factible de $Proy(\mathcal{CP})$ satisface $x_{v_1 j} = x_{v_2 j}$ para todo $j = 1, \ldots, \chi(G)$. Hemos analizado diferentes características de las particiones, pero no pudimos determinar el sistema minimal de ecuaciones de $Proy(\mathcal{CP})$ para toda posible configuración de las particiones de V.

En la siguiente proposición determinamos la dimensión de $Proy(\mathcal{CP})$ para el caso que $\chi(G) \leq \hat{\chi}(G) - 1$.

Proposición 5.1 *Si $\chi(G) \leq n - k - 1$, el poliedro $Proy(\mathcal{CP}) = \mathcal{CP} \cap \{w_j = 0$ para todo $j = n - k + 1, \ldots, n\}$ tiene el siguiente sistema minimal de ecuaciones*

$$\begin{aligned}
&\sum_{j=1}^{n} x_{ij} = 1 && \forall i \in V \\
&w_j = 1 && \forall j = 1, \ldots, \chi(G) \\
&w_j = 0 && \forall j = n-k+1, \ldots, n \\
&x_{ij} = 0 && \forall i \in V \qquad \forall j = n-k+1, \ldots, n
\end{aligned}$$

Demostración:

Claramente las ecuaciones son satisfechas por las soluciones factibles de $Proy(\mathcal{CP})$ y el sistema tiene rango máximo. Veamos entonces que si $\lambda^X X + \lambda^W W = \lambda_0$ se satisface para todo $(X, W) \in Proy(\mathcal{CP})$, entonces (λ^X, λ^W) es combinación lineal de las ecuaciones del sistema. Para esto deberemos ver que:

- $\lambda^X_{vj} = \lambda^X_{vj'} \quad \forall\, v \in V$ y $\forall\, j, j' = 1, \ldots, n-k$
- $\lambda^W_j = 0 \quad \forall\, j = \chi(G)+1, \ldots, n-k$

Sean C^1 un *(n-k-1)-coloreo*, $j \leq n-k-1$ y $v_1, v_2 \in V$ tales que $C^1(v_1) = C^1(v_2) = j$. Cambiándole el color a v_1 por $n-k$, obtenemos C^2 un *(n-k)-coloreo* a partir del cual, como $\lambda^{X^{C^1}} + \lambda^{W^{C^1}} = \lambda^{X^{C^2}} + \lambda^{W^{C^2}}$, podemos concluir que $\lambda^X_{v_1 j} = \lambda^X_{v_1 n-k} + \lambda^W_{n-k} \;\forall\, j = 1, \ldots n-k-1$. De aquí se desprende que $\lambda^X_{v_1 j} = \lambda^X_{v_1 j'} \quad \forall\, j, j' = 1, \ldots, n-k-1$. Entonces, la identidad es válida para todos los *vértices* que comparten color.

Sea v tal que $C^1(v) = j'$ y v es único con ese color. Intercambiando el color j' con j y considerando lo demostrado arriba obtenemos $\lambda^X_{vj} = \lambda^X_{vj'} \quad \forall\, j, j' = 1, \ldots, n-k-1$.

Nos falta ver aún el caso $j = n-k$. Como $\chi(G) \leq n-k-1$, existe C^1 un *(n-k)-coloreo* tal que v_2 es el único coloreado con color $n-k$. Por otro lado, también existe C^2 un *(n-k)-coloreo* tal que v_1 y v_2 son los únicos que comparten el color $n-k$. Resulta entonces que

$$\lambda^X_{v_2 n-k} + \lambda^X_{v_1 c_{v_1}} + \sum_{v \neq v_1, v_2} \lambda^X_{v c_v} = \lambda^X_{v_2 n-k} + \lambda^X_{v_1 n-k} + \sum_{v \neq v_1, v_2} \lambda^X_{v c'_v}$$

donde $C^1(v_1) = c_{v_1}$, $C^1(v) = c_v$ y $C^2(v) = c'_v$ con $c_v, c'_v \leq n-k-1$.

Como ya sabemos que $\lambda^X_{v c_v} = \lambda^X_{v c'_v}$, obtenemos $\lambda^X_{v_1 c_{v_1}} = \lambda^X_{v_1 n-k}$ y también que $\lambda^W_{n-k} = 0$. La identidad es válida para cualquier *vértice* que comparta color con algún otro. Con el mismo procedimiento utilizado antes se puede deducir que es válida para todo $v \in V$.

Por último, construimos un *(j-1)-coloreo* con $\chi(G)+1 \leq j \leq n-k$, de manera tal que v, v' comparten el color $j-1$. Si el color de v es cambiado a j obtenemos un *(j)-coloreo* y la identidad $\lambda^X_{vj-1} = \lambda^X_{vj} + \lambda^W_j$. Por la propiedad demostrada antes resulta $\lambda^W_j = 0$.

□

5.3. Propiedades de las Restricciones del Modelo

Veamos ahora cuales de las inecuaciones del modelo definen facetas de $\mathcal{CP}$. Comenzamos por la restricciones de positividad de las variables.

Proposición 5.2 *Sea $v \in V$,*

1. $x_{vn} \geq 0$ *es faceta de* $\mathcal{CP}$.
2. $x_{vj_0} \geq 0$ *es faceta de* $\mathcal{CP}$ *para todo* $j_0 = 1, \ldots n-1$ *si y sólo si* $V \backslash \{v\}$ *no es una* clique.

Demostración:

1. Consideramos un *(n)-coloreo* que asigne el color n al *vértice* v y generamos $n^2 - \chi(G)$ soluciones factibles con el procedimiento usado en la demostración del teorema 5.1 . Si excluimos el *(n)-coloreo* generador de las soluciones, el resto verifica que $x_{vn} = 0$ y son afínmente independientes.

2. Si $V\backslash\{v\}$ es *clique*, entonces $\chi(G) = n - 1$. En este caso, toda solución factible que verifica $x_{vj_0} = 0$, satisface $\sum_{i \in V\backslash\{v\}} x_{ij_0} = 1$ y por lo tanto la desigualdad no es faceta. Supongamos entonces que hay dos *vértices* $z, z' \in V\backslash\{v\}$ que no son adyacentes. Consideremos un *(n)-coloreo* que asigne color $n-1$ a z, n a z' y al *vértice* v un color distinto a j_0. Nos generamos soluciones factibles como antes y excluimos el *(n)-coloreo* que asigna j_0 a v. Obtenemos de este modo los coloreos necesarios para poder afirmar que $x_{vj_0} \geq 0$ es faceta.

□

Como $\mathcal{CP}$ está contenido en el espacio afín definido por el sistema de ecuaciones (5.1), las desigualdades $x_{vj} \leq 1$ son implicadas por la positividad de las variables.

Siguiendo con las desigualdades del modelo, las restricciones de orden establecidas entre los colores resultan ser facetas. La siguiente proposición así lo demuestra.

Proposición 5.3 *La restricción* $w_{j_0} \geq w_{j_0+1}$ *es faceta de* $\mathcal{CP}$ *para todo* j_0 *tal que* $\chi(G) \leq j_0 \leq n-2$.

Demostración:

Consideremos los coloreos que usamos en el Teorema 5.1 y excluyamos el (j_0)-coloreo. Claramente todos verifican $w_{j_0} = w_{j_0+1}$ y por lo tanto la desigualdad es una faceta de $\mathcal{CP}$.

□

Para finalizar con las facetas definidas por la restricciones del modelo, veamos el siguiente resultado.

Proposición 5.4 *La desigualdad* $w_{j_0} \leq \sum_{v \in V} x_{vj_0}$ *es faceta de* $\mathcal{CP}$ *para todo* $j_0 = 1, \ldots, n-1$ *si y sólo si existe algún* $(\chi(G))$-coloreo *que la verifica por igualdad.*

Demostración:

Si ningún *$(\chi(G))$-coloreo* verifica la igualdad, entonces toda solución factible en la cara definida por la desigualdad satisface $w_{\chi(G)+1} = 1$ y esto contradice la condición de faceta.

Supongamos ahora que existe un *$(\chi(G))$-coloreo* que verifica la igualdad. Consideremos los *(n)-coloreo* y *(n-1)-coloreo* del Teorema 5.1, excluyamos el *(n-1)-coloreo* que asigna color j_0 a v_{n-1} y v_n y agreguemos para cada $j = \chi(G), \ldots, n-2$, un *(j)-coloreo* donde el color j_0 sea usado por un único *vértice*. Estas soluciones son afínmente independientes y nos permiten afirmar que la desigualdad es faceta.

□

La siguiente desigualdad que analizamos no pertenece a la formulación original. Sin embargo, en la implementación de nuestro algoritmo *Branch-and-Cut* fue muy útil considerar una relajación inicial donde reemplazamos las desigualdades (5.2), $x_ij + x_kj \leq w_j$ para todo $j = 1, \ldots, n$ y $[i,k] \in E$, por las desigualdades que presentamos a continuación. La cantidad de restricciones (5.2) del modelo es nm y para grafos de media y alta densidad, la resolución de la relajación lineal insume mucho tiempo. El reemplazo de las restricciones otorga mejor perfomance al algoritmo.

Consideremos $v \in V$ y $v_1, v_2, \ldots, v_r$ un conjunto independiente máximo de $N(v)$. Si $r = 1$, tenemos una *clique* maximal que es analizada más adelante. Supongamos entonces que $r \geq 2$. Para cualquier coloreo podemos hacer las siguientes afirmaciones:

- si v no es coloreado con el coloro j_0, a lo sumo r *vértices* de $N(v)$ pueden usar el color j_0.
- si v usa el color $n - r + j + 1$ para algún $j \leq r - 1$, ningún color se usa en más de $r - j$ *vértices* (en particular el j_0).
- si el *vértice* v es coloreado con el color j_0, todo *vértice* adyacente a v no puede usar este color.

A partir de estas observaciones surge la desigualdad ***Vecindad***

$$\sum_{u \in N(v)} x_{uj_0} + r x_{vj_0} + \sum_{j=1}^{r-1} j x_{vn-r+j+1} \leq r w_{j_0}$$

que resulta válida para todo $j_0 \leq n - r + 1$.

Cualquier *(n)-coloreo* que asigne color j_0 a un *vértice* adyacente a v y color $n - 1$ a v, satisface estrictamente la desigualdad. Todo coloreo que asigne el color j_0 a v la satisface por igualdad. Por lo tanto, la desigualdad no es redundante del sistema minimal de ecuaciones que define a $\mathcal{CP}$ y resulta ser una cara.

La desigualdad no es válida para $\mathcal{RCP}$ (relajación lineal de CP) pues $x_{ij} = \frac{1}{n}$ y $w_j = \frac{2}{n}$ para todo $i, j = 1, \ldots, n$ no la satisface. La nueva relajación no es más débil que $\mathcal{RCP}$.

En la siguiente proposición caracterizamos cuando la desigualdad resulta ser faceta.

Proposición 5.5 *Sea j_0 tal que $j_0 \leq n - r + 1$. La desigualdad **Vecindad***

$$\sum_{u \in N(v)} x_{uj_0} + r x_{vj_0} + \sum_{j=1}^{r-1} j x_{vn-r+j+1} \leq r w_{j_0}$$

es válida de $\mathcal{CP}$ y define una faceta.

Demostración:
Por lo mostrado más arriba, ya sabemos que la desigualdad es válida. Sea F la cara de $\mathcal{CP}$ definida por la desigualdad. Supongamos que $\lambda^X X + \lambda^W W \leq \lambda_0$ es una desigualdad válida para $\mathcal{CP}$ tal que $F \subseteq \mathcal{CP} \cap \{(X, W) : \lambda^X X + \lambda^W W = \lambda_0\}$.

Para afirmar la condición de faceta debemos ver que:

1. $\lambda^X_{zj} = \lambda^X_{zn} + \lambda^W_n \quad \forall\, j = 1, \ldots, n-1,\ j \neq j_0, \quad \forall\, z \neq v$
2. $\lambda^X_{zj_0} - \lambda^X_{zj} = \gamma \quad \forall\, j = 1, \ldots, n-1 \quad \forall\, z \in N(v)$
3. $\lambda^X_{vj_0} = \lambda^X_{vn} + \lambda^W_n + \gamma$
4. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n - (r-1)\gamma \quad \forall\, j = 1, \ldots, n-r+1,\ j \neq j_0$
5. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n + (j-n)\gamma \quad \forall\, j = n-r+2, \ldots, n-1$
6. $\lambda^X_{zj_0} = \lambda^X_{zn} + \lambda^W_n \quad \forall\, z \notin N(v),\ z \neq v$
7. $\lambda^W_j = 0 \quad \forall\, j = \chi(G)+1, \ldots, n-1,\ j \neq j_0$
8. $\lambda^W_{j_0} = -r\gamma$ si $j_0 \geq \chi(G)+1$
9. $\gamma \geq 0$

Probamos cada una de las condiciones.

1. Como $r \geq 2$, existen $v_k, v_i \in N(v)$ *vértices* no adyacentes. Podemos construir los siguientes coloreos en F:

$$\begin{array}{ll} C^1 : (n-1) - coloreo & C^2 : (n) - coloreo \\ C^1(v_k) = C^1(v_i) = j & C^2(v_k) = n \\ C^1(v) = j_0 & C^1(z) = C^2(z) \ \ \forall\, z \neq v_k \\ C^1(z) \in \{1, \ldots, n-1\} \backslash \{j, j_0\} & \\ \forall z \neq v_k, v_i, v & \end{array}$$

   Como $\lambda^{XC^1} + \lambda^{WC^1} = \lambda^{XC^2} + \lambda^{WC^2}$ y los coloreos difieren sólo en la coloración de v_k, es trivial concluir que $\lambda^X_{v_kj} = \lambda^X_{v_kn} + \lambda^W_n$.

   Sea $z \in V$, $z \neq v, v_k$ y C^3 el *(n)-coloreo* tal que $C^3(z) = C^2(v_k)$, $C^3(v_k) = C^2(z)$ y $C^3(u) = C^2(u)$ para todo $u \neq v, v_k$. Como $C^3 \in F$ y difiere de C^2 sólo en la coloración de z y v_k, entonces $\lambda^X_{zj} + \lambda^X_{v_kn} = \lambda^X_{zn} + \lambda^X_{v_kj}$. A partir de lo deducido antes, concluimos que $\lambda^X_{zj} = \lambda^X_{zn} + \lambda^W_n$ para todo $z \neq v$.

2. Sean $u, u' \in N(v)$ y $j = 1, \ldots, n-1$ con $j \neq j_0$. Sean los siguientes coloreos de G en F:

$$\begin{array}{ll} C^1 : (n) - coloreo & C^2 : (n) - coloreo \\ C^1(u) = j & C^2(u) = j_0 \\ C^1(u') = j_0 & C^2(u') = j \\ C^1(v) = n & C^2(z) = C^1(z) \ \ \forall\, z \neq u, u' \\ C^1(z) \in \{1, \ldots, n\} \backslash \{n, j, j_0\} & \\ \forall z \neq v, u, u' & \end{array}$$

Como consecuencia del intercambio de colores y del caso anterior, obtenemos $\lambda_{uj_0} - \lambda_{uj} = \lambda_{u'j_0} - \lambda_{u'j} = \gamma$.

3. Sea $u \in N(v)$. Consideremos los siguientes coloreos de G:

$C^1 : (n) - coloreo$	$C^2 : (n) - coloreo$
$C^1(v) = j_0$	$C^2(v) = n$
$C^1(u) = n$	$C^2(u) = j_0$
$C^1(z) \in \{1, \ldots, n\} \backslash \{n, j_0\}$	$C^2(z) = C^1(z) \ \forall\, z \neq v, v_1$
$\forall z \neq v, v_1$	

Claramente surge la identidad $\lambda^X_{vj_0} + \lambda^X_{un} = \lambda^X_{vn} + \lambda^X_{uj_0}$. Usando los dos casos anteriores, tenemos que $\lambda^X_{vj_0} = \lambda^X_{vn} + \lambda^W_n + \gamma$.

4. Sea $j \leq n - r + 1$ y $j \neq j_0$. Podemos construir los siguientes coloreos de G en F:

$C^1 : (n - r + 1) - coloreo$	$C^2 : (n - r + 1) - coloreo$
$C^1(v) = j_0$	$C^2(v) = j$
$C^1(v_i) = j \ \forall\, i = 1, \ldots, r$	$C^2(v_i) = j_0 \ \forall\, i = 1, \ldots, r$
$C^1(z) \in \{1, \ldots, n - r + 1\} \backslash \{j, j_0\}$	$C^2(z) = C^1(z) \ \forall\, z \neq v, v_1, \ldots, v_r$
$\forall z \neq v, v_1, \ldots, v_r$	

Por las coincidencias de ambos coloreos, deducimos

$$\lambda^X_{vj_0} + \sum_{i=1}^{r} \lambda^X_{v_i j} = \lambda^X_{vj} + \sum_{i=1}^{r} \lambda^X_{v_i j_0}$$

Usando los resultados de 2 y 3, resulta $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n + (1 - r)\gamma$.

5. Sea j tal que $n - r + 2 \leq j \leq n - 1$. Construimos los siguientes coloreos en F:

$C^1 : (j) - coloreo$	$C^2 : (j) - coloreo$
$C^1(v) = j$	$C^2(v) = j_0$
$C^1(v_i) = j_0 \ \forall\, i = 1, \ldots, n - j + 1$	$C^2(v_i) = j \ \forall\, i = 1, \ldots, n - j + 1$
$C^1(z) \in \{1, \ldots, j\} \backslash \{j, j_0\}$	$C^2(z) = C^1(z) \ \forall\, z \neq v, v_1, \ldots, v_{n-j+1}$
$\forall z \neq v, v_1, \ldots, v_{n-j+1}$	

Por las coincidencias de ambos coloreos, deducimos

$$\lambda^X_{vj} + \sum_{i=1}^{n-j+1} \lambda^X_{v_i j_0} = \lambda^X_{vj_0} + \sum_{i=1}^{n-j+1} \lambda^X_{v_i j}$$

de donde resulta $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n + (j - n)\gamma$.

6. Sea $z \notin N(v)$. Podemos construir un *(n-1)-coloreo* que colorea a v y a z con j_0 y luego un *(n)-coloreo* que asigne el color n a z y derivar trivialmente que $\lambda^X_{zj_0} = \lambda^X_{zn} + \lambda^W_n$.

7. Sea $j = \chi(G) + 1, \ldots, n-1$, con $j \neq j_0$. Sabemos que para cualquier *(j-1)-coloreo* existen u, u' *vértices* coloreados con el mismo color. Por otro lado, si $j_0 \leq j - 1$, podemos suponer que v está coloreado con j_0. Si ahora cambiamos el color del *vértice* u a j, tenemos un *(j)-coloreo*. Por las coincidencias de los coloreos y las propiedades derivadas en casos anteriores, obtenemos $\lambda^W_j = 0$

8. Supongamos que $j_0 \geq \chi(G) + 1$. Por las mismas razones que esgrimimos en 7, G puede ser coloreado con un *$(j_0 - 1)$-coloreo* para el que podemos asegurar que existen u, u' coloreados con el mismo color. Basados en este coloreo, construímos un *(j_0)-coloreo* tal que v es coloreado con j_0. Además, en el caso que v fuera único en su color en el *$(j_0 - 1)$-coloreo*, el *vértice* u cambia su color por el color de v. Por casos anteriores, concluimos que $\lambda^W_{j_0} = -r\gamma$.

9. Sea C^1 un *(n)-coloreo* en F tal que $C^1(v) = j_0$, $C^1(v') = n-1$ para $v' \in N(v)$ y $C^1(u) = j_u$ con $j_u \neq j_0, n-1$. Se verifica que

$$\lambda_0 = \sum_{u \neq v,v'} \lambda^X_{uj_u} + \lambda^X_{v'n-1} + \lambda^X_{vj_0} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_n$$

Pero como (λ, λ_0) es una desigualdad válida de $\mathcal{CP}$, entonces el *(n)-coloreo* que intercambia los colores de v' y v satisface la desigualdad, es decir

$$\lambda_0 \geq \sum_{u \neq v,v'} \lambda^X_{uj_u} + \lambda^X_{v'j_0} + \lambda^X_{vn-1} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_n$$

Usando la expresión de λ_0 se deduce que

$$\lambda^X_{v'j_0} + \lambda^X_{vn-1} - \lambda^X_{v'n-1} - \lambda^X_{vj_0} \leq 0$$

Pero por la relación derivada entre los coeficientes sabemos que $\lambda^X_{v'j_0} - \lambda^X_{v'n-1} = \gamma$ y $\lambda^X_{vn-1} - \lambda^X_{vj_0} = -2\gamma$. Entonces $\gamma \geq 0$

Concluimos que (λ^X, λ^W) es combinación lineal de las ecuaciones del modelo y de la desigualdad ***Vecindad***. Por lo tanto la desigualdad ***Vecindad*** define una faceta de $\mathcal{CP}$.

□

5.4. Desigualdades Válidas por Eliminación de Simetría

Existen desigualdades válidas que surgen naturalmente por la manera en que eliminamos soluciones simétricas. La primera que presentamos está basada en una simple observación: si el color j_0 no es usado por una solución factible, entonces ningún *vértice* es coloreado con un color con índice mayor a j_0 en esta solución. Es decir,

$$\sum_{j=j_0}^{n} x_{vj} \leq w_{j_0}$$

es una desigualdad válida para $\mathcal{CP}$.

Si $j_0 \leq n-1$, esta desigualdad no es válida para $\mathcal{RCP}$. Para $j_0 \leq n-2$, la solución $x_{ij} = \frac{1}{n}$ para todo $i,j = 1,\ldots,n$ y $w_j = \frac{2}{n}$ para $j = 1,\ldots,n$ no la satisface. Si $j_0 = n-1$, la solución $x_{vn-1} = x_{vn} = \frac{1}{2}$, $w_{n-1} = w_n = \frac{1}{2}$, $x_{ui} = \frac{1}{n-2}$ para todo $u \neq v$, $i \leq n-2$ y $w_i = \frac{2}{n-2}$ con $i \leq n-2$ no la satisface.

Si $j_0 \geq 2$, cualquier *(n)-coloreo* que asigne color 1 a v verifica estrictamente la desigualdad, lo que implica que no es una restricción redundante del sistema de ecuaciones que define a $\mathcal{CP}$. Además, cualquier *(n)-coloreo* que coloree a v con j_0 la verifica por igualdad. Entonces, la desigualdad es cara de $\mathcal{CP}$.

Bajo algunas condiciones, la desigualdad resulta ser faceta de $\mathcal{CP}$. Veamos este resultado en la siguiente proposición.

Proposición 5.6

1. *Sean $v \in V$ y j_0 tal que $1 \leq j_0 \leq n-1$. La desigualdad Anula Color*

$$\sum_{j=j_0}^{n} x_{vj} \leq w_{j_0}$$

es válida para $\mathcal{CP}$. Si $\chi(G)+1 \leq j_0 \leq n-2$, entonces es faceta.

2. *Si v no es un* vértice *universal y $\chi(G) \leq n-2$, entonces $x_{vn-1} + x_{vn} \leq w_{n-1}$ define una faceta de $\mathcal{CP}$.*

Demostración:
La validez de la desigualdad ya fue mostrada.

1. Sea $\chi(G)+1 \leq j_0 \leq n-2$. Vamos a construir $n^2 - \chi(G) - 1$ soluciones factibles afínmente independientes que pertenezcan a esta cara de $\mathcal{CP}$. Consideremos cualquier *(n)-coloreo* que coloree a v con j_0 y asigne los colores $n-1$ y n a *vértices* no adyacentes. Llamamos v_i al *vértice* coloreado con color i para $i = 1,\ldots,n$. Por lo tanto, $v = v_{j_0}$. Basados en este coloreo, construimos los siguientes coloreos:

   - Sean j,i tales que $1 \leq i,j \leq n-1$, $i \neq j, j_0$ y $j \neq j_0$. Coloreamos con un *(n)-coloreo* que asigna color j a v_n, color i a v_j y color n a v_i. De esta manera obtenemos $(n-2)(n-2)$ coloreos.
   - Para $j_0+1 \leq i \leq n-1$, definimos el *(n)-coloreo* que asigna color j_0 a v_n, color i a v_{j_0} y color n a v_i. Tenemos $(n-1-j_0)$ coloreos.
   - Para $j = 1,\ldots,n-1$, consideramos el *(n)-coloreo* que asigna color j a v_n, color n a v_j y color i a v_i $\forall\, i \neq j$, $i \neq n$. Hay $n-1$ coloreos.

   Sea el *(n-1)-coloreo* que asigna color i a v_i $\forall\, i = 1,\ldots,n-2$ y color $n-1$ a v_{n-1} y v_n. Para $i = 1,\ldots,n-2$, construimos otros *(n-1)-coloreo* intercambiando el color i con el color $n-1$. Con este procedimiento tenemos $n-2$ coloreos.

   Consideremos un *(j_0-1)-coloreo*. Sea c el color de v_{j_0}. Intercambiando los colores c y j para todo $j = 1,\ldots,j_0-1$, $j \neq c$, generamos nuevos *(j_0-1)-coloreo*. Obtenemos así j_0-1 soluciones.

Por último, para cada $j = \chi(G), \ldots, n-2, j \neq j_0 - 1$, un *(j)-coloreo* que coloree a v con j_0 si $j_0 \leq j$.

Tenemos $n^2 - \chi(G) - 1$ soluciones que fueron generadas utilizando el mismo procedimiento que en el caso del teorema 5.1, salvo que hemos eliminado $j_0 - 1$ soluciones correspondientes a los *(n)-coloreo* que asignaban colores $1, \ldots, j_0 - 1$ al *vértice* v_{j_0}. En su reemplazo, construimos $j_0 - 2$ nuevas soluciones de un *$(j_0 - 1)$-coloreo* haciendo variar el color de v_{j_0}. La asignación de color de v_{j_0} es única para cada una de estas soluciones. Con argumentos similares a los usados en la demostración de la dimensión y la última observación, podemos afirmar que las soluciones son afínmente independientes.

2. Como v no es universal, existe $v' \in V$ no adyacente a v. Por otro lado, como $\chi(G) \leq n-2$, podemos afirmar que existen $u, u' \neq v$ *vértices* no adyacentes. Consideremos cualquier *(n)-coloreo* que asigne color $n-1$ a v y color n a v'. Llamamos v_i al *vértice* coloreado con color i para $i = 1, \ldots, n$. Resulta entonces $v = v_{n-1}$, $v' = v_n$. A partir de este coloreo, construimos los siguientes:

   - Sean j, i tales que $1 \leq j \leq n-2$, $i \neq j$, $i \leq n-1$. Intercambiamos colores, asignando el color j a v_n, color i a v_j y color n a v_i. Tenemos $(n-2)(n-2)$ coloreos.
   - Para $j = 1, \ldots, n-1$, consideramos el (n)-coloreo que asigna color j a v_n, color n a v_j y color i a v_i $\forall\, i \neq j$, $i \neq n$.

   Sea un *(n-1)-coloreo* que colorea a v_i con i para $i = 1, \ldots, n-2$, y colorea a v_{n-1} y v_n con $n-1$.

   Sea cualquier *(n-1)-coloreo* que asigna color $n-1$ a v_{n-1} y color $n-2$ to u y u'. Para $i = 1, \ldots, n-3$, intercambiando el color i con el color $n-2$, surgen $n-3$ nuevos coloreos.

   Por último, sea cualquier *(j)-coloreo* para cada $j = \chi(G), \ldots, n-2$. En el *(n-2)-coloreo* anterior, podemos intercambiar los colores c y j para $j = 1, \ldots, n-2$, $j \neq c$, siendo c el color originalmente asignado a v_{n-1}. Hay $n-3$ coloreos.

   Tenemos en total $n^2 - \chi(G) - 1$ soluciones. La propiedad de independencia afín resulta siguiendo los mismos argumentos que en casos anteriores y por lo tanto la cara es faceta de $\mathcal{CP}$.

□

El orden establecido para el uso de los colores por las restricciones del modelo, nos permite afirmar que si un coloreo utiliza el color j_0, también utiliza los colores $1, \ldots, j_0 - 1$. Pero entonces, no puede haber más de $n - j_0 + 1$ *vértices* coloreados con los colores $j_0, \ldots, n$. Es decir,

$$\sum_{i \in V} \sum_{j=j_0}^{n} x_{ij} \leq (n - j_0 + 1) w_{j_0}$$

es una desigualdad válida para $\mathcal{CP}$.

No es una restricción válida para $\mathcal{RCP}$. Si $j_0 < n-1$, la solución $x_{ij} = \frac{1}{j_0 - 1}$ para $i, j = 1, \ldots, j_0 - 1$, $x_{ij} = \frac{1}{n - j_0 + 1}$ para $i, j = j_0, \ldots, n$, $w_j = \frac{2}{j_0 - 1}$ para $j = 1, \ldots, j_0 - 1$ y $w_j = \frac{2}{n - j_0 + 1}$

para $j = j_0, \ldots, n$ no la satisface. Si $j_0 = n-1$, la solución $x_{ii} = w_i = 1$ para $i = 1, \ldots, n-2$, $x_{n-1n-1} = x_{n-1n} = \frac{1}{2}$, $x_{nn-1} = x_{nn} = \frac{1}{2}$ y $w_{n-1} = w_n = \frac{1}{2}$ no la satisface (sin pérdida de generalidad estamos suponiendo que n y $n-1$ no son *vértices* adyacentes).

Si $j_0 \geq 2$, cualquier coloreo que asigne el color 1 a más de un *vértice* verifica la desigualdad estrictamente. Todo *(n)-coloreo* satisface por igualdad la restricción. Por lo tanto, es cara de $\mathcal{CP}$.

Si $j_0 = 1$, se satisface la igualdad para todo coloreo ya que resulta combinación de las ecuaciones del sistema minimal de $\mathcal{CP}$.

Si $j_0 = n$, es una de las ecuaciones del sistema minimal de $\mathcal{CP}$.

Bajo ciertas condiciones, esta desigualdad resulta faceta.

Proposición 5.7 *Sea j_0 tal que $2 \leq j_0 \leq n-1$. La desigualdad*

$$\sum_{i \in V} \sum_{j=j_0}^{n} x_{ij} \leq (n - j_0 + 1) w_{j_0}$$

es válida para $\mathcal{CP}$. Es una faceta de $\mathcal{CP}$ si y sólo si

1. *j_0 es tal que $\chi(G) + 1 \leq j_0 \leq n-1$.*
2. *Existe un conjunto independiente de tamaño $n - j_0 + 1$.*
3. *Para $\chi(G) = j_0 - 1$, existe al menos un $(\chi(G))$-coloreo cuya partición en conjuntos independientes es tal que no todos los conjuntos tienen el mismo cardinal.*

Demostración:
La validez de la desigualdad ya la probamos. Veamos la necesidad de las tres condiciones para que defina una faceta.

Si $j_0 \leq \chi(G)$ entonces $w_{j_0} = 1$. Los coloreos que cumplen la igualdad tienen que usar los primeros $j_0 - 1$ colores en $j_0 - 1$ *vértices*. Entonces toda solución de la cara satisface $\sum_{v \in V} x_{vj} = 1$ para todo j=1, $\ldots, j_0 - 1$ y la desigualdad no define faceta de $\mathcal{CP}$. Si $j_0 = n$ la desigualdad es una de las ecuaciones del sistema minimal que define a $\mathcal{CP}$.

Si todo conjunto independiente es de tamaño menor a $n - j_0 + 1$, toda solución de la cara verifica $w_{j_0+1} = w_{j_0}$ y por lo tanto no es faceta.

Si $\chi(G) = j_0 - 1$ y todo *$(\chi(G))$-coloreo* es tal que todos los conjuntos independientes de la partición tienen cardinal r, entonces toda solución de la cara satisface $\sum_{i \in V} x_{ij} = r - (r-1) w_{j_0}$ para $j = 1, \ldots, j_0 - 1$. Por lo tanto la cara no es faceta.

Tenemos entonces que todas las condiciones enunciadas son necesarias. Veamos ahora que también son suficientes para que la cara resulte faceta.

Sea F la cara de $\mathcal{CP}$ definida por la desigualdad y supongamos que la ecuación $\lambda^X X + \lambda^W W = \lambda_0$ es satisfecha por toda solución factible de F. Para afirmar que F es faceta, debemos ver que:

1. $\lambda^X_{vn} + \lambda^W_n = \lambda^X_{vj} \quad \forall\, v \in V \quad \forall\, j = j_0, \ldots, n-1$
2. $(n - j_0 + 1)\lambda^X_{vj} = (n - j_0 + 1)\lambda^X_{vj_0} + \lambda^W_{j_0} \quad \forall\, v \in V \quad \forall\, j = 1, \ldots, j_0 - 1$

3. $\lambda_j^W = 0$ para $j = \chi(G)+1,\ldots,n-1$ $j \neq j_0$
4. $\lambda_{j_0}^W \leq 0$

Claramente, para todo $u, v \in V$ y cualquier *(n)-coloreo* en F, del intercambio de los colores asignados a u y v resulta un *(n)-coloreo* en F y por lo tanto $\lambda_{uj}^X + \lambda_{vj'}^X = \lambda_{uj'}^X + \lambda_{vj}^X$. Esta propiedad será usada en varias ocasiones durante la demostración.

Veamos ahora la demostración de cada una de las relaciones entre los coeficientes de λ.

1. Sea j tal que $j_0 \leq j \leq n-1$. Sean $u, u' \in V$ *vértices* no adyacentes y un *(n-1)-coloreo* que asigne el color j a u y u'. Cambiándole el color a u por el n, obtenemos que

$$\lambda_{uj}^X = \lambda_{un}^X + \lambda_n^W.$$

Consideremos ahora $v \in V$, $v \neq u$. A partir de la relación

$$\lambda_{un}^X + \lambda_{vj}^X = \lambda_{uj}^X + \lambda_{vn}^X$$

y de lo derivado previamente para u, resulta $\lambda_{vj}^X = \lambda_{vn}^X + \lambda_n^W$.

2. Demostramos la identidad para algún *vértice* $v \in V$, ya que para cualquier otro $u \in V$ basta considerar que

$$\lambda_{vj}^X + \lambda_{uj_0}^X = \lambda_{vj_0}^X + \lambda_{uj}^X$$

para derivar

$$(n-j_0+1)\lambda_{uj}^X = (n-j_0+1)\lambda_{uj_0}^X + \lambda_{j_0}^W.$$

Sea C^1 un *(j_0-1)-coloreo* y $I_1, I_2, \ldots, I_{j_0-1}$ la partición de V en conjuntos independientes asociada con el coloreo C^1. Por hipótesis, podemos suponer que $I_1 = \{v_1, v_2, \ldots, v_{r_1}\}$ y $I_2 = \{u_1, u_2, \ldots, u_{r_2}\}$ con $r_1 > r_2$. Sean c_1 y c_2 los colores de I_1 e I_2. Intercambiando los colores c_1 y c_2, se deriva que

$$\sum_{i=1}^{r_1} \lambda_{v_i c_1}^X + \sum_{i=1}^{r_2} \lambda_{u_i c_2}^X = \sum_{i=1}^{r_1} \lambda_{v_i c_2}^X + \sum_{i=1}^{r_2} \lambda_{u_i c_1}^X.$$

Pero sabemos que

$$\lambda_{v_i c_1}^X = \lambda_{v_i c_2}^X - \lambda_{u_i c_2}^X + \lambda_{u_i c_1}^X \quad \forall\, i = 1,\ldots,r_2$$

de donde obtenemos la identidad

$$\sum_{i=r_2+1}^{r_1} \lambda_{v_i c_1}^X = \sum_{i=r_2+1}^{r_1} \lambda_{v_i c_2}^X. \tag{5.6}$$

Si $r_2 + 1 = r_1$, entonces tenemos que $\lambda_{v_{r_1} c_1}^X = \lambda_{v_{r_1} c_2}^X$ Si $r_2 + 1 < r_1$, sabemos que

$$\lambda_{v_{r_1} c_1} - \lambda_{v_i c_1} = \lambda_{v_{r_1} c_2} - \lambda_{v_i c_2} \text{ para todo } i = r_2+1,\ldots,r_1-1. \tag{5.7}$$

Sumando a 5.6 la identidad 5.7 para todo $j = r_2 + 1, \ldots, r_1 - 1$ obtenemos

$$(r_1 - 1 - r_2)\lambda^X_{v_{r_1}c_1} = (r_1 - 1 - r_2)\lambda^X_{v_{r_1}c_2}$$

es decir que $\lambda^X_{v_{r_1}c_1} = \lambda^X_{v_{r_1}c_2}$.

A partir de aquí se deduce fácilmente que $\lambda^X_{vj} = \lambda^X_{vj'}$, $\forall\, v \in V$ y para todo $j, j' = 1, \ldots, j_0 - 1$.

Sea ahora C^2 un *(j_0)-coloreo* perteneciente a F y notemos $z_1, z_2, \ldots, z_{n-j_0+1}$ a los *vértices* coloreados con j_0. Sea c_i el color de z_i en el coloreo C^1, es decir $C^1(z_i) = c_i$.

Como ya sabemos que $\lambda^X_{vj} = \lambda^X_{vj'}$, $\forall\, v \in V$ y para todo $j, j' = 1, \ldots, j_0 - 1$, de la relación entre C^1 y C^2 resulta que

$$\sum_{i=1}^{n-j_0+1} \lambda^X_{z_i c_i} = \sum_{i=1}^{n-j_0+1} \lambda^X_{z_i j_0} + \lambda^W_{j_0}.$$

Pero

$$\lambda^X_{z_i c_i} = \lambda^X_{z_1 c_i} + \lambda^X_{z_i j_0} - \lambda^X_{z_1 j_0} = \lambda^X_{z_1 c_1} + \lambda^X_{z_i j_0} - \lambda^X_{z_1 j_0}.$$

Entonces

$$(n - j_0 + 1)\lambda^X_{z_1 c_1} = (n - j_0 + 1)\lambda^X_{z_1 j_0} + \lambda^W_{j_0}.$$

3. Sea $j = \chi(G) + 1, \ldots, n - 1$, $j \neq j_0$. Por las hipótesis establecidas, podemos afirmar que existe un *(j-1)-coloreo* en F para el cual existen $u, u' \in V$ que comparten el color $j - 1$. De este coloreo podemos derivar el *(j)-coloreo* en F cuya única diferencia con el anterior sea colorear a u con j. De estas similitudes y por los casos anteriores, derivamos trivialmente que $\lambda^W_j = 0$.

4. Llamamos $\{v_1, \ldots, v_n\}$ a los *vértices* de G. Supongamos, sin pérdida de generalidad, que v_1 y v_n son *vértices* no adyacentes. Sea C^1 un *(n-1)-coloreo* tal que $C^1(v_1) = C^1(v_n) = 1$ y $C^1(v_i) = i$ para $i = 2, \ldots, n - 1$. Sea C^2 un *(n)-coloreo* tal que $C^2(v_n) = n$ y $C^2(v) = C^1(v)$ para $v \neq v_n$. C^2 está en F, entonces se verifica que

$$\lambda_0 = \sum_{i=1}^{n} \lambda^X_{v_i i} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_{j_0} + \lambda^W_n.$$

C^1 no está en F pero verifica la desigualdad

$$\lambda_0 \geq \lambda^X_{v_1 1} + \lambda^X_{v_n 1} + \sum_{i=2}^{n-1} \lambda^X_{v_i i} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_{j_0}.$$

Usando la expresión de λ_0, se deduce que

$$\lambda^X_{v_n n} - \lambda^X_{v_n 1} + \lambda^W_n \geq 0$$

Por la relación derivada entre los coeficientes sabemos que $\lambda^X_{v_n n} + \lambda^W_n = \lambda^X_{v_n j_0}$ y que $\lambda^X_{v_n 1} = \lambda^X_{v_n j_0} + (\lambda^W_{j_0}/(n - j_0 + 1))$. Entonces $\lambda^W_{j_0} \leq 0$.

Concluimos entonces que (λ^X, λ^W) es combinación lineal de las ecuaciones del modelo y de la desigualdad, por lo tanto la cara es faceta de $\mathcal{CP}$.

□

5.5. Desigualdades Válidas Derivadas de Conjuntos Independientes

Sea $\alpha(G)$ el número independiente de G y supongamos que $\alpha(G) \leq n-2$. Como G no es completo, entonces $\alpha(G) \geq 2$. Los *vértices* del *grafo* coloreados con un mismo color son un conjunto independiente. Entonces, la cantidad de vértices que usan un mismo color no puede superar a $\alpha(G)$, cardinal de un conjunto independiente máximo de G. La desigualdad

$$\sum_{i \in V} x_{ij_0} \leq \alpha(G) w_{j_0}$$

es válida para $\mathcal{CP}$.

Una primera observación que podemos hacer es que los coloreos que cumplen la desigualdad por igualdad satisfacen $w_j = 0$ para todo $j \geq n - \alpha(G) + 2$. Esto nos lleva a concluir que la desigualdad es cara de $\mathcal{CP}$ pero no define una faceta. En el caso que j_0 es tal que $j_0 \leq n - \alpha(G)$, las soluciones de la cara satisfacen $\sum_{v \in V} x_{vn-\alpha(G)+1} = w_{n-\alpha(G)+1}$, propiedad que disminuye aún más la dimensión de la cara.

La dimensión de la cara está relacionada de alguna manera con ciertas propiedades del *grafo*. Por ejemplo, si existe $v \in V$ tal que no forma parte de ningún conjunto independiente máximo, entonces las soluciones factibles de la cara satisfacen $x_{vj_0} = 0$. La presencia de *vértices* universales es un caso particular de esta situación. Similarmente, si existen $v, v' \in V$ *vértices* adyacentes que tienen la propiedad que cualquier conjunto independiente máximo contine a alguno de ellos, entonces las soluciones de la cara verifican que $x_{vj_0} + x_{v'j_0} = w_{j_0}$. Por ejemplo, este es el caso de un grafo que sea un *agujero* par o *camino* par. Hasta el momento no hemos podido encontrar condiciones necesarias y suficientes para caracterizar los grafos para los cuales la dimensión de la cara es máxima.

Supongamos que $j_0 \leq n - \alpha(G)$. A partir de la primera observación que hicimos acerca de la desigualdad, podemos realizar el siguiente análisis. En cualquier *(n-α(G)+r)-coloreo* con $r \geq 1$ tenemos $\alpha(G) - r$ *vértices* repitiendo color, por lo tanto la cantidad máxima de *vértices* que pueden estar coloreados con los colores $j_0, n - \alpha(G) + 1, \ldots, n - \alpha(G) + r$ es igual a $1 + (n - \alpha(G) + r) - (n - \alpha(G) + 1) + 1 + \alpha(G) - r = \alpha(G) + 1$. Es decir que

$$\sum_{v \in V} x_{vj_0} + \sum_{j=n-\alpha(G)+1}^{n} \sum_{v \in V} x_{vj} \leq \alpha(G) w_{j_0} + w_{n-\alpha(G)+1}$$

es desigualdad válida de $\mathcal{CP}$.

Cualquier *(n)-coloreo* satisface por igualdad esta desigualdad y un *(n-1)-coloreo* que repita un color distinto a $j_0, n - \alpha(G) + 1, \ldots, n - 1$ la satisface estrictamente. Por lo tanto es una cara de $\mathcal{CP}$.

En el caso particular que $j_0 = n - \alpha(G) + 1$, el razonamiento previo nos lleva a la desigualdad $\sum_{j=j_0}^{n} \sum_{v \in V} x_{vj} \leq (n - j_0 + 1) w_{j_0}$ que ya fue analizada.

En la siguiente proposición caracterizamos cuando la desigualdad resulta faceta.

Proposición 5.8 *Sea $j_0 \leq n - \alpha(G)$. La desigualdad válida*

$$\sum_{v \in V} x_{vj_0} + \sum_{j=n-\alpha(G)+1}^{n} \sum_{v \in V} x_{vj} \leq \alpha(G) w_{j_0} + w_{n-\alpha(G)+1}$$

define faceta de $\mathcal{CP}$ si y sólo si

- $\chi(G) \leq n - \alpha(G)$
- *existe algún* $(\chi(G))$-coloreo *que satisface la igualdad.*

Demostración:

Veamos la necesidad de las condiciones. Si $\chi(G)$ es igual a $n - \alpha(G) + 1$, los coloreos que satisfacen la desigualdad por igualdad verifican además que $\sum_{v \in V} x_{vj} = 1$ para $j \neq j_0$ y $j \leq n - \alpha(G)$. Entonces la desigualdad no es faceta. La segunda condición es necesaria para que la cara no se encuentre contenida en el hiperplano $w_{\chi(G)+1} = 1$.

Sea F la cara de $\mathcal{CP}$. Supongamos que $\lambda^X X + \lambda^W W \leq \lambda_0$ es una desigualdad válida para $\mathcal{CP}$ tal que $F \subseteq \mathcal{CP} \cap \{(X, W) : \lambda^X X + \lambda^W W = \lambda_0\}$. Debemos demostrar que

1. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n \quad \forall\, v \in V,\ \forall\, j = j_0, n - \alpha(G) + 1, \ldots, n - 1$
2. $\lambda^X_{vj} = \lambda^X_{vn-\alpha(G)+1} + \lambda^W_{n-\alpha(G)+1} \quad \forall\, v \in V,\ \forall\, j = 1, \ldots, n - \alpha(G) \quad j \neq j_0$
3. si $j_0 \geq \chi(G) + 1$ entonces $\lambda^W_{j_0} = \alpha(G) \lambda^W_{n-\alpha(G)+1}$
4. $\lambda^W_j = 0$ para $j = \chi(G) + 1, \ldots, n - 1$ y $j \neq j_0, n - \alpha(G) + 1$
5. $\lambda^W_{n-\alpha(G)+1} \leq 0$

1. Sean $j = j_0, n - \alpha(G) + 1, \ldots, n - 1$ y v, v' y $v" \in V$ tal que $v', v"$ son no adyacentes. Sea C^1 un *(n-1)-coloreo* tal que $C^1(v') = C^1(v") = j$. Cambiándole el color a v' por el color n obtenemos C^2 un *(n)-coloreo*. C^1y C^2 están en F y por sus coincidencias concluimos que $\lambda^X_{v'j} = \lambda^X_{v'n} + \lambda^W_n$. Si ahora consideramos C^3 un *(n)-coloreo* en F tal que $C^3(v') = n$ y $C^3(v) = j$, del intercambio de colores entre v y v' sabemos que $\lambda^X_{vj} + \lambda^X_{v'n} = \lambda^X_{vn} + \lambda^X_{v'j}$ a partir de lo cual podemos concluir que $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n$.

2. Sea C^1 un *(n-$\alpha(G)$)-coloreo* en F tal que C^1 repite el color j_0 en exactamente $\alpha(G)$ *vértices*. Si $j_0 > \chi(G)$, simpre es posible construirse un coloreo con esta propiedad. Para el caso que $j_0 \leq \chi(G)$, también resulta posible debido a la tercera hipótesis de la proposición. Podemos afirmar

además que existe otro color j que se repite. Es decir, existen v' y $v"$ tal que $C^1(v') = C^1(v") = j$. Cambiándole el color a v' por el color $n-\alpha(G)+1$ obtenemos $\lambda^X_{v'j} = \lambda^X_{v'n-\alpha(G)+1} + \lambda^W_{n-\alpha(G)+1}$. Si ahora consideramos un *(n)-coloreo* tal que $C^3(v') = n - \alpha(G) + 1$ y $C^3(v) = j$, del intercambio de colores entre v y v' sabemos que $\lambda^X_{vj} + \lambda^X_{v'n-\alpha(G)+1} = \lambda^X_{vn-\alpha(G)+1} + \lambda^X_{v'j}$ a partir de lo cual podemos concluir que $\lambda^X_{vj} = \lambda^X_{vn-\alpha(G)+1} + \lambda^W_{n-\alpha(G)+1}$.

3. Supongamos que j_0 es tal que $j_0 \geq \chi(G) + 1$. Sea C^1 un *$(j_0 - 1)$-coloreo* en F y C^2 un *(j_0)-coloreo* en F. Estos coloreos siempre existen por hipótesis. Llamemos v_i a los *vértices* tales que $C^2(v_i) = j_0$ para $i = 1, \ldots, \alpha(G)$ y c^1_v, c^2_v a los colores que recibe un *vértice* v en el coloreo C^1 y C^2 respectivamente. Sabemos que $c^1_v \leq n - \alpha(G)$ y $c^1_v \neq j_0$ para todo $v \in V$ y $c^2_v \leq n - \alpha(G)$ y $c^2_v \neq j_0$ para todo $v \in V\backslash\{v_1, \ldots, v_{\alpha(G)}\}$. Por encontrarse ambos coloreos en F tenemos que

$$\sum_{v \in V-\{v_1,\ldots,v_{\alpha(G)}\}} \lambda^X_{vc^1_v} + \sum_{i=1}^{\alpha(G)} \lambda^X_{v_i c^1_{v_i}} = \sum_{v \in V-\{v_1,\ldots,v_{\alpha(G)}\}} \lambda^X_{vc^2_v} + \sum_{i=1}^{\alpha(G)} \lambda^X_{v_i j_0} + \lambda^W_{j_0}$$

Por las propiedades demostradas antes sabemos que

$$\lambda^X_{vc^1_v} = \lambda^X_{vc^2_v} \text{ para todo } v \in V\backslash\{v_1, \ldots, v_{\alpha(G)}\}$$

$$\lambda^X_{v_i j_0} = \lambda^X_{v_i n-\alpha(G)+1}$$

y

$$\lambda^X_{v_i c^1_{v_i}} = \lambda^X_{v_i n-\alpha(G)+1} + \lambda^W_{n-\alpha(G)+1}$$

Se desprende entonces que $\lambda^W_{j_0} = \alpha(G)\lambda^W_{n-\alpha(G)+1}$

4. Consideremos ahora $j = \chi(G)+1, \ldots, n-1$ y $j \neq j_0, n-\alpha(G)+1$. Existe C^1 un *(j-1)-coloreo* en F que además podemos suponer que repite el color r con $r = j-1$ si $j-1 \neq j_0$ ó $r = j-2$ en caso contrario. Sea C^2 un *(j)-coloreo* que colorea con j a uno de los *vértices* que tiene el color r. Por las coincidencias de los coloreos C^1 y C^2 resulta $\lambda^X_{vr} = \lambda^X_{rj} + \lambda^W_j$. Pero $\lambda^X_{vr} = \lambda^X_{rj}$ ya que, si $j < n - \alpha(G) + 1$ entonces $r < n - \alpha(G) + 1$ y si $j \geq n - \alpha(G) + 2$ entonces $r = j - 1 \geq n - \alpha(G) + 1$. Entonces que $\lambda^W_j = 0$.

5. Sea $j \neq j_0$ y $j < n - \alpha(G) + 1$. Llamamos $\{v_1, \ldots, v_n\}$ a los *vértices* de G. Supongamos, sin pérdida de generalidad, que v_j y v_n son *vértices* no adyacentes. Sea C^1 un *(n-1)-coloreo* tal que $C^1(v_1) = C^1(v_n) = j$ y $C^1(v_i) = i$ para $i = 1, \ldots, n-1$, $i \neq j$ Sea C^2 un *(n)-coloreo* tal que $C^2(v_n) = n$ y $C^2(v) = C^1(v)$ para $v \neq v_n$. C^2 está en F, entonces se verifica que

$$\lambda_0 = \sum_{i=1}^{n} \lambda^X_{v_i i} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_{n-\alpha(G)+1} + \lambda^W_n$$

C^1 no está en F pero verifica la desigualdad

$$\lambda_0 \geq \lambda^X_{v_n j} + \sum_{i=1}^{n-1} \lambda^X_{v_i i} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_{n-\alpha(G)+1}$$

Usando la expresión de λ_0, se deduce que

$$\lambda^X_{v_n n} - \lambda^X_{v_n j} + \lambda^W_n \geq 0$$

Por la relación derivada entre los coeficientes sabemos que $\lambda^X_{v_n n} + \lambda^W_n = \lambda^X_{v_n n-\alpha(G)+1}$ y que $\lambda^X_{v_n j} = \lambda^X_{v_n n-\alpha(G)+1} + \lambda^W_{n-\alpha(G)+1}$. Entonces $\lambda^W_{n-\alpha(G)+1} \leq 0$

Concluimos entonces que (λ^X, λ^W) es combinación lineal de las ecuaciones del modelo y de la desigualdad, por lo tanto la cara es una faceta de $\mathcal{CP}$.

□

Tal vez la objeción mas fuerte que pueda hacerse sobre estas desigualdades es que dependen de $\alpha(G)$ cuyo cálculo es tan difícil como el de $\chi(G)$ [51].

Sin embargo, para ciertas subestructuras del grafo este número es fácil de calcular y puede derivar desigualdades válidas. Si $G' = G[V']$ con $V' \subset V$, entonces es válida la desigualdad

$$\sum_{v \in V'} x_{vj_0} \leq \alpha(G') w_{j_0}.$$

Un *(n-α(G')+1)-coloreo* que use el color j_0 en un conjunto independiente máximo de G' verifica la igualdad. Entonces, la desigualdad es una cara de $\mathcal{CP}$. La dimensión de la cara depende de la estructura de G' ya que valen las mismas observaciones que hemos realizado para G al inicio de esta sección. Para el caso que $\alpha(G') \geq 2$, la desigualdad no es faceta pues ningún *(n)-coloreo* la satisface por igualdad.

Se puede reforzar la desigualdad para aumentar la dimensión de la cara que define. Con los mismos argumentos que utilizamos antes, resulta válida la desigualdad

$$\sum_{v \in V'} x_{vj_0} + \sum_{v \in V} \sum_{j=n-\alpha(G')+1}^{n} x_{vj} \leq \alpha(G') w_{j_0} + w_{n-\alpha(G')+1}$$

En el caso que $\alpha(G')$ es igual a $\alpha(G)$ la desigualdad está dominada por la que ya hemos analizado.

Supondremos entonces que $1 \leq \alpha(G') < \alpha(G)$. En la siguiente proposición presentamos condiciones necesarias y suficientes sobre los subgrafos para los cuales la desigualdad resulta ser faceta.

Proposición 5.9 *Sea $V' \subset V$ y $G' = G[V']$ tal que $\alpha(G') < \alpha(G)$, $j_0 \leq n - \alpha(G')$ y la desigualdad válida*

$$\sum_{v \in V'} x_{vj_0} + \sum_{v \in V} \sum_{j=n-\alpha(G')+1}^{n} x_{vj} \leq \alpha(G') w_{j_0} + w_{n-\alpha(G')+1}$$

Supongamos que:

- *para todo $v \in V \backslash V'$ existe conjunto independiente de tamaño $\alpha(G') + 1$ en $G[V' \cup \{v\}]$.*
- *existe I, conjunto independiente máximo de G' tal que $V \backslash I$ no es* clique.
- *existe algún* ($\chi(G)$)-coloreo *que satisface por igualdad la desigualdad.*

Entonces la desigualdad define una faceta.

Demostración:

Veamos la necesidad de la primera condición. Sea $v \in V \backslash V'$ y un coloreo que asigne a v el color j_0. Si no existe I conjunto independiente de V' tal que $|I| = \alpha(G')$ y $[v'v] \notin E$ para todo $v' \in V'$, los coloreos que se encuentran en la cara deben usar al menos $n - \alpha(G') + 1$ colores. Por el orden establecido entre los colores, los colores j con $j = 1, \ldots, n - \alpha(G')$ y $j \neq j_0$ deben usarse para colorear $n - \alpha(G') - 1 - 1$ *vértices*, lo que nos lleva a un absurdo. Entonces la cara se encuentra contenida en el hiperplano $x_{vj_0} = 0$ y no es faceta.

La tercera hipótesis es necesaria para que la cara no se encuentre contenida en el hiperplano $w_{\chi(G')+1} = 1$.

La segunda hipótesis no es necesaria. El *grafo* G de 4 *vértices* y aristas $[v_1, v_2], [v_2, v_3], [v_2, v_4]$ tiene a la desigualdad como faceta para $G' = G[v_1, v_2, v_3]$ y sin embargo $V \backslash I$ es una clique para todo conjunto independiente máximo de V'.

Veamos que estas hipótesis son suficientes para que la desigualdad resulte faceta.

Sea F la cara determinada por la desigualdad. Supongamos que $\lambda^X X + \lambda^W W \leq \lambda_0$ es una desigualdad válida para $\mathcal{CP}$ tal que $F \subseteq \mathcal{CP} \cap \{(X, W) : \lambda^X X + \lambda^W W = \lambda_0\}$.

Debemos demostrar que

1. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n \quad \forall\, v \in V,\ \forall\, j = n - \alpha(G') + 1, \ldots, n-1$
2. $\lambda^X_{vj_0} = \lambda^X_{vn} + \lambda^W_n \quad \forall\, v \in V'$
3. $\lambda^X_{vj_0} = \lambda^X_{vn-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1} \quad \forall\, v \in V \backslash V'$
4. $\lambda^X_{vj} = \lambda^X_{vn-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1} \quad \forall\, v \in V,\ \forall\, j \neq j_0$ y $j \leq n - \alpha(G')$
5. si $j_0 \geq \chi(G) + 1$ entonces $\lambda^W_{j_0} = \alpha(G')\lambda^W_{n-\alpha(G')+1}$
6. $\lambda^W_j = 0$ para $j = \chi(G) + 1, \ldots, n-1$ y $j \neq j_0, n - \alpha(G') + 1$
7. $\lambda^W_{n-\alpha(G')+1} \leq 0$

Veamos cada uno de los casos.

1. Sean j tal que $n - \alpha(G') + 1 \leq j \leq n - 1$ y $v \in V$. Sean $v_1 \in V \backslash V'$ y $v_2 \in V$ *vértices* no adyacentes. Consideremos C^1 un *(n-1)-coloreo* en F tal que asigna el color j_0 a algún *vértice* de V' y el color j a los *vértices* v_1 y v_2. Podemos construir C^2 un *(n)-coloreo* en F, que coloree a todos los *vértices* del grafo como C^1, salvo que $C^2(v_1) = n$. De las coincidencias de los coloreos derivamos que $\lambda^X_{v_1 j} = \lambda^X_{v_1 n} + \lambda^W_n$.

   Si ahora consideramos C^3 un *(n)-coloreo* en F tal que $C^3(v) = n$ y $C^3(v_1) = j$ y un *vértice* de V' con color j_0, del intercambio de colores entre v y v_1 sabemos que $\lambda^X_{vn} + \lambda^X_{v_1 j} = \lambda^X_{vj} + \lambda^X_{v_1 n}$ a partir de lo cual podemos concluir que $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n$.

2. Sean $v_1 \in V'$ y $v_2 \in V \backslash V'$ *vértices* no adyacentes. Sea C^1 un *(n-1)-coloreo* en F tal que $C^1(v_1) = C^1(v_2) = j_0$. Basta cambiarle el color a v_1 por n para obtener $\lambda^X_{v_1 j_0} = \lambda^X_{v_1 n} + \lambda^W_n$.

   Sea $v \in V'$, $v \neq v_1$ y C^3 un *(n)-coloreo* en F tal que $C^3(v_1) = j_0$ y $C^3(v) = n$. Intercambiando los colores de v_1 y v, deducimos que $\lambda^X_{vj_0} = \lambda^X_{vn} + \lambda^W_n$.

3. Sea $v \in V \backslash V'$. Por las hipótesis establecidas, podemos construir C^1 un *(n-$\alpha(G')$)-coloreo* en F tal que $C^1(v) = j_0$ y $\alpha(G')$ *vértices* de V' son coloreados con j_0. Cambiándole el color a v por el color $n - \alpha(G') + 1$ concluimos que $\lambda^X_{vj_0} = \lambda^X_{vn-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1}$.

4. Sea j tal que $j \leq n - \alpha(G')$ y $j \neq j_0$. Por hipótesis existe $I \subset V'$ conjunto independiente máximo tal que $V \backslash I$ no es *clique*. Sean $v_1, v_2 \in V \backslash I$ *vértices* no adyacentes y C^1 un *(n-$\alpha(G')$)-coloreo* en F tal que $C^1(z) = j_0$ para todo $z \in I$ y $C^1(v_1) = C^1(v_2) = j$. Cambiando el color de v_1 al color $n - \alpha(G') + 1$ obtenemos $\lambda^X_{v_1 j} = \lambda^X_{v_1 n-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1}$.

   Sea $v \in V$ y C^3 un *(n)-coloreo* en F tal que $C^3(v_1) = n-\alpha(G')+1$ y $C^3(v) = j$. Intercambiando los colores de v_1 y v, deducimos que $\lambda^X_{vj} = \lambda^X_{vn-\alpha(G')+1} + \lambda^W_{n-\alpha(G)+1}$.

5. Supongamos que j_0 es tal que $j_0 \geq \chi(G) + 1$. Sea $I = \{z_1, \ldots, z_{\alpha(G')}\}$ conjunto independiente máximo de G'. Sea C^1 un *($j_0 - 1$)-coloreo* y C^2 un *(j_0)-coloreo* tal que $C^2(z_i) = j_0$ para todo $i = 1, \ldots, \alpha(G')$. Para todo $v \in V$, llamamos c^1_v, c^2_v a los colores asignados a v por los coloreos C^1 y C^2 respectivamente. Como C^1 y C^2 se encuentran en F, tenemos que

$$\sum_{v \in V-I} \lambda^X_{vc^1_v} + \sum_{i=1}^{\alpha(G')} \lambda^X_{z_i c^1_{z_i}} = \sum_{v \in V-I} \lambda^X_{vc^2_v} + \sum_{i=1}^{\alpha(G')} \lambda^X_{z_i j_0} + \lambda^W_{j_0}$$

   Pero $\lambda^X_{vc^1_v} = \lambda^X_{vc^2_v}$ para todo $v \in V \backslash I$ pues c^1_v y c^2_v son menores o iguales a $j_0 - 1$ y $j_0 - 1 \leq n - \alpha(G') - 1$. Además,

$$\lambda^X_{z_i c^1_{z_i}} = \lambda^X_{z_i n-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1} = \lambda^X_{z_i j_0} + \lambda^W_{n-\alpha(G')+1}$$

   A partir de estas propiedades, podemos deducir que $\lambda^W_{j_0} = \alpha(G')\lambda^W_{n-\alpha(G')+1}$

6. Consideremos ahora j para $j = \chi(G) + 1, \ldots, n-1$ y $j \neq j_0, n - \alpha(G') + 1$. Existe C^1 un *(j-1)-coloreo* en F que además podemos suponer que repite el color r, con $r = j - 1$ si $j - 1 \neq j_0$ ó $r = j - 2$ en caso contrario. Sea C^2 un *(j)-coloreo* que colorea con j a uno de los *vértices* que tiene el color r. Por las coincidencias de los coloreos C^1 y C^2 resulta $\lambda^X_{vr} = \lambda^X_{rj} + \lambda^W_j$. Pero $\lambda^X_{vr} = \lambda^X_{rj}$ ya que, si $j < n - \alpha(G') + 1$ entonces $r < n - \alpha(G') + 1$ y si $j \leq n - \alpha(G') + 2$ entonces $r = j - 1 \geq n - \alpha(G') + 1$. De aquí resulta $\lambda^W_j = 0$.

7. Llamamos $\{v_1, \ldots, v_n\}$ a los *vértices* de G. Supongamos, sin pérdida de generalidad, que $v_{j_0} \in V'$ y $v_{n-\alpha(G')+1} \in V \backslash V'$ son *vértices* no adyacentes. Sea C^1 un *(n)-coloreo* tal que $C^1(v_{j_0}) = j_0$, $C^1(v_{n-\alpha(G')+1}) = n - \alpha(G') + 1$ y $C^1(v_i) = i$ para $i = 1, \ldots, n$ con $i \neq j_0, n - \alpha(G') + 1$. Sea C^2 un *(n)-coloreo* que resulta de intercambiar los colores de $v_{n-\alpha(G')+1}$ y v_{j_0}. Como C^1 está en F, entonces se verifica que

$$\lambda_0 = \sum_{i=1}^{n} \lambda^X_{v_i i} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_{n-\alpha(G')+1} + \lambda^W_n$$

C^2 no está en F pero verifica la desigualdad

$$\lambda_0 \geq \lambda^X_{v_{j_0}n-\alpha(G')+1} + \lambda^X_{v_{n-\alpha(G')+1}j_0} + \sum_{i=1,i\neq j_0}^{n-\alpha(G')} \lambda^X_{v_i i} + \sum_{i=n-\alpha(G')+2}^{n} \lambda^X_{v_i i} + \sum_{j=1}^{\chi(G)} \lambda^W_j + \lambda^W_{n-\alpha(G')+1} + \lambda^W_n$$

Usando la expresión de λ_0, se deduce que

$$\lambda^X_{v_{j_0}j_0} + \lambda^X_{v_{n-\alpha(G')+1}n-\alpha(G')+1} - \lambda^X_{v_{j_0}n-\alpha(G')+1} - \lambda^X_{v_{n-\alpha(G')+1}j_0} \geq 0$$

Pero por la relación derivada entre los coeficientes sabemos que $\lambda^X_{v_{j_0}j_0} = \lambda^X_{v_{j_0}n-\alpha(G')+1}$ y que $\lambda^X_{v_{n-\alpha(G')+1}j_0} = \lambda^X_{v_{n-\alpha(G')+1}n-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1}$. Entonces $\lambda^W_{n-\alpha(G')+1} \leq 0$

Concluimos entonces que (λ^X, λ^W) es combinación lineal de las ecuaciones del modelo y de la desigualdad y por lo tanto la cara es una faceta de $\mathcal{CP}$.

□

Para el caso que no se cumpla la primera condición de la proposición 5.9 podemos realizar un proceso de *lifting*, es decir un proceso destinado a incrementar la dimensión de la cara.

Hay un resultado de la teoría de desigualdades válidas [66] que brinda una herramienta muy útil para realizar este proceso y es el siguiente:

Lema: *Sea* $S \subset \mathbb{R}^n$, $S^\delta = S \cap \{x \in^n: x_1 = \delta\}$ *para* $\delta \in \{0,1\}$. *Sea* $\sum_{j=2}^n \pi_j x_j \leq \pi_0$ *una desigualdad válida para* S^0. *Si* $S^1 \neq \emptyset$ *entonces la desigualdad*

$$\alpha_1 x_1 + \sum_{j=2}^{n} \pi_j x_j \leq \pi_0$$

es válida para S *para cualquier* $\alpha_1 \leq \pi_0 - \zeta$ *donde* $\zeta = max(\sum_{j=2}^n \pi_j x_j : x \in S^1)$. *Si la desigualdad es cara de dimensión* k *de* $conv(S^0)$ *y* $\alpha_1 = \pi_0 - \zeta$ *entonces la nueva desigualdad es cara de dimensión* $k+1$ *de* $conv(S)$.

Para aplicar este procedimiento a nuestra desigualdad, sea

$$LiftV = \{v \in V \text{ tal que } x_{vj_0} = 0 \text{ para las soluciones de la cara}\}.$$

Si realizamos el proceso en forma iterativa con los vértices de $LiftV$, el coeficiente máximo de *lifting* será

$$\alpha_{vj_0} = -max_{x_{vj_0}=1}\Big(\sum_{v\in V'} x_{vj_0} + \sum_{j=n-\alpha(G')+1}^{n} \sum_{v\in V} x_{vj} + \sum_{v\in LiftV'} \alpha_{vj_0} x_{vj_0} - \alpha(G')w_{j_0} - w_{n-\alpha(G')+1}\Big)$$

donde $LiftV'$ son los *vértices* de $LiftV$ para los cuales ya se realizó el *lifting*. El valor de α_{vj_0} dependerá de la cantidad de *vértices* adyacentes que tenga v en V' y en $LiftV'$ y que está vinculado al orden en que se tomen las variables para el proceso de *lifting*. Un posible orden es considerar en primer lugar los *vértices* de una *clique* K que sean adyacentes a todos los *vértices* de V'. Supongamos

además que K es maximal respecto a esta propiedad de adyacencia a los *vértices* de V'. Para estos *vértices*, el coeficiente de *lifting* máximo se obtiene con cualquier *(n)-coloreo* que asigne el color j_0 al *vértice* considerado. Resulta entonces que $\alpha_{vj_0} = 1$. Finalizado el proceso de *lifting* se obtiene la desigualdad

$$\sum_{v \in V'} x_{vj_0} + \sum_{j=n-\alpha(G')+1}^{n} \sum_{v \in V} x_{vj} + \sum_{v \in K} x_{vj_0} \leq \alpha(G') w_{j_0} + w_{n-\alpha(G')+1}$$

cuya dimensión respecto a la cara original aumentó en el cardinal de K. Para el resto de los *vértices* no podemos dar una fórmula explícita para su coeficiente pues dependerá de sus adyacencias y del orden en que sean considerados.

Hemos obtenido una desigualdad basada en una subestructura y en su correspondiente número independiente. Como ya hemos mencionado, es beneficioso trabajar sobre subestructuras con $\alpha(G')$ conocido. Resultan de mayor interés aquellas en la cuales algunas de las hipótesis de la Proposición 5.9 se satisfacen sin necesidad de restringir aún más las características propias del grafo. A continuación presentamos algunas estructuras que tienen las propiedades mencionadas.

5.5.1. Desigualdades *Clique*

Consideremos $K \subset V$, una *clique* maximal de G. Claramente, $\alpha(G[K]) = 1$ y la desigualdad *Clique*

$$\sum_{p \in K} x_{pj_0} \leq w_{j_0}$$

es válida para $\mathcal{CP}$ para todo $j_0 = 1, \ldots, n-1$.

Cualquier *(n)-coloreo* que no use a j_0 en la coloración de K verifica estrictamente la desigualdad y cualquier *(n)-coloreo* que use a j_0 para colorear K la cumple por igualdad. Entonces, es una cara de $\mathcal{CP}$.

Para *cliques* con más de tres *vértices*, la desigualdad no es válida para $\mathcal{RCP}$ ya que la solución $x_{ij} = \frac{1}{n}$ y $w_j = \frac{2}{n}$ para todo $i, j = 1, \ldots, n$ no la verifica.

Veamos si se verifican las condiciones para poder afirmar que esta desigualdad define una faceta de $\mathcal{CP}$. Como K es *clique*, cualquier *vértice* de la *clique* es un conjunto independiente máximo. Además, por ser maximal, todo *vértice* en $V \backslash K$ tiene un *vértice* no adyacente en K. Entonces,

- Para todo $v \in V \backslash K$ existe conjunto independiente de medida $\alpha(G') + 1 = 2$ en $G[K \cup \{v\}]$.
- Existe $u \in K$ tal que $V \backslash \{u\}$ no es una *clique*
- Cualquier coloreo que use a j_0 en algún *vértice* de K satisface por igualdad la desigualdad.

Se satisfacen todas las hipótesis de la Proposición 5.9. Podemos afirmar entonces:

Proposición 5.10 *Si K es* clique *maximal, la desigualdad Clique*

$$\sum_{p \in K} x_{pj_0} - w_{j_0} \leq 0$$

es faceta de $\mathcal{CP}$ para todo $j_0 = 1, \ldots, n-1$.

5.5.2. Desigualdades *Agujero*

Sea $C_k = \{v_1, v_2, \ldots, v_k\}$ un *ciclo* de medida $k > 3$ y $\alpha(G[C_k])$ el número independiente de $G[C_k]$. Supongamos que C_k es un *agujero*. Entonces $\alpha(G[C_k]) = \lfloor k/2 \rfloor$ y la desigualdad

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{j=n-\lfloor k/2 \rfloor+1}^{n} \sum_{v \in V} x_{vj} \leq \lfloor k/2 \rfloor w_{j_0} + w_{n-\lfloor k/2 \rfloor+1}$$

es válida para $\mathcal{CP}$ para $j_0 \leq n - \lfloor k/2 \rfloor$.

Para poder afirmar que la desigualdad es faceta, deben cumplirse las hipótesis de la Proposición 5.9. Como estamos considerando el caso en que C_k es un *agujero*, siempre existe I, conjunto independiente máximo, tal que $V \backslash I$ no es *clique*. Sólo es necesario exigir el resto de las hipótesis. En la siguiente proposición presentamos este resultado.

Proposición 5.11 *Sea* $C_k = \{v_1, \ldots, v_k\}$ *un* agujero *de tamaño* k, $k > 3$ *y* $j_0 \leq n - \lfloor k/2 \rfloor$. *La desigualdad* ***Agujero***

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{j=n-\lfloor k/2 \rfloor+1}^{n} \sum_{v \in V} x_{vj} \leq \lfloor k/2 \rfloor w_{j_0} + w_{n-\lfloor k/2 \rfloor+1}$$

es faceta de $\mathcal{CP}$ *si se satisfacen las siguientes condiciones:*

- *para todo* $v \in V \backslash C_k$ *existe un conjunto independiente de medida* $\lfloor k/2 \rfloor + 1$ *en* $G[C_k \cup \{v\}]$,
- *existe un* $(\chi(G))$-coloreo *que satisface la desigualdad por igualdad* .

En el caso que se realice el procedimiento de *lifting* que mencionamos antes, podemos pensar a la desigualdad resultante como la desigualdad original sobre $G[C_k \cup K]$ donde K es una clique maximal adyacente a todo los vértices de C_k. Esta estructura la denominamos ***Rueda Generalizada***. La desigualdad sobre esta estructura resulta ser

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{v \in K} x_{v j_0} + \sum_{j=n-\lfloor k/2 \rfloor+1}^{n} \sum_{v \in V} x_{vj} \leq \lfloor k/2 \rfloor w_{j_0} + w_{n-\lfloor k/2 \rfloor+1}$$

5.5.3. Desigualdades *Complemento de Agujeros*

Sea $\bar{C}_k = \{v_1, \ldots, v_k\}$ el complemento de un *ciclo* de tamaño k. Si C_k es un *agujero*, el cardinal del máximo conjunto independiente de $G[\bar{C}_k]$ es 2.

Por lo tanto

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{j=n-2+1}^{n} \sum_{v \in V} x_{vj} \leq 2w_{j_0} + w_{n-1}$$

es una desigualdad válida de $\mathcal{CP}$ para $j_0 \leq n - 2$.

Nuevamente en este caso, una de las hipótesis de la Proposición 5.9 se satisface ya que podemos afirmar que existe I, conjunto independiente máximo, tal que $V \backslash I$ no es *clique*.

Las condiciones bajo las cuales resulta faceta son dadas en la siguiente proposición.

Proposición 5.12 *Sea $\bar{C}_k = \{v_1, \ldots, v_k\}$ el complemento de un agujero de tamaño k. Si*

- *algún $(\chi(G))$-coloreo verifica la desigualdad por igualdad*
- *para todo $v \in V \backslash \bar{C}_k$ existe un conjunto independiente de cardinal 3 en $G[\bar{C}_k \cup \{v\}]$*

entonces la desigualdad Complemento de Agujero

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{v \in V} x_{vn-1} \leq 2w_{j_0} + w_{n-1} - w_n$$

es faceta de $\mathcal{CP}$.

5.5.4. Desigualdades *Camino*

Sea $P_k = \{v_1, v_2, \ldots, v_k\}$ un *camino* de longitud k. Sabemos que $\alpha(G') = \lceil k/2 \rceil$.

Podemos afirmar que siempre existe $I \subset P_k$, conjunto independiente máximo tal que $V \backslash I$ no es *clique*, salvo para P_3. En este caso particular deberemos pedir que $V \backslash \{v_1, v_3\}$ no sea *clique*.

Se satisface la segunda hipótesis de la Proposición 5.9. Sólo es necesario pedir el resto de las condiciones. Las siguientes proposiciones caracterizan las facetas.

Proposición 5.13 *Sea P_k un* camino *de longitud k y la desigualdad válida*

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{v \in V} \sum_{j=n-\lceil k/2 \rceil + 1}^{n} x_{vj} \leq \lceil k \rceil w_{j_0} + w_{n-\lceil k \rceil + 1}$$

Si se satisfacen las siguientes condiciones:

- *existe un* $(\chi(G))$-coloreo *que satisface la desigualdad por igualdad.*
- *para todo $v \in V \backslash P_k$ existe un conjunto independiente de medida $\lceil k/2 \rceil + 1$ en $G[P_k \cup \{v\}]$,*

entonces define una faceta de $\mathcal{CP}$.

5.5.5. Desigualdades en la Proyección

Si $G' = G[V']$ con $V' \subset V$, sabemos que

$$\sum_{v \in V'} x_{vj_0} + \sum_{j=n-\alpha(G')+1}^{n} \sum_{v \in V} x_{vj} \leq \alpha(G') w_{j_0} + w_{n-\alpha(G')+1}$$

es una cara de $\mathcal{CP}$ y bajo ciertas condiciones resulta faceta.

Como $\chi(G) \leq n - \alpha(G) + 1 \leq n - \alpha(G')$, entonces todo coloreo óptimo satisface $w_{n-\alpha(G')+1} = 0$.

Sea $Proy(\mathcal{CP})$ la proyección de $\mathcal{CP}$ sobre el subespacio $w_{n-\alpha(G')+1} = 0$. Como las soluciones óptimas están en $Proy(\mathcal{CP})$, es interesante analizar la propiedades de las facetas de $\mathcal{CP}$ en la proyección.

Si para un cierto G' la desigualdad es faceta de $\mathcal{CP}$, esto no implica que también lo sea de $Proy(\mathcal{CP})$. Por ejemplo, en el caso de *caminos* pares, la cara determinada por la desigualdad

$$\sum_{i=1}^{2k} x_{v_i j_0} \leq k w_{j_0}$$

está contenida en el subespacio

$$x_{v_1 j_0} = \sum_{j=1, j \neq j_0}^{n-k} x_{v_2 j}$$

Una situación similar ocurre con los *caminos* impares y el subespacio $x_{v_1 j_0} = 0$.

Si bien pudimos determinar condiciones suficientes sobre G' para que la desigualdad basada en el número independiente fuera faceta de $\mathcal{CP}$, no logramos una caracterización general para la proyección.

A continuación presentamos el caso particular de un *agujero*.

Proposición 5.14 *Sea* $C_{2k+1} = \{v_1, \ldots, v_{2k+1}\}$ *un* agujero *de tamaño* $2k+1$ *de* G *y* $j_0 \leq n-k$. *La desigualdad*

$$\sum_{i=1}^{2k+1} x_{v_i j_0} \leq k w_{j_0}$$

define una faceta de $Proy(\mathcal{CP}) = \mathcal{CP} \cap \{w_j = 0 \text{ para } j = n-k+1, \ldots, n\}$ *si se satisfacen las siguientes condiciones:*

- *algún* $(\chi(G))$*-coloreo satisface la igualdad*
- *para todo* $v \in V \backslash C_{2k+1}$ *existe un conjunto independiente de medida* $k+1$ *en* $G[C_{2k+1} \cup \{v\}]$.

Demostración:

Como C_k es un *agujero* nos garantiza la existencia de un conjunto independiente de tamaño k. Consideremos $j, j' \in \{1, \ldots, n\}$ y C^1 un *(n-k)-coloreo* tal que:

$C^1(v_i) = j$ si i es par
$C^1(v_1) = C^1(v_3) = j'$
$C^1(z) \in \{1, \ldots, n-k\} \backslash \{j, j'\} \ \forall\, z \notin C_{2k+1}$

Si $j = j_0$, C^1 satisface por igualdad la desigualdad. Si $j' = j_0$, C^1 satisface la desigualdad en forma estricta. Por lo tanto, la desigualdad es una cara de $Proy(\mathcal{CP})$. Sea F la cara de $Proy(\mathcal{CP})$ definida por la desigualdad ***Agujero***.

Para el caso particular que $G = G[C_5]$ se puede verificar fácilmente que la desigualdad resulta faceta.

Para estudiar la desigualdad debemos conocer la dimensión de la proyección. Al comienzo de este Capítulo mostramos un sistema minimal de ecuaciones para $Proy(\mathcal{CP})$ para el caso que $\chi(G) \leq n-k-1$. Veamos que nos encontramos en esas condiciones. Si $G = G[C_{2k+1}]$, con $k \geq 3$ entonces $\chi(G) = 3 < 2k+1-k-1 = n-k-1$. Si $V \backslash C_{2k+1} \neq \emptyset$, por las hipótesis establecidas, sabemos que

existe $v \in V \backslash C_{2k+1}$ no adyacente a algún *vértice* de C_{2k+1}, por lo que resulta $\chi(G) \leq n-k-1$. Entonces, según ya demostramos al comienzo del Capítulo, el poliedro $Proy(\mathcal{CP})$ tiene el siguiente sistema minimal de ecuaciones:

$$\begin{aligned}
&\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in V \\
&w_j = 1 \qquad \forall j = 1, \ldots, \chi(G) \\
&w_j = 0 \qquad \forall j = n-k+1, \ldots, n \\
&x_{ij} = 0 \qquad \forall i \in V \qquad \forall j = n-k+1, \ldots, n
\end{aligned}$$

Sea F la cara determinada por la desigualdad. Supongamos que $\lambda^X X + \lambda^W W \leq \lambda_0$ es una desigualdad válida para $Proy(\mathcal{CP})$ tal que $F \subseteq Proy(\mathcal{CP}) \cap \{(X,W) : \lambda^X X + \lambda^W W = \lambda_0\}$. Debemos demostrar que:

1. $\lambda^X_{v_i j} - \lambda^X_{v_i j_0} = \lambda^X_{v_{i'} j} - \lambda^X_{v_{i'} j_0} = \gamma \quad \forall\, v_i, v_{i'} \in C_{2k+1},\ \forall\, j \neq j_0$ y $j \leq n-k$.
2. $\lambda^X_{v_i j} = \lambda^X_{v_i j'} \quad \forall\, j, j' = 1, \ldots, n-k \quad j, j' \neq j_0$
3. $\lambda^X_{vj} = \lambda^X_{vj'} \quad \forall\, v \notin C_{2k+1},\ \forall\, j, j' = 1, \ldots, n-k$
4. si $j_0 \geq \chi(G)+1$ entonces $\lambda^W_{j_0} = k\gamma$
5. $\lambda^W_j = 0$ para $j = \chi(G)+1, \ldots, n-k$ y $j \neq j_0$
6. $\gamma \leq 0$

A continuación probamos cada una de las condiciones.

1. Sean $v_i, v_{i+1} \in C_{2k+1}$ y los siguientes coloreos de G en F:

$C^1 : (n-k) - coloreo$	$C^2 : (n-k) - coloreo$
$C^1(v_j) = j_0$ para $i-j$ par y $j \leq i-2$	$C^2(v_i) = j$
$C^1(v_j) = j_0$ para $j-i$ impar y $j \geq i+3$	$C^2(v_{i+1}) = j_0$
$C^1(v_i) = j_0$	$C^2(z) = C^1(z) \ \forall\, z \neq v_i, v_{i+1}$
$C^1(v_{i+1}) = j$	
$C^1(v_{i-1}) = C^1(v_{i+2})$	
$C^1(z) \in \{1, \ldots, n-k\} \backslash \{j, j_0\}$	
$\forall z$ no considerado previamente	

De la coincidencia de ambos coloreos concluimos que

$$\lambda^X_{v_i j} - \lambda^X_{v_i j_0} = \lambda^X_{v_{i+1} j} - \lambda^X_{v_{i+1} j_0}$$

y por transitividad

$$\lambda^X_{v_i j} - \lambda^X_{v_i j_0} = \lambda^X_{v_{i'} j} - \lambda^X_{v_{i'} j_0} = \gamma$$

2. Sea $v_i \in C_{2k+1}$ y los siguientes coloreos de G en F:

$C^1 : (n-k) - coloreo$	$C^2 : (n-k) - coloreo$
$C^1(v_j) = j_0$ si $\mid i-j \mid$ es impar y $j \leq 2k$	$C^2(v_i) = j'$
$C^1(v_i) = j$	$C^2(z) = C^1(z) \ \forall\, z \neq v_i, v_{i+1}$
$C^1(v_{i+2}) = j'$	
$C^1(v_{i+4}) = j$	
$C^1(z) \in \{1,\ldots,n-k\}\backslash\{j,j',j_0\}$	
$\forall z$ no considerado previamente	

Como C^1 y C^2 sólo difieren en el color asignado a v_i, resulta que

$$\lambda^X_{v_i j} = \lambda^X_{v_i j'}$$

3. Sea $v \in V\backslash C_{2k+1}$. Por hipótesis existe $I \subset \{v\} \cup C_{2k+1}$, conjunto independiente de tamaño $k+1$. Sin pérdida de generalidad, y con el sólo fin de simplificar la notación, vamos a suponer que $v_1, v_3, \ldots, v_{2k-1} \in I$. Podemos afirmar entonces que existe un *(n-k)-coloreo* que asigna el mismo color a todos los *vértices* de I y distinto color a los *vértices* restantes. Es decir que podemos colorear a G con los siguientes coloreos en F:

$C^1 : (n-k) - coloreo$	$C^2 : (n-k) - coloreo$
$C^1(v_i) = j_0 \ \forall\, i$ impar $, i \leq 2k-1$	$C^2(v_i) = j_0 \ \forall\, i$ par
$C^1(v) = j_0$	$C^2(v_i) = C^1(v_{i+1}) \ \forall\, i$ impar $i \neq 1, 2k+1$
$C^1(v_2) = j$	$C^2(v_1) = C^1(v) = j$
$C^1(z) \in \{1,\ldots,n-k\}\backslash\{j,j_0\}$	$C^2(z) = C^1(z)$para otros vértices
$\forall z$ no considerado previamente	

Por las coincidencias de los coloreos y el caso 1, concluimos que $\lambda^X_{vj} = \lambda^X_{vj_0}$.

4. Supongamos que j_0 es tal que $\chi(G) + 1 \leq j_0 \leq n-k$. Sea C^1 un *(j_0-1)-coloreo*. Para i par, sea j_i el color de v_i (estos colores no son necesariamente distintos). Como $j_0 - 1 \leq n-k-1$, podemos afirmar que existen al menos $k+1$ *vértices* que repiten color. A partir de C^1 construimos C^2 un *(j_0)-coloreo* en F donde los *vértices* $v_2, v_4, \ldots, v_{2k}$ estén coloreados con j_0. Como en C^1 hay al menos $k+1$ *vértices* repitiendo color, en el caso que algún v_i, con i par, fuera único en su color, este color puede ser asignado a alguno de los $k+1$ *vértices* que repiten color. De esta manera nos aseguramos que todos los colores con índice menor a j_0 son usados por algún *vértice* en el coloreo C^2. Por los casos anteriores, sabemos que $\lambda_{vj} = \lambda_{vj'}$ para todo $v \in V$ y para todo $j, j' \neq j_0$. Entonces

$$\sum_{ipar} \lambda^X_{v_i j_i} = \sum_{ipar} \lambda^X_{v_i j_0} + \lambda^W_{j_0}$$

Pero $\lambda_{v_i j_i} - \lambda_{v_i j_0} = \gamma$, de donde resulta $\lambda^W_{j_0} = k\gamma$

5. Supongamos que j es tal que $\chi(G)+1 \leq j \leq n-k$, $j \neq j_0$. Sea C^1 un *(j-1)-coloreo.* Por hipótesis, podemos suponer que C^1 está en F. Sabemos que existen al menos $k+1$ *vértices* que repiten color. Uno de estos *vértices* puede ser coloreado con el color j obteniendo así un *(j)-coloreo* en F. De aqui podemos derivar fácilmente que $\lambda_j^W = 0$.

6. Sea C^1 un *(n-k)-coloreo* tal que $C^1(v_{2i-1}) = j_0$ para $i = 1, \ldots, k$, $C^1(v_2) = C^1(v_{2k+1}) = j$ con $j \neq j_0$. A partir de C^1, obtenemos C^2 un *(n-k)-coloreo* intercambiando los colores j_0 y j. Como C^1 está en F se satisface $\lambda_0 = \lambda^X X^{C^1} + \lambda^W W^{C^1}$. C^2 no está en F pero verifica la desigualdad $\lambda_0 \geq \lambda^X X^{C^2} + \lambda^W W^{C^2}$. Por la coincidencias de los dos coloreos, se deduce que

$$0 \leq \sum_{i=1}^{k} (\lambda^X_{v_{2i-1}j_0} - \lambda^X_{v_{2i-1}j}) + (\lambda^X_{v_2 j} - \lambda^X_{v_2 j_0}) + (\lambda^X_{v_{2k+1}j} - \lambda^X_{v_{2k+1}j_0})$$

Como $\lambda^X_{v_i j} - \lambda^X_{v_i j_0} = \gamma$ para todo $i = 1, \ldots, 2k+1$, entonces $\gamma \leq 0$.

□

En el caso de las *cliques* y complementos de *agujero* impar no resulta de tanto interés el resultado pues $\alpha(G') = 1$ y $\alpha(G') = 2$ respectivamente. Sólo por completitud enunciamos la propiedad pero omitimos la demostración.

Proposición 5.15 *Si K es* clique *maximal, la desigualdad Clique*

$$\sum_{p \in K} x_{pj_0} - w_{j_0} \leq 0$$

es faceta de $Proy(\mathcal{CP}) = \mathcal{CP} \cap \{w_n = 0\}$

Proposición 5.16 *Sea $\bar{C}_{2k+1} = \{v_1, \ldots, v_{2k+1}\}$ el complemento de un* agujero *de tamaño $2k+1$. La desigualdad Complemento de Agujero Impar*

$$\sum_{i=1}^{2k+1} x_{v_i j_0} \leq 2w_{j_0}$$

es faceta de $Proy(\mathcal{CP}) = \mathcal{CP} \cap \{w_j = 0$ para $j = n-1, n\}$ si se satisfacen las siguientes condiciones:

- *algún* $(\chi(G))$-coloreo *verifica la desigualdad por igualdad*
- *para todo $v \in V$ existe un conjunto independiente de cardinal* 3 *en $G[C_{2k+1} \cup \{v\}]$*

5.6. Desigualdades Válidas Multicolores

A partir de la combinación de desigualdades válidas se obtienen nuevas desigualdades válidas más débiles. Sin embargo, si éstas son sometidas a un proceso de ajuste de los coeficientes, se obtienen nuevas desigualdades válidas que no son implicadas por aquellas usadas en la combinación. Las desigualdades que presentamos en esta Sección surgen de considerar este procedimiento utilizando las que ya derivamos en Secciones anteriores.

5.6.1. *Cadenas de Cliques*

Sean $K_1, K_2, \ldots, K_r$ *cliques* tales que $K_i \cap K_j = \emptyset$ si $j \neq i-1, i+1$ y $|K_i \cap K_{i+1}| \leq 1$ Sea R el cardinal de $\cup_{j=1}^{r} K_j$. Consideremos $c_1, c_2, \ldots, c_r, c_{j_0}$ colores distintos tales que $c_j \leq c_{j_0} - 1 \leq n-2$ $\forall\, j = 1, \ldots, r$.

Ya sabemos que las siguientes desigualdades *Clique* son válidas

$$\sum_{v \in K_i} x_{vc_i} \leq w_{c_i} \quad \forall\, i = 1, \ldots, r$$

y por otro lado, es trivial que resulta válida la desigualdad

$$\sum_{v \in \cup_{i=1}^{r} K_i} \sum_{j=c_{j_0}}^{n} x_{vj} \leq R w_{c_{j_0}}$$

Si sumamos las $r+1$ desigualdades, obtenemos

$$\sum_{v \in K_1} x_{vc_1} + \sum_{v \in K_2} x_{vc_2} + \ldots + \sum_{v \in K_r} x_{vc_r} + \sum_{v \in \cup_{i=1}^{r} K_i} \sum_{j=c_{j_0}}^{n} x_{vj} \leq \sum_{i=1}^{r} w_{c_i} + R w_{c_{j_0}}$$

Como $c_{j_0} > c_i$ $\forall i = 1, \ldots, r$, podemos disminuir el coeficiente de $w_{c_{j_0}}$ a $R - r$. Obtenemos entonces la desigualdad válida

$$\sum_{v \in K_1} x_{vc_1} + \sum_{v \in K_2} x_{vc_2} + \ldots + \sum_{v \in K_r} x_{vc_r} + \sum_{v \in \cup_{i=1}^{r} K_i} \sum_{j=c_{j_0}}^{n} x_{vj} \leq \sum_{i=1}^{r} w_{c_i} + (R-r) w_{c_{j_0}}$$

que domina a la anterior.

Los *(n)-coloreo* que asignan el color c_2 a un *vértice* de $K_1 \backslash K_2$ satisfacen la desigualdad en forma estricta y por lo tanto no es redundante del sistema de ecuaciones. Si queremos que la desigualdad resulte una cara de la mayor dimensión posible, deben cumplirse algunas condiciones. Por ejemplo:

1. Para que exista algún *(n)-coloreo* que satisfaga la desigualdad por igualdad, es necesario que la cantidad de colores involucrados en la desigualdad sean al menos tantos como la cantidad de *vértices* de la misma. Es decir, $n - R \geq c_{j_0} - r$.

2. Si $c_{j_0} \leq \chi(G)$, entonces ningún *vértice* de $K_1 \backslash K_2$ podrá usar otro color que no sea $c_1, c_{j_0}, \ldots, n$. Toda solución de la cara verifica $x_{vc_2} = 0$ para todo $v \in K_1$. Se necesita entonces que $c_{j_0} \geq \chi(G) + 1$.

3. Debe existir al menos un *$(\chi(G))$-coloreo* que verifique la desigualdad por igualdad.

4. Sea un *vértice* $v \in K_i$ y Col_v el conjunto de colores asociados a v en la desigualdad. Debe existir un *$(c_{j_0}$-1)-coloreo* en la cara tal que v es coloreado con cualquier color que no esté en Col_v. Si así no fuera, toda solución de la cara satisface $x_{vj} = 0$ para $j \notin Col_v$.

Estas condiciones que enumeramos son necesarias, pero hasta el momento no hemos podido establecer hipótesis suficientes que nos aseguren la condición de faceta. Sin embargo, para el caso particular de *cliques* de cardinal 2 que conformen un *camino* obtuvimos el siguiente resultado.

Proposición 5.17 *Sea* $P_k = \{v_1, v_2, \ldots, v_k\}$ *un* camino *de longitud* k. *Consideremos* $c_1, c_2, \ldots, c_{k-1}$ *y* c_{j_0} *colores tales que* $c_i < c_{j_0}$ *para todo* $i = 1, \ldots, k-1$ *y* $\chi(G) + 1 \leq c_{j_0} \leq n-2$. *La desigualdad Camino Multicolor*

$$x_{v_1 c_1} + \sum_{i=2}^{k-1} x_{v_i c_{i-1}} + x_{v_i c_i} + x_{v_k c_{k-1}} + \sum_{j=c_{j_0}}^{n} \sum_{i=1}^{k} x_{v_i j} \leq \sum_{i=1}^{k-1} w_{c_i} + w_{c_{j_0}}$$

define una faceta de $\mathcal{CP}$ *si se satisfacen las siguientes condiciones:*

- $V \backslash P_k$ *no es una* clique,
- *para todo* $v \in \cup_{i=1}^{r} K_i$, *existe un* (c_{j_0}-1)-coloreo *tal que* v *es coloreado con colores* $\notin Col_v$
- *existe al menos un* ($\chi(G)$)-coloreo *que satisface la desigualdad por igualdad.*

Demostración:

Sea F la cara de $\mathcal{CP}$ definida por la desigualdad y $\lambda^X X + \lambda^W W \leq \lambda_0$ una desigualdad válida para $\mathcal{CP}$ tal que $F \subseteq \mathcal{CP} \cap \{(X, W) : \lambda^X X + \lambda^W W = \lambda_0\}$. Tenemos que ver que:

1. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n \quad \forall\, v \in V$ y $j = c_{j_0}, \ldots, n-1$
2. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n \quad \forall\, v \in V \backslash P_k$ $j = 1, \ldots, c_{j_0} - 1$
3. $\lambda^X_{v_i j} = \lambda^X_{v_i n} + \lambda^W_n$ para $j \in Col_{v_i}$
4. $\lambda^X_{v_i j} = \lambda^X_{v_i c_{j_0}} + \lambda^W_{c_{j_0}}$ para $j \notin Col_{v_i}$ y $j \leq c_{j_0} - 1$
5. $\lambda^W_{c_{j_0}} = \lambda^W_{c_j}$ para todo $c_j \geq \chi(G) + 1$
6. $\lambda^W_j = 0$ para todo $j \geq \chi(G) + 1$, $j \neq c_1, \ldots, c_{k-1}, c_{j_0}, n$.
7. $\lambda^W_{c_{j_0}} \leq 0$

A continuación demostramos cada una de las condiciones.

1. Sea j tal que $c_{j_0} \leq j \leq n-1$ y los siguientes coloreos en F:

$C^1 : (n-1) - coloreo$
$C^1(v_1) = j$
$C^1(v_k) = j$
$C^1(v_i) = c_i \ \forall\, i = 2, \ldots, k-1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j, c_2, \ldots, c_{k-1}\}$
$\forall z \notin P_k$

$C^2 : (n) - coloreo$
$C^2(v_1) = n$
$C^2(z) = C^1(z) \ \forall\, z \neq v_1$

Como $\lambda^{X^{C^1}} + \lambda^{W^{C^1}} = \lambda^{X^{C^2}} + \lambda^{W^{C^2}}$ y los colores sólo difieren en la coloración de v_1, deducimos que $\lambda_{v_1 j} = \lambda_{v_1 n} + \lambda_n$. Sea $v \neq v_1$ y consideramos los coloreos

$C^1 : (n) - coloreo$ | $C^2 : (n) - coloreo$
$C^1(v_1) = n$ | $C^2(v_1) = j$
$C^1(v) = j$ | $C^2(v) = n$
$C^1(v_s) = c_{s-1} \ \forall s = 2, \ldots, k \ \ v_s \neq v$ | $C^2(z) = C^1(z) \ \forall z \neq v, v_1$
$C^1(z) \in \{1, \ldots, n\} \backslash \{j, c_1, \ldots, c_{k-1}\}$
$\forall z \notin P_k \cup \{v\}$

Como C^1 y C^2 intercambian los colores de v y v_1, vale la identidad $\lambda_{v_1 n} + \lambda_{vj} = \lambda_{v_1 j} + \lambda_{vn}$. Dada la relación entre los coeficientes correspondientes a v_1 derivada antes, concluimos que $\lambda_{vj} = \lambda_{vn} + \lambda_n$ para todo $v \in V$.

2. Sean $v, v' \in V \backslash P_k$ *vértices* no adyacentes y j tal que $1 \leq j \leq c_{j_0} - 1$. Comenzamos por considerar $j \neq c_1, \ldots, c_{k-1}$. Sean los siguientes coloreos en F:

$C^1 : (n-1) - coloreo$ | $C^2 : (n) - coloreo$
$C^1(v) = j$ | $C^2(v) = n$
$C^1(v') = j$ | $C^2(z) = C^1(z) \ \forall z \neq v$
$C^1(v_1) = n - 1$
$C^1(v_s) = c_{s-1} \ \forall s = 2, \ldots, k$
$C^1(z) \in \{1, \ldots, n-2\} \backslash \{j, c_1, \ldots, c_{k-1}\}$
$\forall z \notin P_k \cup \{v, v'\}$

Por las coincidencias de los coloreos obtenemos que $\lambda_{vj} = \lambda_{vn} + \lambda_n$.

Si $j = c_s$ para algún $s = 1, \ldots, k-1$, construimos los coloreos:

$C^1 : (n-1) - coloreo$ | $C^2 : (n) - coloreo$
$C^1(v) = c_s$ | $C^2(v) = n$
$C^1(v') = c_s$ | $C^2(z) = C^1(z) \ \forall z \neq v$
$C^1(v_s) = n - 1$
$C^1(v_i) = c_i \ \forall i = 1, \ldots, s-1$
$C^1(v_i) = c_i \ \forall i = s+1, \ldots, k-1$
$C^1(v_k) = c_{j_0}$
$C^1(z) \in \{1, \ldots, n-2\} \backslash \{c_{j_0}, c_1, \ldots, c_{k-1}\}$
$\forall z \notin P_k \cup \{v, v'\}$

Por las coincidencias de los coloreos obtenemos que $\lambda_{vc_s} = \lambda_{vn} + \lambda_n$. Para el resto de los *vértices* $u \in V \backslash P_k$ basta intercambiar en un *(n)-coloreo* los colores j y n asignados a u y v respectivamente como hemos hecho para el caso anterior.

3. Sea $v_i \in P_k$, $v, v' \in V \backslash P_k$ *vértices* no adyacentes y $j \in Col_{v_i}$. Sean los coloreos en F:

$$C^1 : (n-1) - coloreo \qquad C^2 : (n) - coloreo$$
$$C^1(v) = c_{j_0} \qquad C^2(v_i) = n$$
$$C^1(v') = c_{j_0} \qquad C^2(v) = C^1(v_i)$$
$$C^1(v_s) = \begin{cases} c_s \ \forall\, s = 1, \ldots, k-1 & \text{si } j = c_i \\ c_{s-1} \ \forall\, s = 2, \ldots, k & \text{si } j = c_{i-1} \end{cases} \qquad C^2(z) = C^1(z) \ \forall\, z \neq v, v_i$$
$$C^1(v_k) = n-1 \text{ si } j = c_i$$
$$C^1(v_1) = n-1 \text{ si } j = c_{i-1}$$
$$C^1(z) \in \{1, \ldots, n-2\} \backslash \{c_{j_0}, c_1, \ldots, c_{k-1}\}$$
$$\forall z \notin P_k \cup \{v, v'\}$$

De la coincidencia de los coloreos y por la propiedades demostradas anteriormente, resulta $\lambda_{v_i j} = \lambda_{vn} + \lambda_n \quad \forall\, j \in Col_{v_i}$.

4. Sea $v_i \in P_k$ y $j \in \{1, \ldots, c_{j_0} - 1\} - Col_{v_i}$. Por hipótesis existe un *(c_{j_0}-1)-coloreo* en F tal que v_i es coloreado con color j. A partir de este coloreo, construimos un *(c_{j_0})-coloreo* tal que v_i es coloreado con c_{j_0}, de donde se deduce que $\lambda_{v_i j} = \lambda_{v c_{j_0}} + \lambda_{c_{j_0}}$.

5. Por hipótesis existe *($\chi(G)$)-coloreo* en F, por lo tanto existe *(c_j-1)-coloreo* en F con c_j tal que $c_j \geq \chi(G) + 1$. Podemos suponer que en dicho coloreo v_j es coloreado con c donde $c \notin Col_{v_j}$ y que existe algún *vértice* $u \in V \backslash P_k$ que no es único en su color (esto se logra con una apropiada permutación de los colores). Notemos con c' al color de u. Si ahora cambiamos el color de v_j a c_j y el color de u a c en el caso que v_j fuera único en su color, obtenemos $\lambda_{v_j c} + \lambda_{uc'} = \lambda_{v_j c_j} + \lambda_{uc} + \lambda_{c_j}$.

   Pero sabemos que $\lambda_{v_j c} = \lambda_{v_j c_j} + \lambda_{c_{j_0}}$ y $\lambda_{uc'} = \lambda_{uc}$. Concluimos entonces que $\lambda_{c_j} = \lambda_{c_{j_0}}$.

6. Sea un *(j-1)-coloreo* en F con j tal que $\chi(G) + 1 \leq j \leq n-1$ y $j \notin \{c_1, c_2, \ldots, c_{k-1}, c_{j_0}\}$ Podemos suponer que existen $v, v' \in V - P_k$ *vértices* no adyacentes que repiten color. Cambiándole a v su color por el color j, y teniendo en cuenta que $\lambda_{vj} = \lambda_{vj'}$ para todo $j, j' \leq n-1$, concluimos que $\lambda_j = 0$.

7. Sea C^1 un *(n)-coloreo* tal que $C^1(v_i) = c_i$ para $i = 1, \ldots, k-1$ y $C^1(v_k) = c_{j_0}$. En base a C^1, construimos C^2 un *(n)-coloreo* intercambienao los colores c_{j_0} y c_1. Como C^1 está en F, se satisface $\lambda^X X^{C^1} + \lambda^W W^{C^1} = \lambda_0$. C^2 no está en F pero verifica $\lambda^X X^{C^2} + \lambda^W W^{C^2} \leq \lambda_0$. Entonces $\lambda^X X^{C^1} + \lambda^W W^{C^1} - \lambda^X X^{C^2} - \lambda^W W^{C^2} \geq 0$. De las coincidencias de los dos coloreos derivamos la desigualdad

$$\lambda^X_{v_1 c_1} + \lambda^X_{v_k c_{j_0}} - \lambda^X_{v_1 c_{j_0}} - \lambda^X_{v_k c_1} \geq 0$$

   Pero $\lambda^X_{v_1 c_1} = \lambda^X_{v_1 c_{j_0}}$ y $\lambda^X_{v_k c_{j_0}} - \lambda^X_{v_k c_1} = -\lambda^W_{c_{j_0}}$. Entonces $\lambda^W_{c_{j_0}} \leq 0$

□

5.6.2. *Clique Multicolor*

Sea $K = \{v_1, \ldots, v_p\}$ una *clique* de tamaño p, k tal que $p \leq k \leq n-1$ y $Col = \{j_1, j_2, \ldots, j_{p-1}\} \subset \{1, \ldots, k-1\}$. La desigualdad válida

$$\sum_{i=1}^{p} \sum_{j \in Col} x_{v_i j} \leq \sum_{j \in Col} w_j$$

se obtiene de sumar $p-1$ desigualdades ***Clique***, cada una correspondiente a un color en Col. Además es válido que $\sum_{i=1}^{p} \sum_{j=k}^{n} x_{v_i j} \leq p w_k$. Si sumamos las dos desigualdades y tenemos en cuenta que cualquier coloreo de G utiliza p colores para colorear una *clique* de tamaño p, resulta válida la desigualdad

$$\sum_{i=1}^{p} \sum_{j=k}^{n} x_{v_i j} + \sum_{i=1}^{p} \sum_{j \in Col} x_{v_i j} \leq w_k + \sum_{j \in Col} w_j$$

Cualquier *(n)-coloreo* que asigne los colores $k, j_1, \ldots, j_{p-1}$ a los *vértices* de la *clique* la satisface por igualdad. Si $k = p$ entonces $k \leq \chi(G)$ y la desigualdad es combinación lineal de las ecuaciones del sistema minimal de $\mathcal{CP}$. Si $p < k$, cualquier coloreo que asigne a algún *vértice* de la *clique* un color $\in \{1, \ldots, k-1\} \backslash Col$ satisface estrictamente la desigualdad. Por lo tanto, la desigualdad es cara de $\mathcal{CP}$. Veamos en que caso resulta ser faceta de $\mathcal{CP}$.

Proposición 5.18 *Sea $K \subset V$ una* clique *tal que $V \backslash K$ no es* clique *y $\chi(G) + 1 \leq k \leq n-2$. La desigualdad* ***p-color Clique***

$$\sum_{i=1}^{p} \sum_{j=k}^{n} x_{v_i j} + \sum_{i=1}^{p} \sum_{j \in Col} x_{v_i j} \leq w_k + \sum_{j \in Col} w_j$$

es faceta de $\mathcal{CP}$.

Demostración:

Se F la cara de $\mathcal{CP}$ definida por la desigualdad y $\lambda^X X + \lambda^W W \leq \lambda_0$ una desigualdad válida para $\mathcal{CP}$ tal que $F \subseteq \mathcal{CP} \cap \{(X, W) : \lambda^X X + \lambda^W W = \lambda_0\}$. Tenemos que ver que:

1. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n \quad \forall\, v \notin K$ y $\forall\, j = 1, \ldots, n-1$
2. $\lambda^X_{v_i j} = \lambda^X_{v_i n} + \lambda^W_n$ para $j \in Col \cup \{k, \ldots, n-1\}$ y $\forall\, v_i \in K$
3. $\lambda^X_{v_i j} = \lambda^X_{v_i k} + \lambda^W_k$ para $j \notin Col \cup \{k, \ldots, n\}$ y $\forall\, v_i \in K$
4. $\lambda^W_k = \lambda^W_j$ para todo $j_i \geq \chi(G) + 1$
5. $\lambda^W_j = 0$ para todo $j \geq \chi(G) + 1,\ j \notin Col \cup \{k\}$
6. $\lambda^W_k \leq 0$

A continuación probamos cada uno de los casos.

1. *a*) Comenzaremos viendo el caso en que $j \notin Col$. Como por hipótesis $V\backslash K$ no es *clique*, existen $u, u' \in V\backslash K$ *vértices* no adyacentes. Consideremos los coloreos en F:

$$C^1 : (n-1) - coloreo \qquad C^2 : (n) - coloreo$$
$$C^1(u) = j \qquad C^2(u) = n$$
$$C^1(u') = j \qquad C^2(z) = C^1(z) \ \forall\, z \neq u$$
$$C^1(v_i) = j_i \text{ para } i = 1, \ldots, p-1$$
$$C^1(v_p) = \begin{cases} k & \text{si } j \neq k \\ k+1 & \text{si } j = k \end{cases}$$
$$C^1(z) \in \{1, \ldots, n-1\}\backslash\{j, j_1, \ldots, j_{p-1}\}$$
$$\forall z \notin K \cup \{u, u'\}$$

Como $\lambda^{X^{C^1}} + \lambda^{W^{C^1}} = \lambda^{X^{C^2}} + \lambda^{W^{C^2}}$ y los colores sólo difieren en la coloración de u, resulta $\lambda^X_{uj} = \lambda^X_{un} + \lambda^W_n$. Sea ahora $v \in V\backslash K$, $v \neq u$ y los siguientes coloreos en F:

$$C^1 : (n) - coloreo \qquad C^2 : (n) - coloreo$$
$$C^1(v) = j \qquad C^2(v) = n$$
$$C^1(u) = n \qquad C^2(u) = j$$
$$C^1(v_i) = j_i \qquad C^2(z) = C^1(z) \ \forall\, z \neq u, v$$
$$C^1(v_p) = \begin{cases} k & \text{si } j \neq k \\ k+1 & \text{si } j = k \end{cases}$$

Por la coincidencia de los coloreos y lo demostrado anteriormente se satisface que $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n$.

b) Sea $j \in Col$. Nuevamente consideramos en primer lugar a u, u' los *vértices* no adyacentes de $V\backslash K$ y los coloreos en F:

$$C^1 : (n-1) - coloreo \qquad C^2 : (n) - coloreo$$
$$C^1(u) = j_i \qquad C^2(u) = n$$
$$C^1(u') = j_i \qquad C^2(z) = C^1(z) \ \forall\, z \neq u$$
$$C^1(v_s) = j_s \text{ para } s = 1, \ldots, p-1 \quad s \neq i$$
$$C^1(v_i) = n-2$$
$$C^1(v_p) = n-1$$
$$C^1(z) \in \{1, \ldots, n-3\}\backslash\{j_1, \ldots, j_{p-1}\}$$
$$\forall z \notin K \cup \{u, u'\}$$

De aqui resulta $\lambda^X_{uj_i} = \lambda^X_{un} + \lambda^W_n$. Para cualquier $v \in V\backslash K$, $v \neq u$, coloreamos a G con

los siguientes coloreos en F:

$C^1 : (n) - coloreo$ $\quad$ $C^2 : (n) - coloreo$

$C^1(v) = j_i$ $\quad$ $C^2(v) = n$

$C^1(u) = n$ $\quad$ $C^2(u) = j_i$

$C^1(v_s) = j_s$ para $s = 1, \ldots, p-1 \quad s \neq i$ $\quad$ $C^2(z) = C^1(z) \ \forall\, z \neq v, u$

$C^1(v_i) = n - 2$

$C^1(v_p) = n - 1$

Con los mismos argumentos que usamos antes, resulta $\lambda^X_{vj_i} = \lambda^X_{vn} + \lambda^W_n$

2. Sean $v_i \in K$ y $v \in V \backslash K$. Existe un *(n)-coloreo* en F que colorea a v_i con algún color de $Col \cup \{k, \ldots, n-1\}$ y a v con color n. Haciendo un intercambio de los colores de v_i y de v, fácilmente obtenemos $\lambda^X_{v_ij} = \lambda^X_{v_in} + \lambda^W_n$ para todo $j \in Col \cup \{k, \ldots, n-1\}$.

3. Se $j \notin Col$. Consideremos un *(k-1)-coloreo* en F que coloree a v_i con el color j y el resto de los *vértices* de K con algún color de Col. Como $k - 1 \leq n - 3$, existen al menos dos *vértices* coloreados con el mismo color. A partir de este coloreo, podemos construirnos un *(k)-coloreo* en F cambiando el color de v_i a k. En el caso que v_i fuera el único *vértice* con color j, a cualquier *vértice* que comparta color con otro le cambiamos el color al j. De esta manera, utilizando los resultados previos, obtenemos $\lambda^X_{v_ij} = \lambda^X_{v_ik} + \lambda^W_k$ para todo $j \notin Col \cup \{k, \ldots, n\}$.

4. Sea $j_i \in Col$ tal que $j_i \geq \chi(G) + 1$. Consideramos un *$(j_i - 1)$-coloreo* en el cual los colores de Col que tengan índice menor a j_i son usados en el coloreo de K. Este coloreo está en F. Podemos afirmar que existe al menos un *vértice* en K coloreado con algún color que no está en Col. Supongamos que v_p está coloreado con $j \notin Col$. Por otro lado, como $j_i - 1 \leq n - 4$, existe $u \in V \backslash K$ que comparte color con algún otro *vértice*. Basado en este coloreo, construimos un *(j_i)-coloreo*en F cambiando el color de v_p a j_i. Si v_i era el único coloreado con j, cambiamos el color de u a j. Usando los casos considerados anteriormente, es sencillo llegar a que $\lambda^W_{j_i} = \lambda^W_k$ para todo $j_i \geq \chi(G) + 1$.

5. Sea $j \notin Col \cup \{k\}$ tal que $j \geq \chi(G) + 1$. Consideramos un *(j-1)-coloreo* en F en el cual los colores de Col que tengan índice menor a j son usados en el coloreo de K. Como $j \leq n - 1$, existe $u \in V \backslash K$ que comparte color con algún otro *vértice*. Construimos un *(j)-coloreo* en F cambiando el color de u a j. Por lo demostrado en $1a$ y $1b$, es sencillo llegar a que $\lambda^W_j = 0$.

6. Sea C^1 un *(n)-coloreo* tal que $C^1(v_i) = j_i$, para $i = 1, \ldots, p-1$ y $C^1(v_p) = k$. Como $p < k$, existe $v \in V \backslash K$ tal que $C^1(v) = c < k$. Intercambiando los colores de v y v_p obtenemos C^2 un *(n)-coloreo*. Como C^1 está en F, se satisface $\lambda^X X^{C^1} + \lambda^W W^{C^1} = \lambda_0$. C^2 no está en F pero verifica $\lambda^X X^{C^2} + \lambda^W W^{C^2} \leq \lambda_0$. Entonces $\lambda^X X^{C^1} + \lambda^W W^{C^1} - \lambda^X X^{C^2} - \lambda^W W^{C^2} \geq 0$ De las coincidencias de los dos coloreos derivamos la desigualdad

$$\lambda^X_{v_pk} + \lambda^X_{vc} - \lambda^X_{v_pc} - \lambda^X_{vk} \geq 0$$

Pero $\lambda^X_{vc} = \lambda^X_{vk}$ y $\lambda^X_{v_pk} - \lambda^X_{v_pc} = -\lambda^W_k$, entonces $\lambda^W_k \leq 0$.

□

5.6.3. *Collar Multicolor*

Supongamos que $C_k = \{v_1, \ldots, v_k\}$ es un *ciclo* de tamaño k y $\{j_1, \ldots, j_{k-1}\}$ un conjunto de colores tal que $j_1 > j_i$ para todo $i = 2, \ldots, k-1$. Sabemos que las siguientes $k-2$ desigualdades son válidas:

$$x_{v_2 j_2} + x_{v_3 j_2} \leq w_{j_2}$$

$$x_{v_3 j_3} + x_{v_4 j_3} \leq w_{j_3}$$

$$x_{v_i j_i} + x_{v_{i+1} j_i} \leq w_{j_i}$$

$$x_{v_{k-1} j_{k-1}} + x_{v_k j_{k-1}} \leq w_{k-1}$$

y además

$$x_{v_k j_1} + x_{v_1 j_1} + x_{v_2 j_1} \leq 2w_{j_1}$$

Sumando todas las desigualdades, resulta la desigualdad válida

$$x_{v_1 j_1} + \sum_{i=2}^{k-1} (x_{v_i j_{i-1}} + x_{v_i j_i}) + x_{v_k j_{k-1}} + x_{v_k j_1} \leq 2w_{j_1} + \sum_{i=2}^{k-1} w_{j_i}$$

Esta desigualdad puede ser mejorada reduciendo el coeficiente de w_{j_1} a 1. Veamos porqué. Para los coloreos que no usan el color j_1 vale que $w_{j_1} = 0$ y por lo tanto el coeficiente de la desigualdad no es significativo. Para los coloreos que utilizan el color j_1 en a lo sumo uno de los *vértices* v_1, v_2 ó v_k, es válido que $x_{v_k j_1} + x_{v_1 j_1} + x_{v_2 j_1} \leq w_{j_1}$. Entonces el coeficiente puede ser reducido a 1. Si en cambio v_2 y v_k son coloreados con el color j_1, el *vértice* v_1 no podrá usar j_1 con lo cual $x_{v_1 j_1} = 0$. Quedan entonces $k-3$ *vértices* para los que estamos considerando las variables $x_{v_i j_{i-1}}$ y $x_{v_i j_i}$, cuya suma total será a lo sumo $k-3$. Pero $w_{j_1} = 1$ implica $w_{j_i} = 1$ para todo $i = 2, \ldots, k-1$, por lo que $\sum_{i=2}^{k-1} w_{j_i} = k-2$ compensa la reducción del coeficiente de w_{j_1}.

Entonces, resulta válida la desigualdad

$$x_{v_1 j_1} + \sum_{i=2}^{k-1} (x_{v_i j_{i-1}} + x_{v_i j_i}) + x_{v_k j_{k-1}} + x_{v_k j_1} \leq w_{j_1} + \sum_{i=2}^{k-1} w_{j_i}$$

Los *(n)-coloreo* que colorean a v_i con color j_i para $i = 1, \ldots, k-1$ satisfacen la desigualdad por igualdad. Los *(n)-coloreo* que no colorean a v_1 con j_1, ni a v_2 con j_1 ó j_2, satisfacen estrictamente la desigualdad. Por lo tanto, la desigualdad es una cara de $\mathcal{CP}$. Veamos ahora en que condiciones define faceta de $\mathcal{CP}$.

Proposición 5.19 *Sea $C_k = \{v_1, \ldots, v_k\}$ un* agujero *de tamaño k y $\{j_1, \ldots, j_{k-1}\} \subseteq \{1, \ldots, n-1\}$, $j_1 > j_i$ para $i = 2, \ldots, k-1$. La desigualdad Collar Multicolor*

$$x_{v_1 j_1} + \sum_{i=2}^{k-1} (x_{v_i j_{i-1}} + x_{v_i j_i}) + x_{v_k j_{k-1}} + x_{v_k j_1} \leq w_{j_1} + \sum_{i=2}^{k-1} w_{j_i}$$

define una faceta de $\mathcal{CP}$ si y sólo si

- *para todo* $v \in V \backslash C_k$, *al menos uno de los* vértices v_1, v_2 *o* v_k *es no adyacente a* v.
- *algún* $(\chi(G))$*-coloreo verifica la desigualdad por igualdad.*

Demostración:

La primera condición resulta necesaria. Si $v \in V \backslash C_k$ es adyacente a v_1, v_2 y v_k, los coloreos que satisfacen por igualdad la desigualdad tienen la propiedad que $x_{vj_1} = 0$ y por lo tanto la cara no es faceta. La segunda condición asegura que la cara no está incluída en el hiperplano $w_{\chi(G)+1} = 1$.

Sea F la cara de $\mathcal{CP}$ definida por la desigualdad *Collar Multicolor*. Supongamos que $\lambda^X X + \lambda^W W \leq \lambda_0$ es una desigualdad válida para $\mathcal{CP}$ tal que $F \subseteq \mathcal{CP}$ $\{(X, W) : \lambda^X X + \lambda^W W = \lambda_0\}$. Sean $Col_1 = \{j_1\}$, $Col_k = \{j_{k-1}, j_1\}$ y $Col_i = \{j_{i-1}, j_i\}$ para $i = 2, \ldots k-1$. Debemos demostrar que:

1. $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n$, $v \notin C_k \quad \forall\, j = 1, \ldots, n-1$
2. $\lambda^X_{v_i j} = \lambda^X_{v_i n} + \lambda^W_n \quad \forall\, i = 1, \ldots, k$ y $\quad \forall\, j \notin Col_i$
3. $\lambda^X_{v_i c_i} - \lambda^X_{v_i n} = \lambda^X_{v_{i'} c_{i'}} - \lambda^X_{v_{i'} n} \quad \forall\, c_i \in Col_i$ y $c_{i'} \in Col_{i'}$
4. si $j_i \geq \chi(G) + 1$ entonces $\lambda^X_{v_i j_i} = \lambda^X_{v_i j} - \lambda^W_{j_i} \quad \forall\, j = 1, \ldots, n-1$ y $j \notin Col_i$
5. $\lambda^W_j = 0$ para $j = \chi(G) + 1, \ldots, n-1$ y $j \neq j_1$
6. $\lambda^X_{v_i c_i} - \lambda^X_{v_i j} = \gamma \geq 0$ para $j \notin Col_i$

Comenzamos la demostración considerando casos particulares de los coeficientes λ sobre los *vértices* del *agujero*. Estos nos permite luego analizar las condiciones generales. En cada caso especificamos los coloreos C^1 y C^2 en F de los cuales derivamos las propiedades.

(i)

$C^1 : (n-1) - coloreo$	$C^2 : (n-1) - coloreo$
$C^1(v_1) = j_2$	$C^2(v_1) = j_3$
$C^1(v_s) = j_{s-1} \quad \forall\, s = 2, \ldots, k$	$C^2(z) = C^1(z) \quad \forall\, z \neq v_1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\}$	
$\forall z \notin C_k$	

Dado que C^1 y C^2 sólo difieren en el color de v_1, concluimos que $\lambda^X_{v_1 j_2} = \lambda^X_{v_1 j_3}$

(ii)

$C^1 : (n-1) - coloreo$	$C^2 : (n-1) - coloreo$
$C^1(v_1) = j_i$ con $i \neq 1, 2$	$C^2(v_1) = j_{i'}$ con $i' \neq i, 1, 2$
$C^1(v_s) = j_s \quad \forall\, s = 2, \ldots, k-1$	$C^2(z) = C^1(z) \quad \forall\, z \neq v_1$
$C^1(v_k) = j_1$	
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\}$	
$\forall z \notin C_k$	

Se deduce fácilmente que $\lambda^X_{v_1 j_i} = \lambda^X_{v_1 j_{i'}}$ $\forall\, i, i' \in \{3, \ldots, k-1\}$

(iii)

$C^1 : (n-1) - coloreo$
$C^1(v_1) = j_2$
$C^1(v_2) = C^1(v_k) = j_1$
$C^1(v_s) = j_s \;\; \forall\, s = 3, \ldots, k-1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\}$
$\forall z \notin C_k$

$C^2 : (n-1) - coloreo$
$C^2(v_1) = j_3$
$C^2(v_2) = j_2$
$C^2(z) = C^1(z) \;\; \forall\, z \neq v_1, v_2$

Utilizando lo demostrado en (i), concluimos que $\lambda^X_{v_2 j_1} = \lambda^X_{v_2 j_2}$.

(iv)

$C^1 : (n-1) - coloreo$
$C^1(v_1) = j_{k-1}$
$C^1(v_k) = j_1$
$C^1(v_s) = j_{s-1} \;\; \forall\, s = 2, \ldots, k-1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\}$
$\forall z \notin C_k$

$C^2 : (n-1) - coloreo$
$C^2(v_1) = j_{k-2}$
$C^2(v_k) = j_{k-1}$
$C^2(z) = C^1(z) \;\; \forall\, z \neq v_1, v_k$

Utilizando el caso (ii), se deduce que $\lambda^X_{v_k j_1} = \lambda^X_{v_k j_{k-1}}$

(v) Sea $i = 3, \ldots, k-1$.

$C^1 : (n-1) - coloreo$
$C^1(v_1) = j_i$
$C^1(v_i) = j_{i-1}$
$C^1(v_s) = j_{s-1} \;\; \forall\, s = 2, \ldots, i-1$
$C^1(v_s) = j_s \;\; \forall\, s = i+1, \ldots, k-1$
$C^1(v_k) = j_1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\}$
$\forall z \notin C_k$

$C^2 : (n-1) - coloreo$
$C^2(v_1) = j_{i-1}$
$C^2(v_i) = j_i$
$C^2(z) = C^1(z) \;\; \forall\, z \neq v_1, v_i$

Nuevamente a partir del caso (ii), derivamos $\lambda^X_{v_i j_{i-1}} = \lambda^X_{v_i j_i}$, $\forall\, i = 3, \ldots, k-1$

Veamos ahora la demostración de las condiciones generales.

1. *a)* Sean $v \notin C_k$ y $j_i \neq j_1$. C^1 y C^2 los siguientes coloreos de G en F:

$C^1 : (n-1) - coloreo$
$C^1(v) = j_i$
$C^1(v_s) = j_{s-1} \ \forall\, s = 2, \ldots, i$
$C^1(v_s) = j_s \ \forall\, s = i+1, \ldots, k-1$
$C^1(v_k) = j_1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\}$
$\forall z \neq v, z \notin C_k$

$C^2 : (n) - coloreo$
$C^2(v) = n$
$C^2(v_s) = j_{s-1} \ \forall\, s = 2, \ldots, k$
$C^2(z) = C^1(z) \ \forall\, z \neq v, z \notin C_k$

Los coloreos difieren en los colores asignados a los *vértices* del *agujero* y a v. De los casos particulares demostrados anteriormente, concluimos que $\lambda^X_{vj_i} = \lambda^X_{vn} + \lambda^W_n$, $\forall\, i = 2, \ldots, k-1$.

b) Sea $v \notin C_k$. Si v_1 y v_k son adyacentes a v, sabemos que v_2 no es adyacente a v. Por lo tanto, v y v_2 pueden ser coloreados con el mismo color. En este caso consideramos el siguiente coloreo de G en F:
$C^1 : (n-1) - coloreo$
$C^1(v) = j_1$
$C^1(v_2) = j_1$
$C^1(v_s) = j_{s-1} \ \forall\, s = 3, \ldots, k$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\} \ \forall\, z \neq v, z \notin C_k$

Si v_1 o v_k no son adyacentes a v, alguno de ellos puede compartir color con v. Llamemos r al que se encuentre en estas condiciones y coloreamos a G del siguiente modo:
$C^1 : (n-1) - coloreo$
$C^1(v) = j_1$
$C^1(r) = j_1$
$C^1(v_s) = j_s \ \forall\, s = 2, \ldots, k-1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\} \ \forall\, z \neq v, z \notin C_k$

En cualquiera de los casos, tengamos en cuenta el coloreo C^2 en F:
$C^2 : (n) - coloreo$
$C^2(v) = n$
$C^2(z) = C^1(z) \ \forall\, z \neq v$

Debido a que la única diferencia entre C^1 y C^2 es el color de v, se concluye que $\lambda^X_{vj_1} = \lambda^X_{vn} + \lambda^W_n$.

c) Sean $v \notin C_k$ y $j \notin \{j_1, \ldots, j_{k-1}, n\}$. Por hipótesis, existe $r \in \{v_1, v_2, v_k\}$ tal que r no es adyacente a v y por lo tanto puede ser coloreado con el mismo color que v. De acuerdo a quien sea r, consideramos los siguientes coloreos de G en F:
Para $r = v_1$
$C^1 : (n-1) - coloreo$

$C^1(v) = j$
$C^1(v_1) = j$
$C^1(v_s) = j_{s-1} \ \forall\, s = 2, \ldots, k$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\} \ \forall\, z \neq v, z \notin C_k$

Para $r = v_2$
$C^1 : (n-1) - coloreo$
$C^1(v) = j$
$C^1(v_2) = j$
$C^1(v_1) = j_1$
$C^1(v_s) = j_{s-1} \ \forall\, s = 3, \ldots, k$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\} \ \forall\, z \neq v, z \notin C_k$

Para $r = v_k$
$C^1(v) = j$
$C^1(v_k) = j$
$C^1(v_s) = j_s \ \forall\, s = 1, \ldots, k-1$
$C^1(z) \in \{1, \ldots, n-1\} \backslash \{j_1, \ldots, j_{k-1}\} \ \forall\, z \neq v, z \notin C_k$

En cualquiera de los casos, tengamos en cuenta el coloreo C^2 en F:

$C^2 : (n) - coloreo$
$C^2(v) = n$
$C^2(z) = C^1(z) \ \forall\, z \neq v$

La única diferencia entre C^1 y C^2 es el color de v, los que nos permite concluir que $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n$.

2. Sean $v \notin C_k$, $v_i \in C_k$ y $j \leq n-1$, $j \notin Col_i$. Consideramos los siguientes coloreos de G en F:

$C^1 : (n) - coloreo$
$C^1(v_i) = j$
$C^1(v) = n$
$C^1(v_s) = j_s \ \forall\, s = 1, \ldots, i-1$
$C^1(v_s) = j_{s-1} \ \forall\, s = i+1, \ldots, k$
$C^1(z) \in \{1, \ldots, n\} \backslash \{j_1, \ldots, j_{k-1}, j\}$
$\forall z \neq v, z \notin C_k$

$C^2 : (n) - coloreo$
$C^2(v_i) = n$
$C^2(v) = j$
$C^2(z) = C^1(z) \ \forall\, z \neq v, v_i$

Los coloreos difieren en la coloración de v y v_i. Habiendo demostrado (a), concluimos que $\lambda^X_{v_i j} = \lambda^X_{v_i n} + \lambda^W_n$.

3. Sea $i = 1, \ldots, k-1$. Consideremos los siguientes coloreos en F:

$$\begin{array}{ll} C^1 : (n) - coloreo & C^2 : (n) - coloreo \\ C^1(v_i) = j_i & C^2(v_i) = n \\ C^1(v_{i+1}) = n & C^2(v_{i+1}) = j_i \\ C^1(v_s) = j_s \ \forall\, s = 1, \ldots, i-1 & C^2(z) = C^1(z) \ \forall\, z \neq v_i, v_{i+1} \\ C^1(v_s) = j_{s-1} \ \forall\, s = i+2, \ldots, k & \\ C^1(z) \in \{1, \ldots, n\} \backslash \{j_1, \ldots, j_{k-1}\} & \\ \forall z \notin C_k & \end{array}$$

Es trivial llegar a la identidad $\lambda^X_{v_i j_i} - \lambda^X_{v_i n} = \lambda^X_{v_{i+1} j_i} - \lambda^X_{v_{i+1} n}$.

4. Sea $j_i = \chi(G)+1, \ldots, n-1$. Por hipótesis existe un *$(\chi(G))$-coloreo* en F, por lo tanto también existe un *$(j_i - 1)$-coloreo* en F. Para cualquier *$(j_i - 1)$-coloreo* en F podemos afirmar que ningún *vértice* está coloreado con j_i y que existe $v \neq v_i$ cuyo color es compartido con algún otro *vértice*. A partir de este coloreo se puede construir un *(j_i)-coloreo* en F, coloreando a v_i con j_i y en caso de ser necesario cambiando el color de v al color que tenía antes v_i. De las coincidencias de ambos coloreos, deducimos que $\lambda^X_{v_i j} = \lambda^X_{v_i j_i} + \lambda^W_{j_i}$.

5. Sea $j = \chi(G)+1, \ldots, n-1$, $j \neq j_1, \ldots, j_{k-1}$. Para cualquier *(j-1)-coloreo* en F, existe $v \notin C_k$ que comparte color con algún otro *vértice*, del cual podemos derivar el *(j)-coloreo* en F cuya única diferencia con el anterior sea colorear a v con j. De estas similitudes y por los casos anteriores, derivamos trivialmente que $\lambda^W_j = 0$.

6. Como $\lambda^X_{v_i c_i} - \lambda^X_{v_i n}$ es una constante para todo $i = 1, \ldots, k$ y vale la condición 2, podemos afirmar que $\lambda^X_{v_i c_i} - \lambda^X_{v_i j} = \gamma$ para $j \notin Col_i$. Falta ver que $\gamma \geq 0$. Sea C^1 un *(n)-coloreo* en F tal que $C^1(v_i) = j_i$ para $i = 1, \ldots, k-1$ y $C^1(v_k) = j$. Intercambiando los colores de v_2 y v_k obtenemos C^2 un *(n)-coloreo* que no está en F. Se satisface que

$$\lambda^X X^{C^2} + \lambda^W W^{C^2} \leq \lambda_0 = \lambda^X X^{C^1} + \lambda^W W^{C^1}$$

Entonces $\lambda^X X^{C^1} + \lambda^W W^{C^1} - \lambda^X X^{C^2} - \lambda^W W^{C^2} \geq 0$ De las coincidencias de los dos coloreos derivamos la desigualdad

$$\lambda^X_{v_2 j_2} + \lambda^X_{v_k j} - \lambda^X_{v_2 j} - \lambda^X_{v_k j_2} \geq 0$$

Pero como $\lambda^X_{v_2 j_2} - \lambda^X_{v_2 j} = \gamma$ y $\lambda^X_{v_k j} = \lambda^X_{v_k j_2}$, resulta que $\gamma \geq 0$.

Concluimos entonces que (λ^X, λ^W) es combinación lineal de las ecuaciones del modelo y de la desigualdad ***Collar Multicolor***. Por lo tanto define una faceta de $\mathcal{CP}$.

□

5.7. El Poliedro $\mathcal{SCP}$

Nuestro trabajo con modelos de programación entera para el *problema de coloreo de grafos* se inició con el estudio de $\mathcal{SCP}$, el poliedro asociado con el modelo clásico. Recordamos que $\mathcal{SCP}$ es

la cápsula convexa de las soluciones de:

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= 1 && \forall\ i \in V \\ x_{ij} + x_{kj} &\leq w_j && \text{si } [i,k] \in E\ \forall j = 1,\ldots,n \\ x_{ij}, w_j &\in \{0,1\} && \forall i \in V\ \forall j = 1,\ldots,n \end{aligned}$$

Las desigualdades válidas de $\mathcal{SCP}$ que caracterizamos no tuvieron un gran impacto en mejorar la performance de un algoritmo *Branch-and-Cut* que usa $\mathcal{RSCP}$, la relajación lineal de $\mathcal{SCP}$. Sin embargo, el estudio de $\mathcal{SCP}$ facilitó parcialmente la derivación de alguno de los resultados para $\mathcal{CP}$. Para que se vea reflejada la relación entre la estructura poliedral de $\mathcal{SCP}$ y de $\mathcal{CP}$, haremos un breve resumen de lo principales resultados, omitiendo sus demostraciones que pueden ser consultadas en [18].

Proposición 5.20 *Las ecuaciones* $\sum_{j=1}^{n} x_{ij} = 1$ *par todo* $i \in V$ *definen un sistema minimal para* $\mathcal{SCP}$ *y por lo tanto* $dim(\mathcal{SCP}) = n^2$.

De las restricciones originales del modelo resultan ser facetas las triviales de positividad y cota superior. Observar que esta propiedad no se mantuvo completamente para el poliedro $\mathcal{CP}$.

Proposición 5.21 *Sean* $j_0 \in \{1,\ldots,n\}$ *y* $v \in V$. *Las restricciones* $x_{pojo} \geq 0$ *y* $w_{j_0} \leq 1$ *definen facetas para* $\mathcal{SCP}$.

En el caso de las desigualdades válidas basadas en subgrafos y el número independiente de los mismos, obtuvimos la desigualdad de ***Clique*** y de ***Agujero***. Las siguientes proposiciones caracterizan su condición de facetas de $\mathcal{SCP}$.

Proposición 5.22 *Sea* $j_0 \in \{1,\ldots,n\}$ *y* $K \subset V$ *una clique maximal. La desigualdad*

$$\sum_{p \in K} x_{pj_0} - w_{j_0} \leq 0$$

es faceta de $\mathcal{CP}$.

Proposición 5.23 *Sea* $C_{2k+1} = \{v_1, ..., v_{2k+1}\}$, $k \geq 2$, *un agujero impar. La desigualdad*

$$\sum_{p \in C_{2k+1}} x_{pj_0} \leq k w_{j_0}$$

es válida para $\mathcal{SCP}$ *para todo* $j_0 = 1,\ldots,n$.

Si $\forall\ v \in V \setminus C_{2k+1}$, $G[C_{2k+1} \cup p]$ *tiene un conjunto independiente de medidad* $k+1$, *la desigualdad es faceta de* $\mathcal{SCP}$.

Si K es una clique tal que cada vértice *de K es adyacente a cada uno de los* vértices *de C_{2k+1} y es maximal respecto a esta propiedad, entonces la desigualdad*

$$\sum_{p \in C_{2k+1}} x_{pj_0} + k \sum_{v \in K} x_{vj_0} \leq k w_{j_0}$$

es faceta de $\mathcal{SCP}$.

En el caso de las ***Clique***, se mantuvo su condición de faceta para $\mathcal{CP}$ (salvo para $j_0 = n$). Sin embargo, no fue éste el caso para las ***Agujeros*** que si bien determinan una cara, su dimensión es mucho menor a la $dim(\mathcal{CP}) - 1$. Necesitamos incluir otras variables para reforzar la desigualdad, resultando una desigualdad no válida para $\mathcal{SCP}$.

Las siguientes desigualdades son las que dieron origen a la desigualdad ***Collar Multicolor*** y ***Cadena Multicolor***.

Proposición 5.24 *Sea $C_k = v_1, ..., v_k$, $k \geq 4$, un agujero. Consideremos $\{j_1, \ldots, j_{k-1}\}$ un conjunto de $k-1$ colores.*
La desigualdad

$$x_{v_1 j_1} + \sum_{i=2}^{k-1} (x_{v_i j_{i-1}} + x_{v_i j_i}) + x_{v_k j_{k-1}} + x_{v_k j_1} - w_{j_1} \leq k - 2$$

es válida para $\mathcal{SCP}$.

Si para todo $v \in V \setminus C_k$, v tiene al menos un vértice no adyacente en $\{v_1, v_2, v_k\}$, la desigualdad define una faceta de $\mathcal{SCP}$.

Proposición 5.25 *Sea $P_k = v_1, \ldots, v_k$, $k \geq 3$, un camino y $\{j_1, ..., j_k\}$ un conjunto de k colores. La desigualdad*

$$x_{v_1 j_1} + x_{v_1 j_2} + \sum_{i=2}^{k-1} (x_{v_i j_1} + x_{v_i j_i} + x_{v_i j_{i+1}}) + x_{v_k j_1} + x_{v_k j_k} - w_{j_1} \leq k - 1$$

es válida para $\mathcal{SCP}$.
Si $\forall\, v \in V \setminus P_k$, existe $v_i \in P_k$ no adyacente a v, entonces es faceta de $\mathcal{CP}$.

Si K es una clique tal que cada vértice *de K es adyacente a cada uno de los* vértices *de P_k y es maximal respecto a esta propiedad, entonces*

$$x_{v_1 j_1} + x_{v_1 j_2} + \sum_{i=2}^{k-1} (x_{v_i j_1} + x_{v_i j_i} + x_{v_i j_{i+1}}) + x_{v_k j_1} + x_{v_k j_k} + \sum_{v \in K} x_{v j_1} - w_{j_1} \leq k - 1$$

es faceta de $\mathcal{SCP}$.

Debido a que en $\mathcal{SCP}$ los colores no guardan ninguna relación entre sí, las desigualdades multicolores son muy débiles para $\mathcal{CP}$. Para aumentar la dimensión de la cara, las fortalecimos con la inclusión de las variables w_j correspondientes a los colores involucrados en la desigualdad y en algún caso con el agregado de variables que involucran otros colores.

Cabe señalar que en todos los casos, el estudio de la dimensión de una cara de $\mathcal{SCP}$ resultó mucho más sencillo que para $\mathcal{CP}$. La construcción de coloreos que pertenecen a las caras es más fácil pues no debe respetarse un orden entre los colores. Razón también para que las condiciones para que una desigualdad resulte faceta no sean tan restrictivas en algunos casos como lo son para $\mathcal{CP}$.

Hemos presentando en este capítulo los resultados de nuestro estudio poliedral de $\mathcal{CP}$. Por supuesto no es una caracterización completa del mismo y quedan aún familias de desigualdades válidas por estudiar. Mediante PORTA [15], un programa que calcula la cápsula convexa de un conjunto de puntos de coordenadas 0-1, hemos obtenido la caracterización completa del poliedro para *grafos* muy pequeños de apenas 6 o 7 *vértices* y hay un gran número de facetas que no pertenecen a ninguna de las familias descriptas.

Más allá del interés estrictamente teórico de las propiedades del poliedro, éstas adquieren mayor relevancia cuando brindan una herramienta útil para desarrollar algoritmos. Este es nuestro propósito. En el próximo Capítulo presentamos nuestro algoritmo *BC-Col* basado en estos resultados poliedrales.

Capítulo 6

El Algoritmo *BC-Col*

El propósito de este Capítulo es describir las caracteríticas principales de ***BC-Col***, un algoritmo *Branch-and-Cut* para el *problema de coloreo de grafos*. Ya hemos descripto en el Capítulo 3 las características generales de un algoritmo *Branch-and-Cut* y señalamos que su performance depende de muchos factores. Una buena relajación lineal y planos de corte con procedimientos de separación adecuados, ayuda al mejoramiento de las cotas inferiores. Esto es clave para podar ramas del árbol y disminuir el espacio de búsqueda y es aquí donde el estudio poliedral del problema es fundamental para considerar buenas desigualdes válidas. Por otro lado, las estrategias de recorrido del árbol, selección de nodos y algunos otros detalles de implementación aportan al conjunto del comportamiento del algoritmo. No hay una elección óptima de cada una de las alternativas del algoritmo válida para cualquier instancia. Las características propias del problema son las que ayudan a determinar una buena elección.

En las próximas secciones describimos detalladamente nuestro algoritmo. La presentación está dividida en dos partes. En la primera presentamos las características asociadas al esquema de *Branch-and-Bound*. En la segunda parte, mostramos los algoritmos de separación y otros detalles vinculados a la etapa de *Cutting*.

6.1. Esquema *Branch-and-Bound*

En esta Sección presentamos las características fundamentales de los factores que definen el esquema *Branch-and-Bound* de nuestro algoritmo. Esencialmente estos están enfocados en las estrategias de generación y manejo del árbol de búsqueda, preprocesamiento y mejoramiento de la relajación lineal.

6.1.1. Cotas Inferior y Superior

La experiencia demuestra que contar con buenas cotas iniciales, inferior y superior, es fundamental para reducir el tamaõ del árbol de búsqueda en un algoritmo *Branch-and- Cut*. La eficiencia del proceso de *Bound* depende en gran medida de la bondad de las cotas. Sin embargo, en un contexto de un algoritmo *Branch-and-Cut*, es necesario lograr un equilibrio entre el tiempo requerido

en obtener las cotas y la calidad de las mismas. Por este motivo elegimos procedimientos rápidos y sencillos que, de acuerdo a nuestra experimentación, brindan cotas de buena calidad.

Describimos a continuación los algoritmos usados para el cálculo de estas cotas.

Heurística Inicial

Hemos observado que en muchas instancias, **DSATUR** [10] encuentra rápidamente buenos coloreos, incluso en muchos casos el óptimo. La mayor parte del tiempo la insume en corroborar que no existe un coloreo mejor. Limitar el tiempo para el proceso de búsqueda resulta ser una muy buen estrategia como heurística inicial. La implementación es muy sencilla y cumple los requerimientos básicos para una heurística inicial: buenas cotas superiores en corto tiempo. Llamamos $\hat{\chi}$ a la cantidad de colores usados por el coloreo encontrado por la heurística.

Clique Maximal

El tamaño de una *clique* máxima de G, $\omega(G)$, es una cota inferior del número cromático. El problema de encontrar $\omega(G)$ es tan difícil como el de encontrar $\chi(G)$, es decir es *NP-Hard* [51]. Ante esta dificultad, implementamos una heurística para calcular una cota inferior de $\omega(G)$. La heurística es de tipo goloso. Se inicializa una *clique* con un *vértice* del grafo. En forma secuencial se agrega el *vértice* de mayor grado adyacente a los *vértices* de la *clique* hasta que no se encuentren más *vértices* con esta propiedad. Realizamos *Iter* iteraciones de este procedimiento, siendo *Iter* un parámetro de entrada. En la iteración i se toma como *vértice* inicial al *vértice* de mayor grado en el *grafo* que resulta de eliminar los *vértices* que ya fueron considerados para inicialar la *clique*. La heurística es muy sencilla, pero lo suficientemente efectiva para otorgar una buena cota inferior. Llamamos $\hat{\omega}$ al tamaño de la *clique* encontrada por la heurística. En la etapa final, los *vértices* del grafo son ordenados de tal manera que los $\hat{\omega}$ primeros corresponden a los *vértices* que forman la *clique*.

6.1.2. Preprocesamiento

La etapa de preprocesamiento tiene un objetivo claro: reducir el tamaño del problema en el número de variables y/o restricciones. En muchos casos, esta etapa marca la diferencia entre poder resolver o no una instancia tanto desde el punto de vista del tiempo como de requerimiento de memoria. Nuestra modelo de programación lineal entera tiene $n^2 + n$ variables y $n + mn + 2n$ restricciones. Con esta medida, la resolución de las relajaciones lineales puede consumir mucho tiempo, incluso para *grafos* no demasiado grandes (70 u 80 *vértices*).

Para disminuir la cantidad de variables y de restricciones hemos aplicado varias técnicas que describimos a continuación.

Vértices de la *Clique*

Disponemos de una *clique* maximal obtenida por una heurística cuyos vértices son $1, \ldots, \hat{\omega}$. Los *vértices* de la *clique* tienen que ser coloreados con distintos colores. Le asignamos los colores $1, \ldots, \hat{\omega}$ respectivamente. De esta manera, todas las variables x_{ij} para $i = 1, \ldots, \hat{\omega}$ y $j = 1, \ldots, n$ son eliminadas del modelo. Si un *vértice* v es adyacente al *vértice* i de la *clique*, no será posible colorearlo con el color i. Es decir, la variable x_{vi} es eliminada y las restricciones $x_{vk} + x_{ik} \leq 1$ son

eliminadas para todo $[v,i] \in E$ con $i = 1,\ldots,\hat{\omega}$ y $k = 1,\ldots,n$. Las variables w_j con $j = 1,\ldots,\hat{\omega}$ son fijadas en valor 1.

Eliminación de Colores

La cota superior $\hat{\chi}$ permite eliminar las variables w_j y x_{vj} para todo $j = \hat{\chi},\ldots,n$ y para todo $v \in V$. De esta manera, el conjunto de soluciones factibles se restringe al conjunto de los coloreos que usen a lo sumo $\hat{\chi} - 1$ colores. Si este conjunto es vacío, la solución óptima es la encontrada por la heurística inicial. Si no, allí se encuentra el óptimo y por eso es suficiente restringir la búsqueda en ese conjunto.

Eliminación de Vértices del Grafo

Ciertos *vértices* del *grafo* tienen la particularidad que si son eliminados, un coloreo en el *grafo* reducido puede ser extendido a un coloreo en el *grafo* original sin aumentar la cantidad de colores usados. Al reducir el tamaño del *grafo*, se reduce la cantidad de variables y restricciones del modelo. Por cada *vértice* v eliminado del *grafo*, las variables asociadas a v son eliminadas del modelo. Para eliminar *vértices* del *grafo* aplicamos dos procedimientos que detallamos a continuación.

- *Por adyacencias a la clique:*

  Sea $v \in V\backslash\{v_1,\ldots,v_{\hat{\omega}}\}$. Si v no es adyacente a v_k para algún $k = 1,\ldots,\hat{\omega}$ y todos los *vértices* adyacentes a v son adyacentes a v_k, entonces a partir de cualquier coloreo de $G[V\backslash\{v\}]$ puede derivarse uno para G. Basta colorear a v con el color k. Los *vértices* con esta propiedad son eliminados del grafo.

  Para aplicar el procedimiento se ordenan los *vértices* en forma ascendente por grado y en forma secuencial se chequea si pueden ser eliminados.

- *Por grado:*

  La eliminación de *vértices* por este criterio está fundamentada en el siguiente resultado:

  Lema: Sea $v \in V$ tal que $\delta(v) \leq \hat{\omega} - 2$. El número cromático de $G[V\backslash\{v\}]$ es igual a $\chi(G)$.

  Demostración: Supongamos que $\chi(G[V\backslash\{v\}) = \chi(G) - 1$. Entonces existe una partición $V_1, V_2, \ldots, V_{\chi(G)-1}$ de $V\backslash\{v\}$ en $\chi(G) - 1$ conjuntos independientes de $G[V\backslash\{v\}]$. Como $\delta(v) \leq \hat{\omega} - 2 \leq \chi(G) - 2$, en algún V_i, con $i = 1,\ldots,\chi(G) - 1$, no existen *vértices* adyacentes a v. Es decir, para algún $i \in \{1,\ldots,\chi(G) - 1\}$, $V_i \cup \{v\}$ es un conjunto independiente de V. Pero entonces, $V_1, V_2, \ldots, V_i \cup \{v\}, \ldots, V_{\chi(G)-1}$ es una partición de V en $\chi(G) - 1$ conjuntos independientes y por lo tanto tenemos un coloreo de G con menor cantidad de colores que $\chi(G)$. Esto es un absurdo.

  El proceso de eliminación se aplica recursivamente hasta que no queden *vértices* que verifiquen las condiciones del Lema.

Relajación Lineal

Después de la aplicación de los procedimientos arriba descriptos, el modelo tiene $m(\hat{\chi} - 1)$ restricciones de adyacencias: $x_{ij} + x_{kj} \leq w_j$ para todo $[i,k] \in E$ y $j = 1, \ldots, \hat{\chi} - 1$. A pesar de la reducción lograda, esta cantidad de restricciones todavía puede resultar grande cuando se resuelven las relajaciones lineales. Analizamos varias alternativas para reducir aún más el tamaño de la formulación.

En principio consideramos la posibilidad de encontrar una familia de *cliques* de tal manera que toda *arista* del grafo perteneciera al menos a una *clique*. Si $\hat{m}$ es la cantidad de *cliques* necesarias, las $m(\hat{\chi} - 1)$ restricciones de adyacencia pueden ser reemplazadas por $\hat{m}(\hat{\chi} - 1)$. Para que la reducción en el número de restricciones sea significativa, es conveniente encontrar una familia con el menor cardinal posible. Sin embargo esto no es fácil ya este problema es el conocido *problema de cubrimiento de aristas por cliques* que es $NP - Hard$ [34]. Nos vimos obligados a aplicar un procedimiento heurístico. Experimentamos con varios criterios golosos pero la reducción en la cantidad de restricciones no es suficiente. Los experimientos computacionales no fueron alentadores respecto a la reducción en el tiempo de resolución de las relajaciones lineales.

La segunda alternativa fue eliminar las restricciones de adyacencias de la formulación. Resulta así una relajación menos ajustada. Esperábamos que la incorporación de las desigualdades *Clique* en el algoritmo de planos de corte, de alguna manera compensaran la pérdida en la calidad de la relajación. Nuevamente los experimientos computacionales indicaron que no era una buena opción. La evolución en el incremento de la cota inferior por el agregado de planos de corte no resultó muy efectiva.

Por último, la alternativa es el reemplazo de las restricciones de adyacencia por

$$\sum_{i \in N(v)} x_{ij} + \mu x_{vj} \leq \mu w_j$$

donde $N(v)$ es la vecindad de v y μ es el cardinal de una partición en cliques de $N(v)$. Notar que, como μ es una cota superior del cardinal del máximo conjunto independiente de $G[N(v)]$, esta desigualdad resulta ser una relajación de las desigualdades de ***Vecindad***. Obtener el máximo cardinal de una partición es *NP-Hard* [34] y por eso aplicamos una heurística. Si bien la relajación es menos ajustada, la reducción a $n\hat{\chi}$ restricciones resultó muy conveniente en la práctica.

Para mejorar la relajación lineal, además se agregaron las restricciones

$$\sum_{j=1}^{\hat{\chi}} w_j \geq \sum_{j=1}^{\hat{\chi}} j x_{ij} \quad \forall\, i \in V$$

Estas desigualdades no son reduntantes y eliminan soluciones fraccionarias como $x_{ij} = 1/k$ para todo $i \in V, j \leq k$ y $3 \leq k \geq \hat{\chi}$.

6.1.3. Selección de Variable de *Branching*

La generación del árbol de búsqueda está definida por el proceso de *Branching*. En este etapa, el espacio de soluciones factibles asociado a un nodo se divide en dos o más conjuntos que representan los nuevos nodos (hijos) del árbol.

En los experimentos preliminares utilizamos el criterio clásico de dicotomía en una variable para generar los nodos del árbol. Se generan dos subproblemas por nodo, asociando a cada uno el conjunto de soluciones factibles donde la variable de *Branching* es fijada en 0 ó 1 respectivamente. Además de las alternativas conocidas que mencionamos en el Capítulo 3 (variable más cercana a 1/2, más cercana a 1, etc), implementamos algunas basadas en las propiedades de los *vértices*. Por ejemplo, entre las variables x^*_{vj} fraccionarias elegir la correspondiente al *vértice* v de mayor grado y el color j más usado. Ninguna de ellas resultó conveniente. Si x_{ij} es la variable de *Branching*, en el nodo hijo que resulta de fijar x_{ij} en 0 se incorpora una información poco relevante para el coloreo. Estamos diciendo que el *vértice* i no usa el color j, dejando libertad para la elección de cualquier otro color. En cambio, el nodo hijo correspondiente a $x_{ij} = 1$ restringe mucho más la región. Esto marca un desequilibrio en el árbol al que le atribuimos la mala performance del algoritmo.

También implementamos estrategias de *Branching* utilizando restricciones en lugar de variables. Experimentamos con dos ideas:

- *Por subconjunto de colores:* Sea Col^v_1 y Col^v_2 una partición del conjunto de colores factibles para el *vértice* v. Un coloreo del *grafo* asigna a v un color de sólo uno de los subconjuntos. Es decir, el espacio de soluciones factibles puede dividirse en dos: aquellos coloreos que verifican $\sum_{j \in Col^v_1} x_{vj} = 1$ y los que verifican $\sum_{j \in Col^v_2} x_{vj} = 1$. Si el *vértice* v y la partición de los colores es elegida de manera tal que existe $j_1 \in Col^v_1$ con $0 < x^*_{vj_1} < 1$ y existe existe $j_2 \in Col^v_2$ con $0 < x^*_{vj_2} < 1$, puede realizarse un proceso de *Branching*. Se generan dos nuevos nodos agregando las restricciones $\sum_{j \in Col^v_1} x_{vj} = 1$ y $\sum_{j \in Col^v_2} x_{vj} = 1$ respectivamente.

- *Por vértices no adyacentes:* Si v_1 y v_2 son *vértices* no adyacentes, un coloreo puede asignar a ambos el mismo color o no. Si j es un color tal que $1 < x^*_{v_1 j} + x^*_{v_2 j} < 2$ se pueden generar dos nuevos nodos agregando las restricciones $x^*_{v_1 j} + x^*_{v_2 j} \leq 1$ y $x^*_{v_1 j} + x^*_{v_2 j} = 2$ respectivamente .

Ninguna de estas dos últimas estrategias tuvo una buena performance en la práctica.

Finalmente, nuestro critero toma como base las ideas del algoritmo de Brélaz [10]. Sea (x*,w*) la solución fraccionaria óptima de la relajación lineal en un nodo del árbol de búsqueda. Entre los *vértices* cuyas variables x_{ij} son fraccionarias, se elige el *vértice* que tenga la vecindad *más colorida*. Es decir, el *vértice* con mayor cantidad de colores distintos usados por sus adyacentes que ya tienen fijo su color en la rama del árbol. En caso de empate consideramos dos posibilidades, la propuesta por Brélaz en DSATUR [10] y la sugerida por Sewell [73].

- *VB1:* el *vértice* de mayor grado.

- *VB2:* para cada *vértice* v del empate y para cada color c factible para v se calcula la cantidad de adyacentes que podrían utilizar el color c. El *vértice* con mayor suma es elegido.

Sea v el *vértice* elegido y k la cantidad de colores usados hasta el momento en la coloración parcial del grafo. El número k está determinado por la cantidad de variables w_j fijadas en 1. Para cada $j = 1, \ldots, k$, si x_{vj} no está fijada en 0 y no fue eliminada del conjunto de variables, se genera un nodo hijo fijando la variable $x_{vj} = 1$. En el caso que $k + 1$ sea menor que la cota superior obtenida hasta el momento, se abre otro hijo fijando $x_{vk+1} = 1$.

Si bien por cada nodo la cantidad de hijos que se abren es mayor que en el caso de dicotomía, la experiencia nos mostró una marcada superioridad de esta estrategia.

6.1.4. Estrategias de Recorrido del Árbol

Después de la etapa de *Branching* se debe seleccionar un nodo de la lista L de nodos aún no explorados (*abiertos*). Esto determina la forma en que es recorrido el árbol. Como ya mencionamos, las estrategias básicas son tres: **DFS** (profundidad), **BFS** (ancho) y **BestF** (mejor cota). Con las opciones **BFS** y **BestF** se presentaron problemas de memoria y por eso nos decidimos por **DFS** para *BC-Col*.

Las alternativas que implementamos difieren en el orden en que son ingresados los nodos hijos en la lista L de nodos abiertos. Cada nodo hijo tiene asociado un color y es lo que usamos para definir un orden. Los órdenes considerados son:

- **Orden 1:** ascendente en índice de color.
- **Orden 2:** primero el color $k + 1$ y luego en orden ascendente en índice de color.
- **Orden 3:** para cada color, se calcula la cantidad de *vértices* que tienen fijo ese color. Se ordenan los colores de acuerdo a esa cantidad de mayor a menor. Nos referiremos a este orden como *mayor a menor uso*.
- **Orden 4:** de *menor a mayor uso*.

La estrategia de *Branching* y el orden en ingresar los nodos a la lista resultan claves en la performance del algoritmo. En el próximo Capítulo analizaremos las distintas combinaciones, para poder establecer empiricamente cúal es la mejor.

6.1.5. Fijado de Variables por Implicaciones Lógicas

En muchos problemas combinatorios, una decisión tomada en cierta variable del problema trae como consecuencia una serie de decisiones implícitas. En el caso del *problema de coloreo*, si se decide colorear a un *vértice* i con el color j, ninguno de sus *vértices* adyacentes puede usar el color j. Esto ocurre cada vez que se elige una variable de *Branching*. Si x_{ij} es la variable de *Branching*, entonces $x_{il} = 0$ para todo $l \neq j$ y $x_{vj} = 0$ para todo *vértice* v adyacente a i. Además, $w_j = 1$. Por otro lado, si Z^* es el valor óptimo de la relajación lineal en un nodo del árbol, entonces cualquier coloreo que se encuentre en la región factible asociada a ese nodo debe usar al menos $\lceil Z^* \rceil$ colores. Por lo tanto, pueden fijarse las variables w_j en valor 1 para $j = 1, \ldots, \lceil Z^* \rceil$. *BC-Col* tiene implementandas estas simples consideraciones pues reducen el tamaño de problema y las relajaciones lineales resultan más rápidas de resolver.

6.1.6. Enumeración Implícita

Para *grafos pequeños*, la enumeración implícita es más rápida que un algoritmo *Branch-and-Cut*. El porcentaje de tiempo invertido en la resolución de las relajaciones lineales en los nodos del árbol de búsqueda es muy significativo y una enumeración brinda en estos casos una mejor performance. El mejoramiento de las cotas inferiores a partir del valor ópitmo de las relajaciones es más significativa en los primeros niveles del árbol y disminuye a medida que aumenta la profundidad. En *BC-Col* consideramos esta situación. En cada nodo del árbol sabemos la cantidad de vértices que aún no tienen fijo su color. Si esa cantidad es *pequeña*, la ramas del árbol que se derivan de ese nodo son exploradas usando enumeración implícita. La denominación *pequeño* no es muy rigurosa. De acuerdo a nuestra experimentación, la enumeración implícita es rápida en *grafos* con baja densidad (inferior al 20 %). En este caso, se pueden resolver instancias generadas al azar de hasta 100 *vértices* en pocos segundos. Los *grafos* con densidades intermedias son mucho más difíciles. En grafos al azar con poco más de 60 *vértices* y densidades entre 30 % y 70 % se evidencia una baja performance de la enumeración implícita. Los *grafos* de alta densidad (mayor al 70 %) presentan dificultades para este enfoque en grafos con 70 *vértices* o más.

En nuestra implementación, el nivel del árbol donde se toma la decisión de realizar una enumeración implícita es un parámetro de nuestro algoritmo *BC-Col* y la misma es realizada con el algoritmo DSATUR [10].

6.2. Etapa de *Cutting*: Algoritmos de Separación

La aplicación de un algoritmo de planos de corte tiene una etapa decisiva: encontrar desigualdades violadas. Es decir, la etapa de *separación*. Para lograr una buena performance global los procedimientos de separación deben ser rápidos. En algunos casos este objetivo puede cumplirse con algoritmos exactos, pero en otros es necesario implementar heurísticas. Estas últimas no aseguran que siempre se pueda detectar desigualdes violadas, pero es una solución de compromiso ante el alto costo computacional de algunos problemas de separación. Desarrollamos procedimientos de separación para algunas de las desigualdades válidas obtenidas a partir del estudio poliedral de $\mathcal{CP}$ que presentamos en el Capítulo anterior. A continuación describimos los algoritmos.

6.2.1. Separación de Desigualdades *Clique*

Dada $K \subset V$ una *clique* maximal de G, la desigualdad *Clique*

$$\sum_{v \in K} x_{vj_0} \leq w_{j_0}$$

es válida para $\mathcal{CP}$.

Las desigualdades *Clique* fueron estudiadas por Padberg [67] para el poliedro asociado al *problema de conjunto independiente* y tienen una amplia aplicación en muchos problemas de programación entera binaria. Varios paquetes de uso general como CPLEX [23] implementan estos cortes.

El problema de separación de estas desigualdades es *NP-Hard* y esencialmente hay 2 estrategias básicas para implementar un algoritmo de separación:

- Generar al comienzo del algoritmo *Branch-and-Cut* un conjunto de *cliques* maximales y guardarlas en una tabla. Para el proceso de separación, chequear en la tabla la existencia de desigualdades *Clique* violadas.
- Buscar *cliques* maximales en el momento de la separación utilizando información de la solución de la relajación lineal.

La primera estrategia tiene la ventaja de buscar *cliques* maximales en el grafo una única vez evitando así reiterar esta búsqueda en etapas posteriores. La cantidad de *cliques* maximales de un grafo puede ser muy grande y puede resultar restrictivo, desde el punto de vista de tiempo y memoria, armar y conservar una tabla con ellas. Esta es la estrategia usada por CPLEX. En [4], se presenta un estudio comparativo de ambas estrategias para problemas generales. Si bien no se llega a una conclusión contundente hay cierta ventaja de la segunda sobre la primera. Nuesta experiencia no fue buena al implementar la primera. No sólo por los problemas de memoria sino también porque el proceso encuentra pocas desigualdades violadas. La generación de *cliques* del *grafo* que no usan de alguna manera la información de la solución que se intenta separar reduce la posibilidad de detectar una *clique* violada. Nos inclinamos entonces por la segunda estrategia. Para eso implementamos una heurística basada en un proceso goloso.

Sea (x^*, w^*) la solución de la relajación. Para cada $w^*_{j_0}$, se arma una lista ordenada en forma decreciente de las variables $x^*_{ij_0}$ fraccionarias o nulas. En forma iterativa, por cada uno de los *vértices* de las variables fraccionarias de la lista buscamos *cliques* maximales. En la iteración j, se inicializa una *clique* con el *vértice* v_1 correspondiente a la variable $x^*_{v_1 j_0}$ que se encuentra en el j-esimo lugar de la lista. La cantidad de *cliques* que se buscan para cada *vértice* está determinada por el parámetro $Cant_{int}$. Para buscar la *clique* número k, se elige v_2 como el k-esimo *vértice* de mayor valor $x^*_{v_2 j_0}$ en la lista entre los adyacentes a v_1. Luego, en forma secuencial, se recorre la lista a partir del lugar que ocupa v_2. Todo *vértice* adyacente a los *vértices* que se encuentran ya en la *clique* es agregado a la misma.

En un principio sólo consideramos una *clique* por *vértice* y observamos que no era una cantidad suficiente de cortes. En el otro extremo, la generacion de *cliques* a partir de todos los posibles v_1, v_2 resulta excesiva y genera muchas desigualdades con soporte muy similar. Según la experiencia, el parámetro $Cant_{int}$ fue fijado en 3.

6.2.2. Separación de Desigualdades *p-color Clique*

Sea $K = \{v_1, \ldots, v_p\} \subset V$ una clique, j_0 un color tal que $p \leq j_0 \leq n-1$ y $Col = \{j_1, j_2, \ldots, j_{p-1}\} \subset \{1, \ldots, j_0 - 1\}$ La desigualdad ***p-color Clique***

$$\sum_{i=1}^{p} \sum_{j=j_0}^{n} x_{v_i j} + \sum_{i=1}^{p} \sum_{j \in Col} x_{v_i j} \leq w_{j_0} + \sum_{j \in Col} w_j$$

es válida para $\mathcal{CP}$.

Para implementar la separación de estas desigualdades resultó muy conveniente aprovechar el proceso previo de detección de desigualdades *Clique* violadas. Sea $K = v_1, v_2, \ldots, v_p$ la *clique*

generada por dicho proceso y $p \leq j_0$. Sabemos que $\sum_{i=1}^{p} x^*_{v_i j_0} > w^*_{j_0}$ y por lo tanto resulta que $\sum_{j=j_0}^{n} \sum_{i=1}^{p} x^*_{v_i j} > w^*_{j_0}$. Esta última desigualdad puede ser aún más estricta que la anterior debido a que sumamos sobre más colores. Parece razonable entonces utilizar a K para buscar una desigualdad ***p-color Clique*** violada. Falta determinar el conjunto de $p-1$ colores. Una estrategia *golosa* es elegir los colores sobre los cuales los *vértices* de la *clique* violan o verifican lo más ajustadamente posible la desigualdad ***Clique*** correspondiente a ellos. Para todos los colores $j = 1, \ldots, j_0 - 1$, calculamos $Sum_j = \sum_{i=1}^{p} x^*_{v_i j} - w^*_j$. Ordenamos en forma decreciente los valores Sum_j y elejimos los $p-1$ primeros. Si $Sum_{j_1} + Sum_{j_2} + \ldots + Sum_{j_{p-1}} + \sum_{j=j_0}^{n} \sum_{i=1}^{p} x^*_{v_i j} - w^*_{j_0} > 0$, la desigualdad ***p-color Clique*** está violada.

6.2.3. Separación de Desigualdades *Anula Color*

Sea $v \in V$. Por el orden establecido entre los colores, la desigualdad ***Anula Color***

$$\sum_{j=j_0}^{n} x_{vj} \leq w_{j_0}$$

es válida para $\mathcal{CP}$.

Hay a lo sumo $n(\hat{\chi} - \hat{\omega})$ posibles desigualdades ***Anula Color***. Para que la desigualdad pueda estar violada, es condición necesaria que w_{j_0} sea fraccionaria. Si $w_{j_0} = 0$, ningún color de índice mayor o igual a j_0 puede ser utilizado y por lo tanto $\sum_{j=j_0}^{n} x_{vj} = 0$ para todo $v \in V$. Además, como $\sum_{j=1}^{n} x_{vj} = 1$, podemos asegurar que $\sum_{j=j_0}^{n} x_{vj} \leq 1$ y por lo tanto tampoco será posible violar la desigualdad ***Anula Color*** si $w_{j_0} = 1$.

Con un proceso de enumeración, para cada color j_0 tal que $w^* j_0$ es fraccionaria, se calcula para cada vértice v del grafo la suma $\sum_{j=j_0}^{\hat{\chi}} x^*_{vj}$. Si esta suma es superior a $w^*_{j_0}$, la desigualdad ***Anula Color*** está violada.

6.2.4. Separación de Desigualdades *Camino Multicolor*

Sea $P_k = \{v_1, v_2, \ldots, v_k\}$ un *camino* de longitud k. Consideremos $c_1, c_2, \ldots, c_{k-1}$ y c_{j_0} colores tales que $c_i < c_{j_0}$ para todo $i = 1, \ldots, k-1$ y $c_{j_0} \leq n-2$. La desigualdad ***Camino Multicolor***

$$x_{v_1 c_1} + \sum_{i=2}^{k-1} x_{v_i c_{i-1}} + x_{v_i c_i} + x_{v_k c_{k-1}} + \sum_{j=c_{j_0}}^{n} \sum_{i=1}^{k} x_{v_i j} \leq \sum_{i=1}^{k-1} w_{c_i} + w_{c_{j_0}}$$

es válida de $\mathcal{CP}$.

Para que la desigualdad pueda estar violada, es condición necesaria que $w_{c_{j_0}}$ no tenga valor 1. En ese caso, el término derecho de la desigualdad toma el valor k que es la cantidad de *vértices* del *camino*. La suma de colores en cada *vértice* del *camino* es a lo sumo 1 y por lo tanto, sobre todos los vértices no supera el valor k. En el caso que $w_{j_0} = 0$, si bien la desigualdad puede estar violada esto sucede sólo si alguna restricción de adyacencia lo está. La experiencia nos mostró que son muy pocas las veces que se detectan violaciones en esta situación y por eso decidimos no considerarla.

Sea j_0 tal que $0 < w^*_{j_0} < 1$. Asignamos a las *aristas* del grafo un peso. Para cada par de *vértices* adyacentes u, v, elegimos el color $c < j_0$ tal que $x^*_{uc} + x^*_{vc} - w^*_c \geq x^*_{uj} + x^*_{vj} - w^*_j$ para cualquier otro color $j \neq c$, $j < j_0$. El peso de la *arista* $[uv]$ es $c_{uv} = x^*_{uc} + x^*_{vc} - w^*_c + \frac{1}{2}(\sum_{j=j_0}^{\hat{\chi}} x^*_{uj} + x^*_{vj})$.

Buscamos un *camino* de peso máximo para lo cual procedemos de la siguiente manera: Se inicializa un *camino* con un *vértice* v y se realizan $\delta(v)$ iteraciones. En la iteración k elige el próximo *vértice* como el *vértice* adyacente *no prohibido* cuya *arista* es la k-esima de mayor peso entre las incidentes a v. Se reitera el procedimiento considerando como v al último *vértice* agregado al *camino* hasta alcanzar un criterio de parada.

¿Qué queremos decir con *no prohibido* y cúales son los criterios de parada? La experiencia computacional nos mostró que no es conveniente que los *vértices* pertenezcan a muchos *caminos* pues las desigualdades resultan con soporte similar. Esto es controlado por el parámetro $MaxVeces$. Si un *vértice* ya formó parte de $MaxVeces$ *caminos*, entonces no puede ser nuevamente elegido y está *prohibido*. Además *caminos* muy *largos* no son buenos candidatos a violar la desigualdad. El parámetro $LongMax$ sirve para detener la búsqueda en el caso que la longitud del *camino* haya alcanzado ese valor. Si la longitud del *camino* sumada a $0{,}5(\sum_{j=j_0}^{\hat{\chi}} x^*_{v_1 j} + x^*_{v_r j})$ es mayor que $w^*_{j_0}$ con v_1 y v_r el primer y último vértice del camino, la desigualdad *Camino Multicolor* está violada.

6.2.5. Separación de Desigualdades *Agujero*

Sea $C_k = \{v_1, v_2, \ldots, v_k\}$ un *ciclo* de medida k. La desigualdad ***Agujero***

$$\sum_{i=1}^{k} x_{v_i j_0} \leq \lfloor k/2 \rfloor w_{j_0}$$

es válida de $\mathcal{CP}$.

El procedimiento de separación que usamos está basado en el algoritmo polinomial *GLS* propuesto por Grotschel et al. en [41] para separar desigualdades de ciclo para el problema de conjunto independiente.

Sea j_0 un color tal que $w^*_{j_0}$ no es nula. Se arma un grafo bipartito $B = (V_1 \cup V_2, E')$ de la siguiente manera: por cada *vértice* v tal que $x^*_{vj_0}$ es fraccionaria, incluimos un *vértice* v_1 en V_1 y otro v_2 en V_2. Además si $[u, v]$ es una *arista* de G, entonces $[u_1, v_2]$ y $[v_1, u_2]$ son *aristas* del grafo bipartito con peso igual al máximo entre 0 y $w^*_{j_0} - x^*_{uj_0} - x^*_{vj_0}$.

El algoritmo *GLS* trabaja sobre una relajación del problema de conjunto independiente sobre la cual puede afirmar que los pesos que asigna a las *aristas* son positivos. Este no es nuestro caso y por eso asignamos un peso nulo para evitar pesos negativos.

Consideremos un *vértice* v de V y calculemos P_v el *camino* mínimo en B entre los *vértices* v_1 y v_2. Los *vértices* que constituyen el *camino* forman un *ciclo* en G. Este es el que consideramos para detectar una desigualdad de *Agujero* violada. No podemos afirmar, como en el caso del problema de conjunto independiente, que la desigualdad está violada. Pero le experiencia computacional nos

mostró una buena performance del algoritmo respecto a la detección de violaciones. En nuestra implementación, usamos el algoritmo de Dijkstra para construir el *camino* de longitud mínima.

6.2.6. Separación vs *Branching*

Después de resolver la relajación lineal de un nodo del árbol, se debe decidir si se generan cortes o se procede a realizar el *Branching*. Es de esperar que los planos de corte ayuden a mejorar las cotas y esto permita podar ramas del árbol. Sin embargo, el proceso de búsqueda de desigualdades violadas y la posterior resolución de la relajación tienen un costo. Debe entonces lograrse un equilibrio entre estas dos posibilidades. Para manejar esta decisión utilizamos dos parámetros:

- ***Skip Factor:*** El valor de *skip factor* indica cada cúantos nodos explorados del árbol se aplica el proceso de separación. También puede relacionarse la decisión con el nivel del árbol en lugar de los nodos.
- ***IPC:*** Limita la cantidad de iteraciones que se realizan del algoritmo de planos de corte en cada nodo del árbol.

Los valores de estos parámetros no son fáciles de determinar. En el Capítulo 7 hacemos un análisis para determinar un valor conveniente a través de la experimentación con *grafos* de variada densidad.

También disponemos de dos parámetros para manejar las iteraciones de acuerdo al incremento de la cota inferior. Si los valores óptimos de dos sucesivas relajaciones difieren en menos un valor (*TailOffPercent*) y esto se mantiene durante una cierta cantidad de iteraciones (*TailOffLps*), el proceso se termina.

En la experimentación con *BC-Col* optamos por limitar las iteraciones de acuerdo al parámetro *IPC*.

6.2.7. Detalles de Implementación

Nuestro algoritmo *BC-Col* fue implementado usando el entorno ABACUS [75] y el paquete de optimización CPLEX [23]. ABACUS es muy robusto y tiene implementadas las estrategias básicas de *Branching* y recorrido del árbol, entre otros factores que definen un algoritmo *Branch-and-Cut*. Pero tal vez su principal ventaja es que brinda el marco y la posibilidad de implementar estrategias particulares. Tiene un diseño que otorga un marco sobre el cual esencialmente el manejo del árbol de enumeración, de los planos de corte y de las relajaciones asociadas a cada nodo están a su cargo. El programador tiene posibilidad de modificarlos a través de parámetros o en algunos casos, de la inclusión de procedimientos que complementan el esquema general.

A continuación describimos las características básicas de ABACUS que consideramos para implementar *BC-Col*.

- *Resolución de las Relajaciones Lineales:* ABACUS tiene una interface con CPLEX, XPRESS y SOPLEX. Cuando se necesita resolver una relajación lineal, ABACUS utiliza alguna de

las rutinas que proveen estos paquetes. La interface es automática y sólo se requiere que el programador especifique con que paquete quiere interactuar. En nuestro caso utilizamos CPLEX.

- *Manejo de la Memoria:* Parte del manejo de la memoria se realiza a través de una estructura denominada *pool*. Existen *pools* específicos para las variables, para las restricciones y para los cortes.
  - Pool de variables: Se almacenan datos propios de las variables (por ejemplo el *vértice* y color al que están asociadas) que facilita la manipulación de las variables.
  - Pool de restricciones: Se guardan las restricciones de la relajación lineal inicial del problema. Es estático y no puede ser modificado en el transcurso de la ejecución. Cuando la cantidad de restricciones es muy grande, existe la posibilidad de crear un *pool* adicional donde se guardan parte de las restricciones del problema. El ABACUS considera la relajación lineal sin estas restricciones y resuelve. Antes de proseguir con el esquema general del *Branch-and-Cut*, chequea si todas las restricciones del *pool* son satisfechas por la solución actual. Si no es así, agrega a la formulación las que se encuentren violadas y re-optimiza la relajación. Si en algún momento estas restricciones se vuelven inactivas, son nuevamente eliminadas de la formulación. Al comienzo de nuestra implementación consideramos esta posibilidad para las restricciones de adyacencia pero los resultados no fueron lo suficientemente buenos como para que valiera la pena tenerla en cuenta.
  - Pool de cortes: Se almacenan las desigualdades violadas que se generan a través de los procesos de separación por orden de aparición. Este *pool* es dinámico y dimensionado al comienzo de la ejecución. Una desigualdad permanece allí hasta que el espacio es requerido para el ingreso de otro corte.
- *Manejo de los cortes:* Después de resolver la relajación en un nodo del árbol se procede a realizar el *Branching* o a aplicar un algoritmo de planos de corte. En este último caso, los desigualdades válidas violadas que se encuentren con los algoritmos de separación son agregadas a la formulación y se re-optimiza la relajación. Se realiza tantas iteraciones de este proceso como lo indique el parámetro *IPC*. Estas desigualdades permanecen en la formulación hasta que se vuelven inactivas, en cuyo caso son eliminadas.

  Ya describimos que cada vez que se genera un corte, éste es almacenado en el *pool* de cortes. ABACUS brinda la posibilidad que antes de llamar a los procesos específicos de separación, se chequee si existen desigualdades violadas en ese *pool*. Este es un procedimiento interno del ABACUS y suele ser muy rápido. Este manejo tiene el beneficio de disponer globalmente de cortes que fueron generados en un nodo en particular. Estos cortes tal vez no sean detectados en otro nodo si el proceso de separación utiliza métodos heurísticos. De esta manera se cuenta con dos posibles fuentes de planos de corte: el *pool* de cortes y los algoritmos de separación. ABACUS deja la decisión al usuario de cúales procesos ejecutar.

  Si ambos procesos son ejecutados, puede ocurrir que una desigualdad violada sea detectada dos veces y por lo tanto aparezca duplicada en la relajación. ABACUS proporciona una herramienta opcional para evitar estas duplicaciones. Según nuestra experiencia, el incremento de tiempo en los algoritmos de separación debido al chequeo de duplicación es despreciable y es beneficioso aplicarlo.

En *BC-Col*, utilizamos las dos posibles fuentes de cortes: el *pool* de cortes y los algoritmos de separación con chequeo de duplicación. Con el fin de evitar el agregado de un gran número de desigualdades, imponemos un límite superior a cada una de las familias de cortes. Además, si la cantidad de cortes encontrados en el *pool* supera una cierta cantidad, no llamamos a los procesos de separación. Todos estos valores son manejados por parámetros de entrada.

La decisión de cuando y cuantas veces aplica un algoritmo de planos de cortes es determinada por los valores de los parámetros ***skip factor*** y IPC descriptos antes.

En este Capítulo describimos las principales caracteríticas de *BC-Col*. Es difícil establecer a priori cuál es la elección más adecuada para las diferentes alternativas del algoritmo. A través de la experimentación se puede evaluar el comportamiento del algoritmo. De esta manera se puede definir empíricamente una buena estrategia y ajustar los valores de los parámetros. Éste es el propóstio del próximo Capítulo.

Capítulo 7

Experiencia Computacional

En este Capítulo mostramos nuestra experimentación computacional con *BC-Col*. Las pruebas realizadas tuvieron el objetivo de evaluar la performance del mismo y de ajustar los valores de los parámetros.

En el diseño de *BC-Col* tuvimos en cuenta varios factores: preprocesamiento, algoritmos de separación para algunas familias de desigualdades válidas que surgieron del estudio poliedral de $\mathcal{CP}$, diferentes reglas de *Branching* y estrategias de recorrido del árbol. Las combinación entre las diferentes componentes con sus distintas alternativas puede impactar significativamente en el algoritmo. Nuestro propósito fue, a través de la experiencia computacional, encontrar un buen balance entre las diferentes opciones.

El algoritmo está implementado en C++, utilizando el entorno ABACUS [75] y CPLEX [23]. Todos los experimentos computacionales fueron realizados en una plataforma SUN Ultra Sparc 4.

7.1. Instancias de Prueba

Trabajamos con instancias generadas al azar y con la librería de problemas test de DIMACS, Center for Discrete Mathematics & Theoretical Computer Science [29].

En las Tablas 7.1 y 7.2 mostramos las instancias de DIMACS con la cantidad de *vértices*, cantidad de *aristas*, el *número cromático* y una referencia al origen y características del *grafo*. Además, los valores de $\hat{\omega}$ y $\hat{\chi}$ obtenidos con nuestras heurísticas iniciales, a las cuales le hemos dado un máximo de 5 segundos de CPU. Un signo '?' significa que el número cromático es desconocido. Notar que en varias instancias, las cotas inferior y superior coinciden y por lo tanto se obtuvo el óptimo sin necesidad de la etapa de *Branch-and-Cut*. En general, estos casos corresponden a *grafos* que originalmente o después del proceso previo de eliminación de *vértices*, tienen poca cantidad de *vértices* o son poco densos.

Problema	vértices	aristas	$\hat{\omega}$	$\hat{\chi}$	χ
DSJC125_1 (1)	125	736	4	5	?
DSJC125_5 (1)	125	3891	9	20	?
DSJC125_9 (1)	125	6961	32	49	?
DSJC250_1 (1)	250	3218	4	9	?
DSJC250_5 (1)	250	15668	11	36	?
DSJC250_9 (1)	250	27897	37	88	?
DSJC500_1 (1)	500	12458	5	15	?
DSJC500_5 (1)	500	62624	12	63	?
DSJC500_9 (1)	500	1124367	47	161	?
DSJR500_1 (1)	500	3555	12	12	?
DSJR500_1C (1)	500	121275	72	87	?
DSJR500_5 (1)	500	58862	117	131	?
DSJC1000_1 (1)	1000	49629	6	26	?
DSJC1000_5 (1)	1000	249826	14	116	?
DSJC1000_9 (1)	1000	449449			?
fpsol2_i_1 (2)	496	11654	55	65	65
fpsol2_i_2 (2)	451	8691	29	30	30
fpsol2_i_3 (2)	425	8688	29	30	30
inithx.i.1 (2)	864	18707	**54**	**54**	**54**
inithx.i.2 (2)	645	13979	**31**	**31**	**31**
inithx.i.3 (2)	621	13969	**31**	**31**	**31**
latin_squ_10 (3)	900	307350	90	129	?
le450_15a (4)	450	8168	15	17	15
le450_15b (4)	450	8169	15	17	15
le450_15c (4)	450	16680	15	24	15
le450_15d (4)	450	16750	15	23	15
le450_25a (4)	450	8260	**25**	**25**	**25**
le450_25b (4)	450	8263	**25**	**25**	**25**
le450_25c (4)	450	17343	**25**	**28**	**25**
le450_25d (4)	450	17425	25	28	25
le450_5a (4)	450	5714	5	9	5
le450_5b (4)	450	5734	5	9	5
le450_5c (4)	450	9803	**5**	**5**	**5**
le450_5d (4)	450	9757	5	10	5
mulsol.i.1 (2)	197	3925	**49**	**49**	**49**
mulsol.i.2 (2)	188	3885	**31**	**31**	**31**
mulsol.i.3 (2)	184	3916	**31**	**31**	**31**
mulsol.i.4 (2)	185	3946	**31**	**31**	**31**
mulsol.i.5 (2)	185	3973	**31**	**31**	**31**
school1 (5)	385	19095	**14**	**14**	**14**
school1_nsh (5)	352	14612	**14**	**14**	**14**
zeroin.i.1 (2)	211	4100	**49**	**49**	**49**
zeroin.i.2 (2)	211	3541	**30**	**30**	**30**
zeroin.i.3 (2)	206	3540	**30**	**30**	**30**

Problema	vértices	aristas	$\hat{\omega}$	$\hat{\chi}$	χ
anna (6a)	138	493	**11**	**11**	**11**
david (6a)	87	406	**11**	**11**	**11**
homer (6a)	561	1629	**13**	**13**	**13**
huck (6a)	74	301	**11**	**11**	**11**
jean (6a)	80	254	**10**	**10**	**10**
games120 (6b)	120	638	**9**	**9**	**9**
miles1000 (6c)	128	3216	41	42	42
miles1500 (6c)	128	5198	71	73	73
miles250 (6c)	128	387	**8**	**8**	**8**
miles500 (6c)	128	1170	**20**	**20**	**20**
miles750 (6c)	128	2113	**31**	**31**	**31**
queen10_10 (6d)	100	2940	10	12	?
queen11_11 (6d)	121	3960	11	14	11
queen12_12 (6d)	144	5192	12	15	?
queen13_13 (6d)	169	6656	13	16	13
queen14_14 (6d)	196	8372	14	17	?
queen15_15 (6d)	225	10360	15	18	?
queen16_16 (6d)	256	12640	16	20	?
queen5_5 (6d)	25	160	5	**5**	**5**
queen6_6 (6d)	36	290	**6**	**7**	**7**
queen7_7 (6d)	49	476	**7**	**7**	**7**
queen8_12 (6d)	96	1368	**12**	**12**	**12**
queen8_8 (6d)	64	728	8	9	9
queen9_9 (6d)	81	1056	9	11	10
myciel3 (7)	11	20	2	4	4
myciel4 (7)	23	71	2	5	5
myciel5 (7)	47	236	2	6	6
myciel6 (7)	95	755	2	7	7
myciel7 (7)	191	2360	2	8	8
mug88_1 (8)	88	146	3	4	4
mug88_25 (8)	88	146	3	4	4
mug100_1 (8)	100	166	3	4	4
mug100_25 (8)	100	166	3	4	4
abb313GPIA (9)	1557	46546	8	10	?
ash331GPIA (9)	662	4185	3	4	?
ash608GPIA (9)	1216	7844	3	4	?
ash958GPIA (9)	1916	12506	3	5	?
will199GPIA (9)	701	6772	5	7	?
1-Insertions_4 (10)	67	232	2	5	?
1-Insertions_5 (10)	202	1227	2	6	?
1-Insertions_6 (10)	607	6337	2	7	?
2-Insertions_3 (10)	37	72	2	4	4
2-Insertions_4 (10)	149	541	2	5	4
2-Insertions_5 (10)	597	3936	2	6	?

Tabla 7.1: Instancias DIMACS

Problema	vértices	aristas	$\hat{\omega}$	$\hat{\chi}$	χ	Problema	vértices	aristas	$\hat{\omega}$	$\hat{\chi}$	χ
3-Insertions_3 (10)	56	110	2	4	4	4-FullIns_4 (10)	690	6650	6	8	?
3-Insertions_4 (10)	281	1046	2	5	?	4-FullIns_5 (10)	4146	77305	6	9	?
3-Insertions_5 (10)	1406	9695	2	6	?	5-FullIns_3 (10)	154	792	7	8	?
4-Insertions_3 (10)	79	156	2	4	?	5-FullIns_4 (10)	1085	11395	7	9	?
4-Insertions_4 (10)	475	1795	2	5	?	wap01 (11)	2368	110871	41	46	?
1-FullIns_3 (10)	30	100	3	4	?	wap02 (11)	2464	111742	40	45	?
1-FullIns_4 (10)	93	593	3	5	?	wap03 (11)	4730	286722	40	56	?
1-FullIns_5 (10)	282	3247	3	6	?	wap04 (11)	5231	294902	40	50	?
2-FullIns_3 (10)	52	201	4	5	?	wap05 (11)	905	43081	50	51	?
2-FullIns_4 (10)	212	1621	4	6	?	wap06 (11)	947	43571	40	44	?
2-FullIns_5 (10)	852	12201	4	7	?	wap07 (11)	1809	103368	40	46	?
3-FullIns_3 (10)	80	346	5	6	?	wap08 (11)	1870	104176	40	47	?
3-FullIns_4 (10)	405	3524	5	7	?	qg_order30 (12)	900	26100	**30**	**30**	**30**
3-FullIns_5 (10)	2030	33751	5	8	?	qg_order40 (12)	1600	62400	40	42	40
4-FullIns_3 (10)	114	541	6	7	?	qg_order60 (12)	3600	212400	60	63	60

Tabla 7.2: Instancias DIMACS

(1) Grafos generados al azar (*David Johnson*).

(2) Problemas reales de alocación de registros (*Gary Lewandowski*).

(3) Grafos Leighton generados con número cromático conocido (*Craig Morgenstern*).

(4) Problemas de asignación de aulas (*Gary Lewandowski*).

(5) Grafos *Latin square* (*Gary Lewandowski*).

(6a) Grafos de libros: dado un libro, cada personaje es representado por un vértice. Dos vértices son adyacentes si los personajes se encuentran durante la historia. Los libros son Anna Karenina, David Copperfield, Homero, Huckleberry Finn y Los Miserables (*Donald Knuth*).

(6b) Grafos deportivos: para un campeonato deportivo se crea un vértice por cada equipo y una arista entre equipos que compiten. En particular, es el grafo de la temporada futboística de 1990 en USA (*Donald Knuth*).

(6c) Grafos geográficos: los vértices representan un conjunto de ciudades. Dos vértices son adyacentes si la distancia entre las ciudades correspondientes es menor a un valor fijo (*Donald Knuth*).

(6d) Grafos Reina: se considera un tablero de ajedrez con n^2 casillas. Por cada casilla se define un vértice. Dos vértices son adyacentes si las casillas se encuentran en la misma fila, columna o diagonal (*Donald Knuth*).

(7) Grafos Mycielski: grafos para los que la medida de una clique máxima es 2 pero el número cromático aumenta en la medida del grafo (*Michael Trick*).

(8) Grafos casi 3-coloreables con clique máxima de medida 4 muy difícil de encontrar (*Kuzonori Mizuno*).

(9) Grafos obtenidos para determinar matrices Jacobinas ralas (*Shahadat Hossain*).

(10) Generalizaciones de grafos Mycielski (*M. Caramia y P. Dell'Olmo*).

(11) Problemas reales de diseño de redes de fibra óptica (*Arie Koster*).

(12) Grafos *Latin square* (*Carla Gomes*).

También usamos instancias propias generadas al azar, $G(n,p)$, donde n es la cantidad de *vértices*. Estos *grafos* fueron generados de tal manera que todo par u,v de *vértices*, tiene probabilidad p de ser adyacentes. De esta manera, la densidad del *grafo* es aproximadamente $100p$.

7.2. Reducción de Vértices

Hay instancias para las cuales el tamaño de la relajación lineal es muy grande. El tiempo utilizado por las rutinas de CPLEX para resolverlas es casi prohibitivo e incluso se presentan problemas de memoria. Esto hace que sea imposible abordar el problema con un algoritmo *Branch-and-Cut*.

El proceso de eliminación de *vértices* resulta en muchos casos una etapa esencial para la resolución del problema. La eliminación de *vértices* fue significativa en las instancias de DIMACS. No se repite la misma la situación con los *grafos* al azar. Por la forma en que son generados, estos *grafos* tienen los grados de los *vértices* muy similares. Debido a esta propiedad, los criterios aplicados para la eliminación de *vértices* son satisfechos por muy pocos o por ningún *vértice* del *grafo*.

Para ilustrar la importancia de esta etapa, en la Tabla 7.3 mostramos los tamaños de las relajaciones antes y después de la reducción de *vértices* correspondientes a 5 instancias DIMACS. Como puede observarse, la reducción es muy significativa en *grafos* de distintas densidades.

Problema	% Dens	Var	Restricc	Var	Restricc
fpsol2_i_1	9	20749	21630	2288	2519
4-FullIns_3	8	540	404	756	560
zeroin_i_1	24	5460	5944	82	122
miles1500	63	2315	2428	84	111
DSJC500_1c	97	6756	7611	5348	6023

Tabla 7.3: Relajaciones Reducidas

En la Tabla 7.4 mostramos las instancias DIMACS en las que logramos disminuir el tamaño del *grafo*. En la columna 2 aparece la densidad y en la tercera columna la cantidad original de *vértices*. En la cuarta columna la cantidad de *vértices* que queda después del proceso de eliminación. En la última columna indicamos el porcentaje de la disminución de *vértices*. Si bien hay instancias de variada densidad, en general, el proceso tiene mayor incidencia en los *grafos* con densidad baja o alta.

Este preprocesamiento es imprescindible cuando queremos resolver instancias medianas a grandes. Sin esta etapa hay instancias para las cuales no es posible aplicar *BC-Col* o los tiempos de resolución son prohibitivos.

Problema	% Dens.	n	$\hat{n}$	% Red.	Problema	% Dens.	n	$\hat{n}$	% Red.
DSJR500_1	3	500	109	78	miles1000	39	128	50	61
DSJR500_1C	97	500	410	18	miles1500	63	128	85	34
DSJR500_5	47	500	491	2	miles250	5	128	15	88
fpsol2_i_1	9	496	171	66	miles500	14	128	28	78
fpsol2_i_2	9	451	164	64	miles750	26	128	37	71
fpsol2_i_3	10	425	163	62	abb313GPIA	4	1557	1400	10
inithx.i.1	5	864	115	87	ash331GPIA	2	662	661	1
inithx.i.2	7	645	182	72	ash608GPIA	1	1216	1215	1
inithx.i.3	7	621	172	72	ash958GPIA	1	1916	1915	1
latin_square_10	76	900	129	86	will199GPIA	3	701	697	1
le450_15a	8	450	409	9	1-FullIns_3	22	30	21	30
le450_15b	8	450	413	8	1-FullIns_4	14	93	63	32
le450_25a	8	450	271	40	1-FullIns_5	8	282	189	33
le450_25b	8	450	302	33	2-FullIns_3	15	52	40	23
le450_25c	17	450	436	3	2-FullIns_4	7	212	160	25
le450_25d	17	450	436	3	2-FullIns_5	3	852	640	25
mulsol.i.1	20	197	49	75	3-FullIns_3	11	80	65	19
mulsol.i.2	22	188	100	47	3-FullIns_4	4	405	325	20
mulsol.i.3	23	184	101	45	3-FullIns_5	2	2030	1625	20
mulsol.i.4	23	185	102	45	4-FullIns_3	8	114	84	26
mulsol.i.5	23	185	102	45	4-FullIns_4	3	690	576	17
school1	26	385	358	7	4-FullIns_5	1	4146	3456	17
school1_nsh	24	352	328	7	5-FullIns_3	7	154	79	49
zeroin.i.1	18	211	63	70	5-FullIns_4	2	1085	931	14
zeroin.i.2	16	211	57	73	wap01	4	2368	1771	25
zeroin.i.3	17	206	56	73	wap02	4	2464	2174	12
anna	5	138	17	88	wap03	3	4730	4701	1
david	11	87	11	87	wap04	2	5231	5204	1
homer	1	561	38	93	wap05	11	905	665	27
huck	11	74	11	85	wap06	10	947	787	17
jean	8	80	13	84	wap07	6	1809	1655	9
games120	9	120	119	1	wap08	6	1870	1696	9

Tabla 7.4: Reducción de Vértices

7.3. Variable de *Branching* y Estrategia de Recorrido

Si bien el tamaño del árbol de búsqueda depende también de los cortes que se implementen, en nuestra experimentación inicial observamos que el comportamiento de las estrategias de *Branching* y recorrido era similar se aplicaran o no planos de corte. Por este motivo, para presentar nuestra evaluación de dichas estrategias hemos decidido no considerar la aplicación de planos de corte en *BC-Col.*

Las estrategias de selección de variable de *Branching* que comparamos son *VB1* y *VB2* combinadas con las estrategias de recorrido *O1*,*O2*,*O3* y *O4*. Además consideramos *VB0-1*, regla de *Branching* por dicotomía en la variable *más fraccionaria* con búsqueda en profundidad.

Las estrategias de recorrido basadas en búsqueda a lo *ancho* o *mejor* cota las descartamos. Las

experiencias preliminares mostraron rápidamente que pueden presentarse problemas de memoria por el tamaño del árbol.

Consideramos *grafos* generados al azar con densidades 30%, 50%, 70% y 90%. Para cada estrategia, resolvimos 10 instancias de cada densidad y tomamos el promedio de tiempo y cantidad de nodos del árbol. Los Gráficos 7.1 resumen los resultados de nuestra experimentación.

Si fijamos la estrategia de recorrido, en casi todos los casos la cantidad de nodos y el tiempo fue inferior con *VB2*. En [73], Sewell propone un algoritmo enumerativo que utiliza *VB2+O1* para la generación del árbol. De acuerdo a su experiencia, el tamaño del árbol de búsqueda es inferior al tamaño del árbol generado por el algoritmo DSATUR que utiliza *VB1+O1*. Sin embargo, en muchas instancias el tiempo es mayor. Esto es consecuencia del mayor trabajo requerido para elegir un *vértice* con la estrategia *VB2* que con *VB1*. Si bien nuestra experiencia con estos algoritmos enumerativos confirma los resultados de Sewell, no observamos el mismo comportamiento al considerarlos en un esquema *Branch-and-Bound*.

Respecto a las estrategias de recorrido, en términos generales podemos afirmar que con *O2* se obtuvieron mejores resultados. Al comienzo de nuestra experimentacion teníamos confianza en que las alternativas *O3* y *O4* funcionaran bien. Suponíamos que tal vez requirieran mayor tiempo, pero esperábamos una reducción en el tamaño del árbol. Los resultados nos probaron lo contrario. Sabemos que las soluciones factibles que se encuentran a lo largo del proceso de búsqueda brindan cotas superiores del óptimo. Éstas se usan para podar el árbol. Con *O2* se recorre en primer lugar la rama correspondiente a un color nuevo. Esto posibilita encontrar soluciones factibles más fácilmente pues se dispone de mayor cantidad de colores para colorear el *grafo*. Nuestra experiencia demuestra que este proceder *goloso* de *O2* da buenos resultados.

Basados en este análisis, nuestra propuesta para la elección de variable de *Branching* y recorrido del árbol es $VB2 + O2$.

Por último, en los Gráficos 7.2 comparamos la estrategia clásica de dicotomía *VB0-1* con la peor y la mejor de nuestras estrategias. Ésta es la estrategia clásica utilizada por la mayoría de los paquetes de optimización, presentándose variaciones en el criterio de selección de la variable de *branching*.

La experiencia con la estrategia de dicotomía confirmó la superioridad de las estrategias propuestas. La performance es claramente inferior al resto, presentándose casos que superaron el límite de tiempo de CPU impuesto (5hs), en particular para *grafos* de media densidad. Este resultado no nos estrañó. En el capítulo anterior ya mencionamos la característica poco balanceada del árbol que explica esta baja performance.

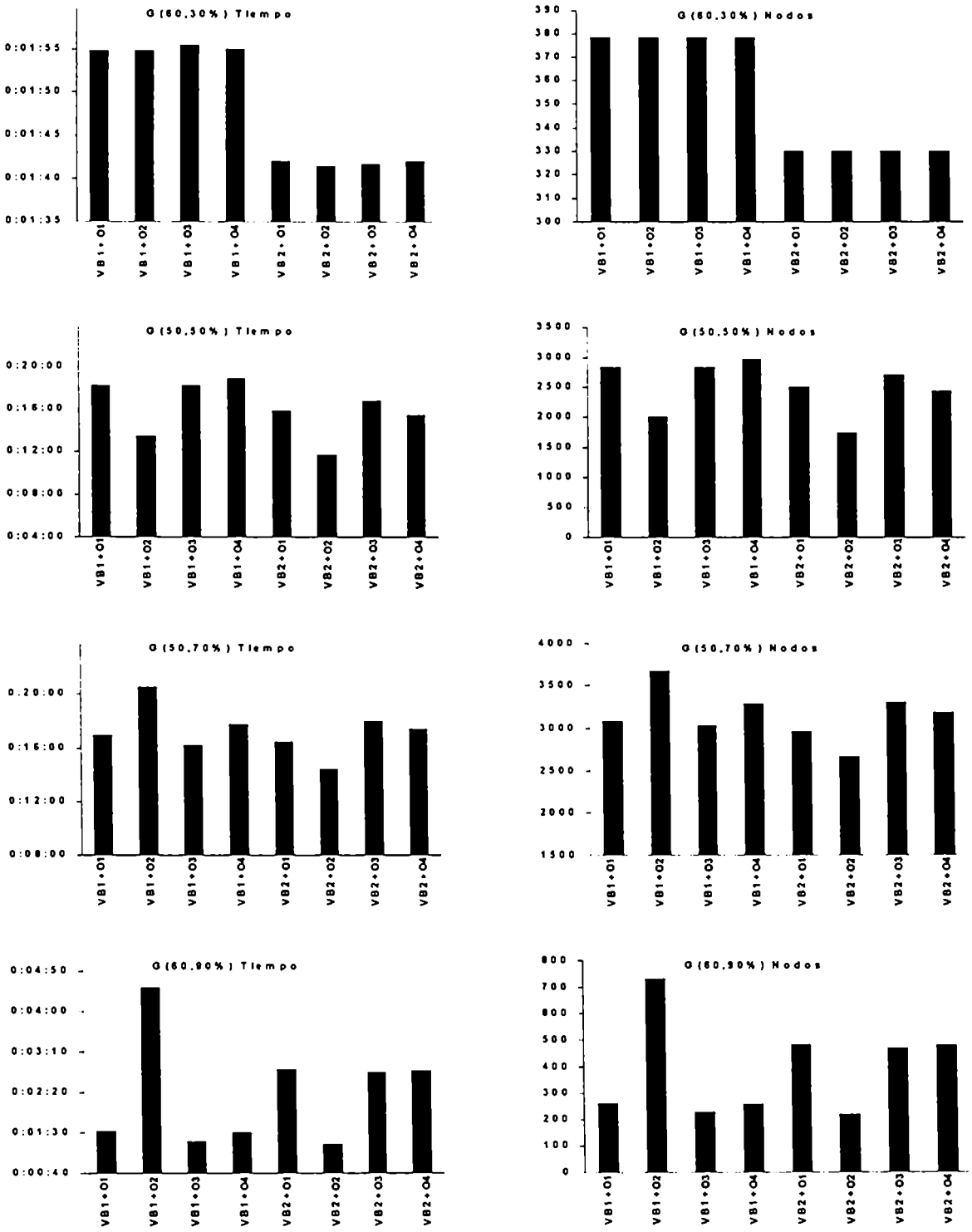


Gráfico 7.1: *Variable de Branching + Estrategias de Recorrido*

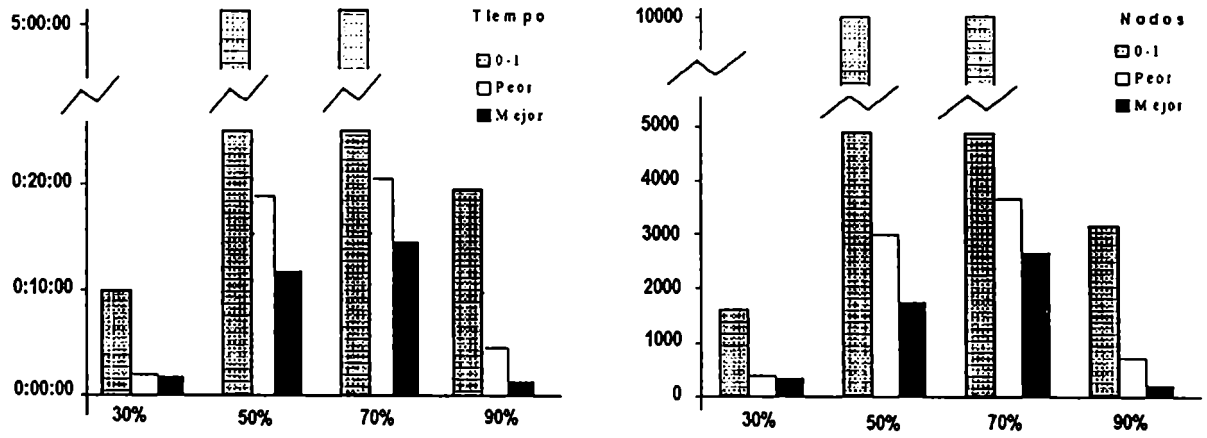


Gráfico 7.2: 0-1 vs Mejor vs Peor

7.4. Evaluación de las Desigualdades Válidas

Una manera de evaluar la calidad de los planos de corte es observar el incremento producido en la cota inferior cuando son agregados a la relajación lineal. A mayor incremento, mejor la calidad de la desigualdad válida. Sin embargo, hay que buscar un balance entre diferentes aspectos. Si los cortes son muy densos, se incrementa el requerimiento de memoria y la resolución de la relajación es más lenta. Si los algoritmos de separación insumen mucho tiempo en relación al beneficio obtenido en el incremento de la cota inferior, no vale la pena incluirlos en el algoritmo. A continuación mostramos el análisis sobre distintos aspectos que hicimos de las desigualdades válidas que fundamenta nuestro juicio sobre las mismas.

7.4.1. Evolución de la Cota Inferior

La primera evaluación de las diferentes familias de cortes fue realizada considerando el mejoramiento de la cota inferior a nivel nodo raíz. Utilizamos *grafos* al azar de 60, 70, 100 y 125 *vértices* y generamos instancias para densidades bajas (menores a 30 %), densidades medias (entre 40 % y 60 %) y densidades altas (mayores a 70 %). Realizamos 50 iteraciones del algoritmo de planos de corte, con un máximo por iteración de 300 cortes ***Clique***, 50 ***Anula Color***, 300 ***p-color Clique***, 300 ***Camino Multicolor*** y 300 ***Agujero***.

Como la función objetivo es entera, el redondeo al entero superior del valor óptimo fraccionario de la relajación lineal brinda una cota inferior del número cromático. Esto permite fijar en 1 el valor de las variables w_j cuyos índices son menores o iguales a dicha cota. Las desigualdades válidas tienen mayor posibilidad de ser violadas en el caso de la presencia de variables w_j fraccionarias. Esto nos llevó a plantearnos la alternativa de redondear y fijar variables después de finalizada la etapa de *cutting* o hacerlo después de cada relajación lineal. Un comportamiento típico del algoritmo en ambas situaciones puede observarse en los Gráficos 7.3. Los datos corresponden a un *grafo* de 100 *vértices* y 70 % de densidad donde se aplicaron todos los cortes durante 50 iteraciones del algoritmo.

Si bien pudimos detectar que la cantidad de cortes encontrados cuando no redondeamos es mayor, el crecimiento de la función objetivo es más rápido con la segunda opción. En general, se alcanza la misma cota en la misma cantidad de iteraciones, pero se tarda más como consecuencia del tiempo involucrado en la separación de las desigualdades.

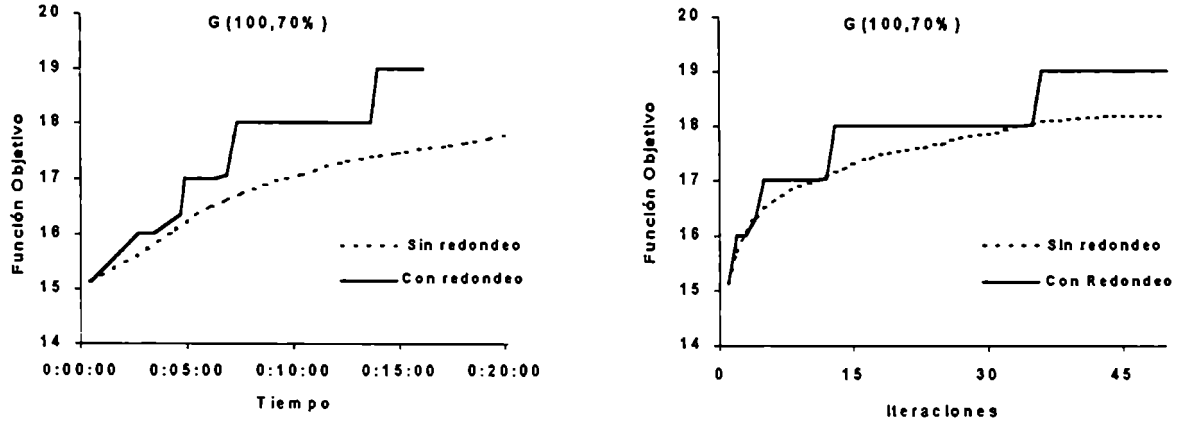


Gráfico 7.3: Redondear vs No Redondear

Como disponemos de 5 familias de cortes, cabe preguntarse si existe alguna combinación entre ellas que tenga una mejor performance comparada con las otras. De todos los factores que estudiamos, este fue el de mayor dificultad y que nos llevó el mayor tiempo de estudio para obtener una respuesta. Experimentamos con muchas instancias de diferente cantidad de *vértices* y densidades, buscando alguna caracterítica que nos indicara la ventaja de un corte sobre otro.

En las Tablas 7.5, 7.6 y 7.7 reportamos nuestra experiencia sobre *grafos* de 125 *vértices*. Indicamos el valor de la cota inferior, el tiempo total y número de iteración en la que es obtenida y el tiempo correspondiente al proceso de separación.
La primera conclusión categórica es que sin las desigualdades *Clique*, el algoritmo de planos de corte no logra la misma calidad en la cota inferior. La presencia de estas desigualdades es clave para mejorar las relajaciones, por lo tanto descartamos rotundamente cualquier combinación de planos de corte que no las incluye.

C_1	Clique	C_{10}	Clique+Anula Color+Agujero
C_2	Clique+Anula Color	C_{11}	Clique+Camino Mulicolor+Agujero
C_3	Clique+Camino Mulicolor	C_{12}	Clique+p-color Clique+Agujero
C_4	Clique+p-color Clique	C_{13}	Clique+Anula Color+CaminoMulticolor+Agujero
C_5	Clique+Anula Color+Camino Mulicolor	$C14$	Clique+Anula Color+p-color Clique+Agujero
C_6	Clique+Anula Color+p-color Clique	$C15$	Clique+Camino Mulicolor+p-color Clique+Agujero
C_7	Clique+Camino Mulicolor+p-color Clique	$C16$	Clique+Anula Color+Camino Mulicolor+p-color Clique+Agujero
C_8	Clique+Anula Color+Camino Mulicolor+p-color Clique	$C17$	Anula Color+Camino Mulicolor+p-color Clique+Agujero
C_9	Clique+Agujero		

	I1			I2			I3			I4		
	$\hat{\omega}$	$\hat{x}$	Op-Rel	$\hat{\omega}$	$\hat{x}$	Op-Rel	$\hat{\omega}$	$\hat{x}$	Op-Rel	$\hat{\omega}$	$\hat{x}$	Op-Rel
	6	11	0	6	10	0	6	11	0	5	11	0.074
	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter
C_1	1	108	9	1	85	3	1	95	9	2	187	11
C_2	1	108	9	1	85	3	1	96	9	2	187	11
C_3	1	108	9	1	85	3	1	95	9	2	202	10
C_4	1	109	9	1	86	3	1	96	9	2	221	10
C_5	1	108	9	1	85	3	1	95	9	2	203	10
C_6	1	109	9	1	87	3	1	96	9	2	221	10
C_7	1	109	9	1	84	3	1	96	9	2	260	10
C_8	1	109	9	1	84	3	1	96	9	2	230	11
C_9	1	142	9	1	266	4	1	131	9	2	208	10
C_{10}	1	142	9	1	266	4	1	131	9	2	209	10
C_{11}	1	141	9	1	265	4	1	130	9	2	236	11
C_{12}	1	142	9	1	265	4	1	130	9	2	234	10
C_{13}	1	142	9	1	265	4	1	130	9	2	237	11
C_{14}	1	142	9	1	266	4	1	130	9	2	233	10
C_{15}	1	142	9	1	268	4	1	131	9	2	261	9
C_{16}	1	142	9	1	268	4	1	131	9	2	261	9
C_{17}	0	27	1	1	64	5	0	23	1	1	48	1

	I5			I6			I7			I8		
	$\hat{\omega}$	$\hat{x}$	Op-Rel	$\hat{\omega}$	$\hat{x}$	Op-Rel	$\hat{\omega}$	$\hat{x}$	Op-Rel	$\hat{\omega}$	$\hat{x}$	Op-Rel
	6	12	0	7	13	0	7	12	0	6	12	0.062
	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter
C_1	2	347	23	1	131	10	1	154	12	2	213	12
C_2	2	348	23	1	131	10	1	154	12	2	213	12
C_3	2	406	23	1	134	10	1	155	12	2	178	11
C_4	2	423	24	1	134	10	1	155	12	2	204	11
C_5	2	406	23	1	131	10	1	153	12	2	210	11
C_6	2	424	24	1	134	10	1	155	12	2	204	11
C_7	2	407	22	1	132	10	1	153	12	2	194	9
C_8	2	407	22	1	133	10	1	153	12	2	194	9
C_9	2	407	26	1	119	8	1	195	13	2	250	13
C_{10}	2	407	26	1	119	8	1	195	13	2	250	13
C_{11}	2	424	21	1	120	8	1	195	13	2	285	11
C_{12}	2	451	24	1	120	8	1	195	13	2	290	12
C_{13}	2	425	21	1	120	8	1	195	13	2	240	11
C_{14}	2	451	24	1	120	8	1	195	13	2	290	12
C_{15}	2	447	22	1	120	8	1	195	13	2	261	11
C_{16}	2	447	22	1	120	8	1	195	13	2	262	11
C_{17}	1	129	3	0	26	1	0	21	1	1	38	1

Tabla 7.5: Planos de Corte en Grafos de Baja Densidad

En los *grafos* de baja densidad, el valor óptimo de la relajación inicial es generalmente nulo. La cota inferior no mejora sustancialmente y es alcanzada en las primeras iteraciones del algoritmo después de las cuales se estabiliza. Incluso hay instancias en las que el algoritmo termina antes del límite de 50 iteraciones por no encontrar cortes violados. No hay una marcada diferencia en el número de iteración en la que cada combinación alcanza la cota, pero se evidencia un aumento del tiempo en las combinaciones que incorporan a las desigualdades *Agujero*.

En los *grafos* de media densidad, la relajación inicial tampoco otorga una cota inferior de buena calidad. Sin embargo, la incorporación de los cortes mejora el valor óptimo de la relajación. En ningún caso la diferencia de los valores óptimos correspondientes a iteraciones sucesivas es superior a 1. En general, la primera mejora de la cota se produce durante las primeras iteraciones del

	I1			I2			I3			I4		
	$\dot{\omega}$	$\dot{\chi}$	Op-Rel	$\dot{\omega}$	$\dot{\chi}$	Op-Rel	$\dot{\omega}$	$\dot{\chi}$	Op-Rel	$\dot{\omega}$	$\dot{\chi}$	Op-Rel
	8	16	0	8	15	0	8	16	0	8	16	0
	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter
C_1	2	484	21	2	515	22	1	531	27	2	452	22
C_2	2	484	21	2	517	22	1	533	27	2	451	22
C_3	2	446	20	2	499	20	1	526	26	2	471	21
C_4	2	498	18	2	542	20	1	580	23	2	462	20
C_5	2	447	20	2	503	20	1	533	26	2	477	21
C_6	2	503	18	2	550	20	1	589	23	2	468	20
C_7	2	554	19	2	549	19	1	556	23	2	521	22
C_8	2	555	19	2	559	19	1	555	23	2	521	22
C_9	2	534	20	2	749	22	1	607	25	2	567	23
C_{10}	2	532	20	2	760	22	1	602	25	2	566	23
C_{11}	2	502	21	2	588	20	1	579	25	2	501	21
C_{12}	2	570	20	2	659	18	1	648	23	2	537	21
C_{13}	2	502	21	2	594	20	1	582	25	2	500	21
C_{14}	2	570	20	2	659	18	1	648	23	2	534	21
C_{15}	2	539	19	2	686	20	1	628	23	2	557	22
C_{16}	2	542	19	2	685	20	1	629	23	2	557	22
C_{17}	0	27	1	1	294	20	0	27	1	1	277	15
	I5			I6			I7			I8		
	$\dot{\omega}$	$\dot{\chi}$	Op-Rel	$\dot{\omega}$	$\dot{\chi}$	Op-Rel	$\dot{\omega}$	$\dot{\chi}$	Op-Rel	$\dot{\omega}$	$\dot{\chi}$	Op-Rel
	7	16	0.057	9	20	0.271	11	24	0.667	11	24	0.698
	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter
C_1	3	935	32	3	1118	25	4	2337	43	4	2244	39
C_2	3	930	32	3	997	27	4	2919	38	4	2148	35
C_3	3	954	34	3	917	23	4	3121	35	4	2268	38
C_4	3	1128	32	3	1434	18	4	2986	42	4	2691	37
C_5	3	954	34	3	1036	20	4	2440	41	4	2161	38
C_6	3	1131	32	3	1135	18	4	2896	41	4	2293	37
C_7	3	1293	34	3	1326	10	4	2365	38	4	2426	38
C_8	3	1294	34	3	1060	19	4	2693	30	4	2723	35
C_9	3	941	31	3	951	25	4	2690	30	4	2749	36
C_{10}	3	947	31	3	807	21	4	2059	35	4	2612	37
C_{11}	3	1076	36	3	1131	25	4	2204	42	4	3421	36
C_{12}	3	1170	35	3	1125	18	4	2645	39	4	2664	35
C_{13}	3	1076	36	3	975	24	4	2216	38	4	2333	38
C_{14}	3	1174	35	3	1283	19	4	2913	41	4	3102	36
C_{15}	3	1066	33	3	1034	20	4	2481	40	4	2551	38
C_{16}	3	1067	33	3	1106	19	4	3320	40	4	2691	34
C_{17}	1	55	1	2	519	7	2	1178	3	2	1571	11

Tabla 7.6: Planos de Corte en Grafos de Media Densidad

algoritmo, luego se estabiliza, vuelve a mejorar promediando el número de iteraciones y finaliza estabilizada. No se presentaron casos en los cuales el algoritmo finalizó por no poder encontrar cortes violados.

Para los *grafos* de alta densidad el valor óptimo de la relajación incrementa sustancialmente la cota inferior inicial que además mejora en las sucesivas iteraciones del algoritmo. Hay una mayor diferencia en el número de iteración donde cada combinación alcanza la cota pero varía de una instancia a otra sin evidenciarse relación entre las mismas. No hay una combinación que prime sobre otra, aunque la incorporación de las desigualdades *Agujero* empeora los tiempos.

Para complementar nuestro estudio con otro análisis de los datos, decidimos utilizar una *medida de eficiencia*. Nuestra idea es medir cuán lejos se encuentra una determinada combinación respecto de la mejor combinación para cada una de las instancias. Para eso, para cada instancia i, sea $Tiempo_i^{C_j}$ el tiempo en obtener la cota inferior por la combinación C_j de cortes. Notamos por $Mejor_i$ a $min\{Tiempo_i^{C_j}, j = 1, \ldots, 16\}$. Para cada combinación C_j calculamos P_j, la suma sobre

	I1			I2			I3			I4		
	$\hat{\omega}$	$\hat{\chi}$	Op-Rel	$\hat{\omega}$	$\hat{\chi}$	Op-Rel	$\hat{\omega}$	$\hat{\chi}$	Op-Rel	$\hat{\omega}$	$\hat{\chi}$	Op-Rel
	14	31	1.646	15	29	1.138	14	30	1.859	16	30	0.746
	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter
C_1	5	2500	27	5	3005	46	5	2108	22	4	1360	26
C_2	5	2502	27	5	2848	50	5	2470	25	4	1666	22
C_3	5	2755	29	5	4072	49	5	1752	25	4	1678	26
C_4	5	3990	28	5	3039	47	5	2236	25	4	1742	23
C_5	5	3083	30	5	3322	47	5	1976	27	4	1685	25
C_6	5	3993	28	5	4979	46	5	2254	26	4	2194	24
C_7	5	3379	32	4	2159	14	5	2039	24	4	1698	22
C_8	5	3375	32	5	4233	48	5	1926	25	4	2148	25
C_9	5	2599	29	5	3213	47	5	2141	25	4	1200	23
C_{10}	5	2602	29	5	3140	43	5	1954	24	4	1744	23
C_{11}	5	2674	29	4	2639	15	5	2199	24	4	1866	22
C_{12}	5	2965	30	4	1976	15	5	2878	22	4	2002	22
C_{13}	5	2863	27	5	2864	47	5	2981	26	4	1463	23
C_{14}	5	2967	30	5	4690	49	5	2973	26	4	1960	24
C_{15}	5	3697	29	4	1926	16	5	2873	25	4	1873	25
C_{16}	5	3696	29	5	3767	50	5	2843	25	4	2584	23
C_{17}	2	125	1	2	84	1	3	2330	6	2	1198	5
	I5			I6			I7			I8		
	$\hat{\omega}$	$\hat{\chi}$	Op-Rel	$\hat{\omega}$	$\hat{\chi}$	Op-Rel	$\hat{\omega}$	$\hat{\chi}$	Op-Rel	$\hat{\omega}$	$\hat{\chi}$	Op-Rel
	29	49	4.886	31	48	4.469	32	48	4.184	33	48	4.067
	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter	CotaInf	Tiempo	Iter
C_1	8	2067	33	8	1430	50	8	1244	38	8	1043	40
C_2	8	1989	41	8	1209	40	8	1576	433	8	1037	45
C_3	8	2395	36	8	1694	50	8	1724	42	8	963	38
C_4	8	2349	36	8	1388	43	8	1784	39	8	1142	42
C_5	8	1710	36	8	1248	40	8	1372	38	8	1025	39
C_6	8	2334	333	8	1548	43	8	1978	40	8	1009	39
C_7	8	2258	30	8	1501	39	8	1643	45	8	1454	45
C_8	8	3188	37	8	1631	46	8	1511	40	8	1348	45
C_9	8	2046	37	8	1442	46	8	1444	34	8	936	36
C_{10}	8	1697	32	8	1376	42	8	1229	355	8	969	38
C_{11}	8	2488	35	8	1729	42	8	1614	433	8	1261	46
C_{12}	8	2209	35	8	1735	48	8	1866	40	8	1392	42
C_{13}	8	2103	34	8	1514	38	8	1510	43	8	982	39
C_{14}	8	3288	39	8	2011	47	8	2047	41	8	1417	44
C_{15}	8	2313	35	8	2386	47	8	1849	42	8	1364	48
C_{16}	8	2634	36	8	1772	43	8	2166	43	8	1489	42
C_{17}	5	95	1	5	79	1	5	66	1	5	50	1

Tabla 7.7: Planos de Corte en Grafos de Alta Densidad

las 8 instancias de la diferencia relativa entre $Tiempo_i^{C_j}$ y $Mejor_i$. Es decir

$$P_j = \sum_{i=1}^{8} (Tiempo_i^{C_j} - Mejor_i)/Mejor_i$$

Una combinación que tenga un valor bajo de esta medida, nos indica un buen esquema de corte. En el Gráfico 7.4 mostramos los valores obtenidos.

Podemos observar que la desigualdad ***p-color Clique*** no está presente entre las combinaciones de mejor comportamiento, independientemente de la densidad del *grafo*. En *grafos* de alta densidad, la presencia de los cortes ***Agujero*** es notablemente perjudicial. En *grafos* de media y baja densidad, si bien no tan categóricamente como en el caso de baja densidad, su presencia no es esencial para obtener una buena combinación de cortes.

Para *grafos* de densidad media, la combinación C_5 de ***Cliques***, ***Anula Color*** y ***Camino Multicolor*** resulta la mejor. De acuerdo a nuestra experiencia y a la reportada en muchos trabajos de la literatura, el *problema de coloreo* en *grafos* de media densidad suele ser más difícil. Es por eso que decidimos basar nuestra decisión en estos *grafos* e inclinarnos por la combinación C_5.

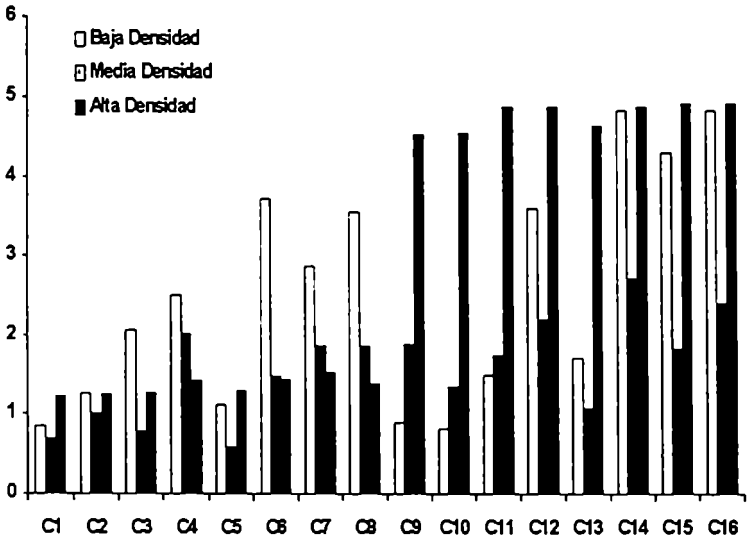


Gráfico 7.4: Eficiencia de Esquemas de Cortes

De cualquier manera, esta elección es suficientemente buena para *grafos* de baja y alta densidad, encontrándose en todos los casos no muy alejada de la mejor combinación.

7.4.2. Tiempo de Separación

Un factor que no influye en nuestra elección es el tiempo correspondiente a los procedimientos de separación. La Tabla 7.8 muestra para cada una de las combinaciones el tiempo promedio sobre las 8 instancias para realizar las 50 iteraciones del algoritmo de planos de corte, el tiempo promedio del proceso de separación y su porcentaje sobre el tiempo total .

Se puede observar que la proporción sobre el tiempo total de los tiempos de separación no es importante, aunque se acrecienta a medida que baja la densidad del *grafo*. Las relajaciones sobre *grafos* de baja densidad se resuelven más rápido y por eso la proporción de tiempo de separación aumenta. El proceso para separar las desigualdades ***Agujero*** es factor determinante para el aumento de la proporción en el tiempo total de separación y en menor medida, el proceso para separar desigualdades ***p-color Clique***. Consideramos que el tiempo de los algoritmos de separación no es un factor decisivo para privilegiar una combinación sobre otra.

7.4.3. Eficiencia de los Algoritmos de Separación

Para evaluar la eficiencia de los algoritmos de separación para encontrar desigualdades violadas tomamos como medida el porcentaje de cortes violados sobre la cantidad de cortes analizados por cada uno de los procedimientos. En la Tabla 7.9 mostramos los resultados para 5 iteraciones del algoritmo de planos de corte en grafos de distinta densidad.

Las desigualdades ***Anula Color*** y ***Camino Multicolor*** tienen posibilidad de estar violadas únicamente en el caso que la variables w_k que se encuentran en la respectiva restricción tienen valor fraccionario. Si esto no ocurre el algoritmo de separación no es llamado. Un signo * en la tabla

	Alta densidad			Media densidad			Baja densidad		
	Total	Separación	%	Total	Separación	%	Total	Separación	%
C1	2363	55	2	1274	46	4	236	33	14
C2	2243	56	2	1381	46	3	236	33	14
C3	2541	64	3	1351	53	4	247	34	14
C4	2722	82	3	1581	60	4	258	36	14
C5	2405	62	3	1275	54	4	248	34	14
C6	3032	83	3	1448	61	4	258	36	14
C7	2720	85	3	1478	56	4	263	40	15
C8	2821	86	3	1494	56	4	259	40	15
C9	2306	111	5	1450	98	7	309	74	24
C10	2327	113	5	1395	99	7	309	74	24
C11	2420	127	5	1483	105	7	327	81	25
C12	3309	179	5	1560	118	8	331	79	24
C13	2493	126	5	1329	103	8	326	81	25
C14	3580	178	5	1626	118	7	331	79	24
C15	3017	192	6	1487	123	8	347	90	26
C16	3147	190	6	1642	124	8	346	90	26
C17	1004	43	4	725	111	15	242	99	41

Tabla 7.8: Tiempo de Separación vs Tiempo Total

indica esta situación. El proceso de separación de la desigualdad ***Camino Multicolor*** tiene buen índice de eficiencia, independientemente de la densidad del *grafo*. Las desigualdades ***Anula Color*** son exploradas por medio de *fuerza bruta* y el porcentaje de las que se encuentran violadas es muy bajo. A pesar de esto, como el procecimiento es muy rápido y su inclusión muestra beneficios en la performance del algoritmo, se justifica su inclusión.

El proceso de separación más ineficiente corresponde a las desigualdades ***p-color Clique***. Es un proceso heurístico y estos porcentajes pueden llevar a concluir que tal vez no sea una buena estrategia la utilizada. Sin embargo, la experimentación hecha sobre grafos con menor cantidad de vértices haciendo una búsqueda más exahustiva no mostró mejores resultados. Todo parece indicar que no son desigualdades que sean frecuentemente violadas por las soluciones óptimas de las relajaciones.

La eficiencia de la separación de ***Agujero*** aumenta a medida que disminuye la densidad del *grafo* y la de ***Cliques*** mantiene un índice parejo independientemente de la densidad.

Son varios los criterios que utilizamos para evaluar las desigualdades válidas: evolución de la cota inferior, tiempos de los algoritmos de separación y un índice de eficiencia. Todos ellos están relacionados y nos inducen a conclusiones que no entran en conflicto. Es indispensable la presencia de las desigualdades ***Clique***, que junto con las ***Anula Color*** y ***Camino Multicolor*** se combinan dando un buen esquema de planos de corte. Las desigualdades ***p-color Clique*** y ***Agujero*** no tienen características que justifiquen su inclusión.

Dens %	Clique	p-Color	Camino	Anula Color	Ciclos
90	31.4	31.4	80.0	0.8	29.5
	29.6	3.9	*	*	26.4
	27.6	0.0	*	*	21.3
	25.9	0.0	*	*	13.6
	21.5	11.6	7.9	*	11.5
70	17.6	17.6	76.0	2.2	83.2
	20.1	0.3	12.6	*	74.6
	22.0	0.0	*	*	73.3
	19.9	0.6	2.0	*	62.1
	20.2	0.0	*	*	53.3
60	16.8	16.8	70.0	3.9	90.4
	20.9	0.2	10.6	*	88.5
	21.5	0.0	21.6	*	83.0
	20.2	0.3	8.0	*	73.4
	20.1	0.0	*	*	69.8

Dens %	Clique	p-Color	Camino	Anula Color	Ciclos
50	20.2	14.6	70.0	1.3	94.9
	24.2	0.1	24.0	*	91.8
	20.7	3.5	40.6	0.0	91.1
	22.4	0.0	*	*	87.0
	20.1	1.5	0.2	*	82.0
40	28.2	3.5	80.0	0.0	95.4
	26.5	0.0	*	*	94.1
	24.7	0.0	*	*	89.3
	19.9	8.1	53.3	0.0	91.2
	23.4	0.0	*	*	85.3
30	33.6	2.1	80.0	*	89.3
	27.9	1.9	80.0	*	87.4
	26.5	0.5	100.0	*	86.4
	25.5	0.0	*	*	85.4
	23.6	0.0	*	*	80.8

Tabla 7.9: Eficiencia Algoritmos de Separación

7.5. ¿Cortes ó *Branching*?

Las decisiones de cúando y por cúantas iteraciones aplicar un algoritmo de planos de corte antes de realizar un *Branching* es un factor crucial en la performance del algoritmo. Los planos de corte son muy efectivos desde el punto de vista del incremento de la cota inferior. Sin embargo, dentro de un esquema *Branch-and-Cut*, hay que lograr un equilibrio entre dicho beneficio y el tiempo requerido. A continuación analizamos posibles alternativas.

7.5.1. Iteraciones del Algoritmo de Planos de Corte

Inicialmente, nuestra cota inferior está dada por el tamaño de una *clique* maximal obtenida por la heurística golosa. Las experiencias presentadas en la sección anterior sobre el mejoramiento de la cota inferior en la relajación inicial, nos inducen a considerar en ***BC-Col*** que vale la pena invertir esfuerzo en el nodo raíz realizando más de una iteración del algoritmo de planos de corte. En relación al tamaño de los *grafos* que consideramos en nuestra experimentación, y teniendo en cuenta los valores de tiempo y las iteraciones en las que se producen los incrementos, 20 iteraciones en el nodo raíz resulta un valor que logra cierto equilibrio entre el beneficio y el tiempo.

Una segunda etapa de nuestro análisis se enfocó en la determinación del número de iteraciones en el resto de los nodos del árbol. Esto es manejado a través del parámetro **IPC**. En general, obtenido un incremento significativo en el nodo raíz, los cambios en el valor de la cota inferior no son tan marcados en el resto de los nodos. Por lo tanto, no se justifica realizar muchas iteraciones del algoritmo de planos de corte salvo para el nodo raíz.

Experimentamos con ***BC-Col*** con diferentes valores de **IPC**. En los Gráficos 7.5 presentamos nuestros resultados con 20 iteraciones en el nodo raíz y 1, 2, 4 y 6 iteraciones en el resto de los nodos del árbol. Consideramos las desigualdades ***Clique***, ***Anula Color*** y ***Camino Multicolor*** como planos de corte. Los resultados están dados sobre el promedio de 10 instancias de densidad

30, 50, 70 y 90% respectivamente. Por la diferencia de escala entre los tiempos de las diversas densidades, normalizamos a 1 el tiempo requerido por *BC-Col* con 2 iteraciones por nodo.

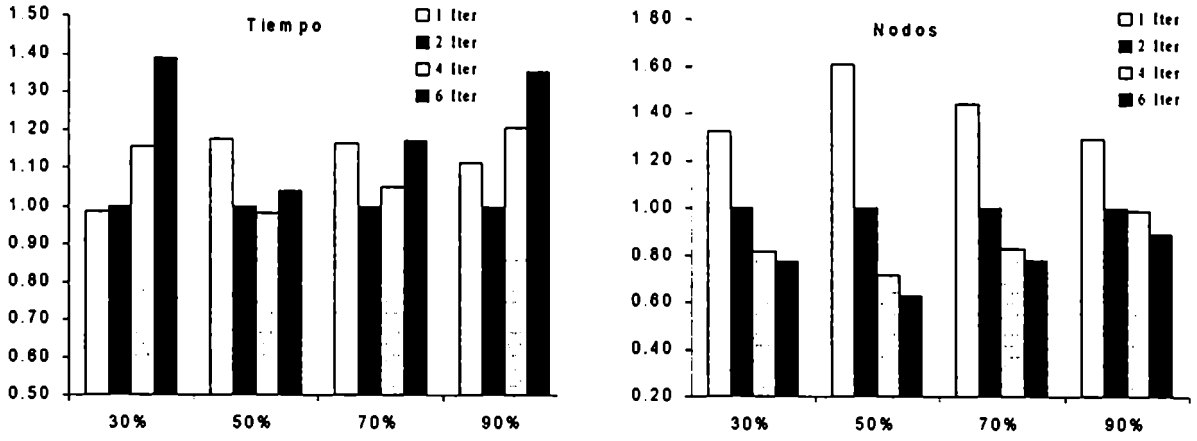


Gráfico 7.5: *Parámetro IPC*

Los gráficos reflejan claramente que a medida que aplicamos más iteraciones del algoritmo de planos de corte, mayor es la reducción en la cantidad de nodos explorados del árbol. Sin embargo, los tiempos de ejecución aumentan, evidenciando que el tiempo de resolver las relajaciones no se ve compensado por la disminución del tamaño del árbol. Con 2 iteraciones del algoritmo de planos de corte se obtiene el equilibrio buscado: reducción de nodos y tiempo. Esta es la opción que mejor performance tiene.

7.5.2. Skip Factor

En una segunda etapa analizamos cuándo aplicar planos de corte. Uno de los criterios más usuales es relacionar esta decisión con los nodos del árbol. Es decir, establecer cada cúantos nodos explorados (*skip factor*) se decide aplicar planos de corte antes del *Branching*. Para poder analizar el comportamiento de nuestro algoritmo frente a diferentes valores del *skip factor* experimentamos con *grafos* de 60 y 70 *vértices* con densidades 30, 50, 70 y 90%. En los Gráficos 7.6 mostramos el tiempo y cantidad de nodos para valores de *skip factor* de 1, 2, 4 y 8 respectivamente. Los resultados son los promedios sobre 10 instancias de cada una de las densidades. Por problemas de escala, normalizamos a 1 los valores correspondientes a *BC-Col* con *skip factor* igual a 1.

Se evidencia un claro dominio del valor 1 para el *skip factor*, ya sea desde el punto de vista del tiempo como del tamaño del árbol de búsqueda. Esta conclusión refuerza nuestra valoración sobre los planos de corte. La influencia sobre las relajaciones justifica usarlos en todos los nodos.

7.6. Comparando Relajaciones

En el Capítulo 4 definimos 4 poliedros: $\mathcal{SCP}$, $\mathcal{CP}$, $\mathcal{CP}'$ y $\mathcal{CP}''$. El poliedro $\mathcal{SCP}$ está asociado a la formulación clásica del problema de coloreo. Los poliedros $\mathcal{CP}$, $\mathcal{CP}'$ y $\mathcal{CP}''$ corresponden al conjunto

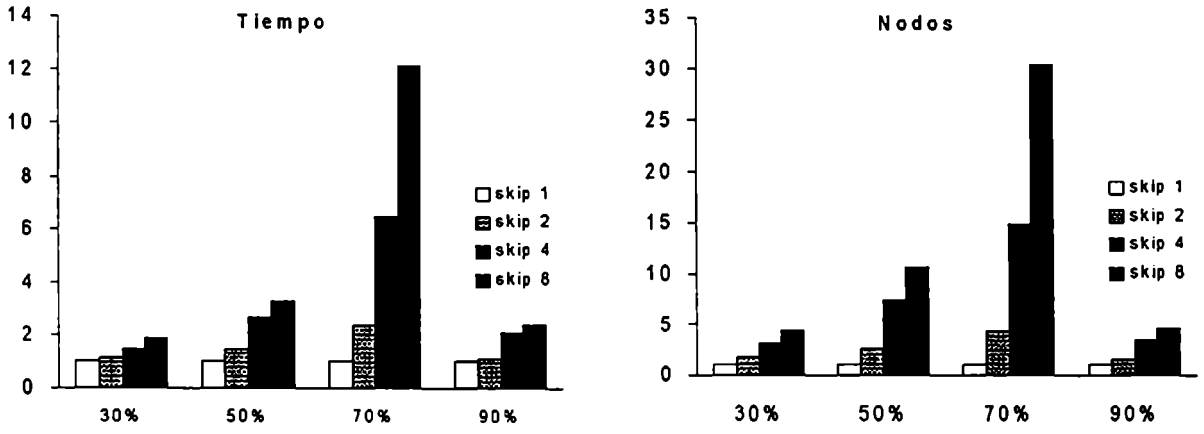


Gráfico 7.6: *Skip Factor*

de soluciones de los tres modelos que presentamos con diferentes criterios para la eliminacion de simetría.

Desde el punto de vista del estudio poliedral, ya hemos señalado la dificultad que presentan los poliedros $\mathcal{CP}'$ y $\mathcal{CP}''$. Sin embargo, como ambos poliedros están incluídos en $\mathcal{CP}$, las desigualdades válidas que surgieron de nuestro estudio también lo son para estos poliedros. Por otro lado, algunas desigualdades válidas que derivamos para $\mathcal{SCP}$ en [18] son compartidas con $\mathcal{CP}$. Por ejemplo, las desigualdades ***Clique***.

De acuerdo a nuestra experiencia, la inclusión de las desigualdades ***Clique*** en un algoritmo de planos de corte resulta fundamental para mejorar la cota inferior que otorga el valor óptimo de la relajación lineal. Por este motivo, nos parece razonable utilizar estas desigualdes en un algoritmo de planos de corte para comparar las diferentes relajaciones.

Utilizamos *grafos* al azar de 125 *vértices* y de densidades 30, 50, 70 y 90 %. En las 4 formulaciones fijamos la coloración de la *clique* maximal. Las restricciones que eliminan soluciones simétricas fueron consideradas para los *vértices* restantes y para los colores con índice superior a $\hat{\omega}$. Hicimos 50 iteraciones de un algoritmo de planos de corte con las desigualdades ***Clique***. En los Gráficos 7.7 mostramos la evolución del la función objetivo para las 4 relajaciones.

La relajación lineal de $\mathcal{SCP}$ es la de peor comportamiento y no alcanza a obtener los mismos valores de la cota inferior. La gran cantidad de soluciones simétricas que se encuentran en el poliedro son la causa del lento progreso en el incremento de la función objetivo e incluso del tiempo que demora cada iteración del algoritmo de planos de corte.

Las otras tres relajaciones tienen un comportamiento similar desde el punto de vista del mejoramiento de la cota inferior. Sin embargo, el tiempo de resolución de las relajaciones en cada iteración es superior para $\mathcal{CP}''$. Esto resulta lógico pues $\mathcal{CP}''$ tiene $(n-\hat{\omega})(\hat{\chi}-\hat{\omega})$ restricciones más que $\mathcal{CP}$.

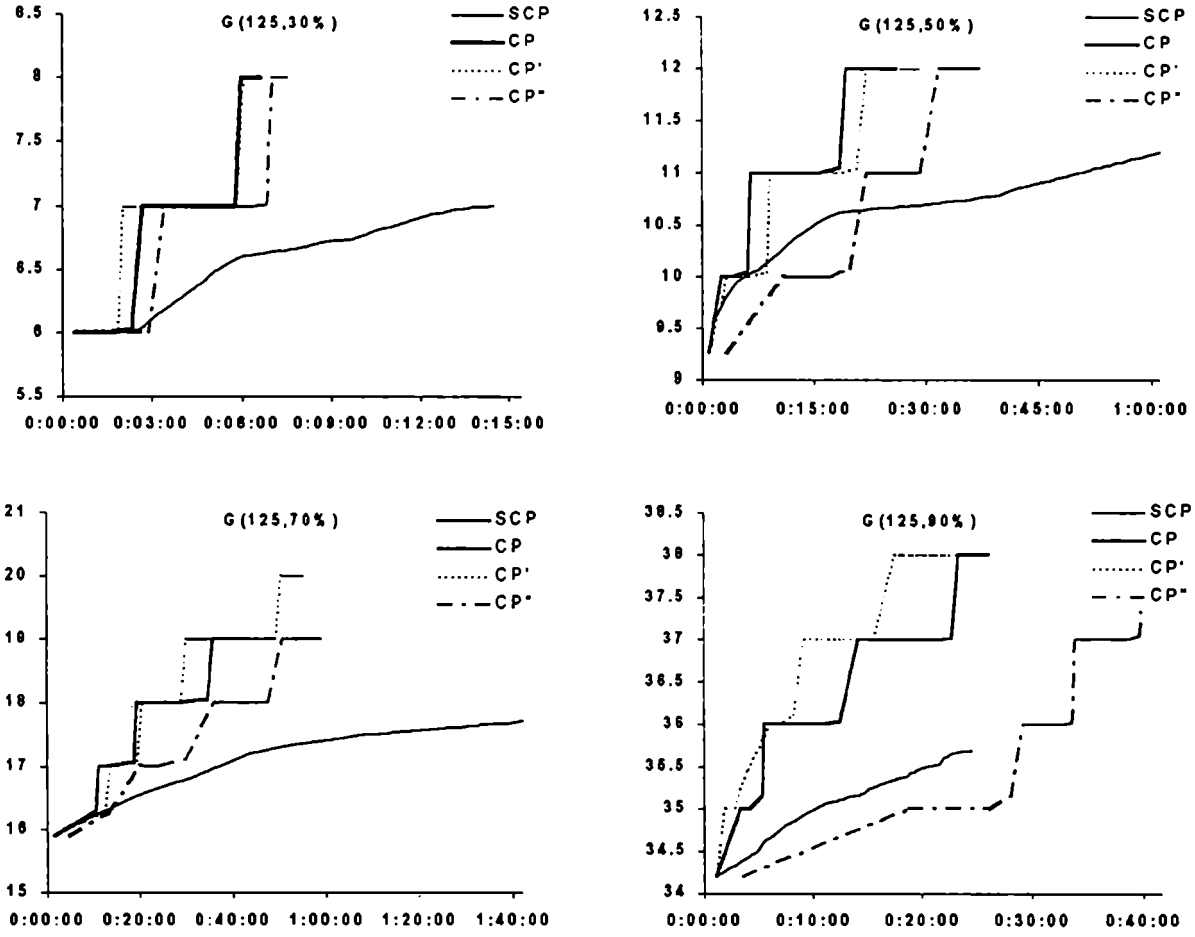


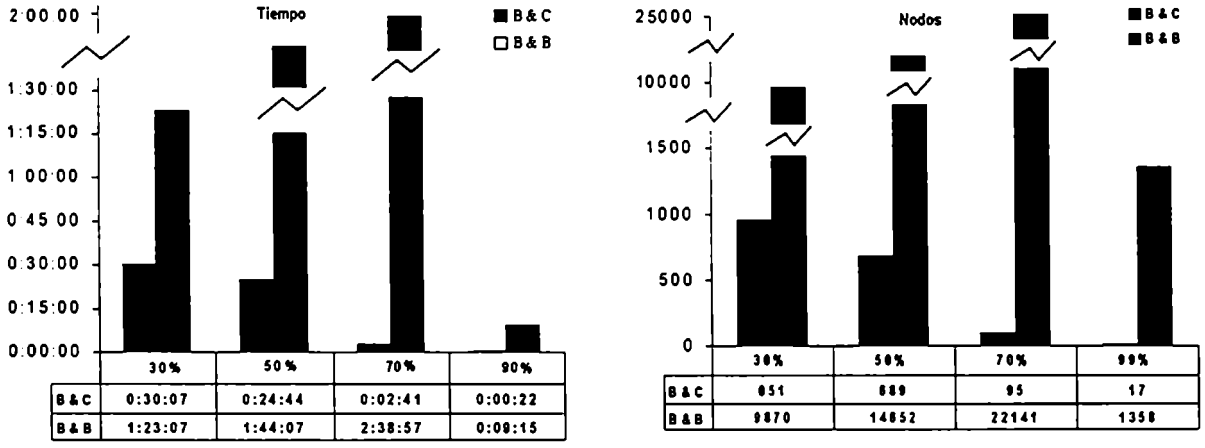
Gráfico 7.7: Comparando Relajaciones

En algunos casos, la relajación de $\mathcal{CP}'$ superó a $\mathcal{CP}$. Este comportamiento se evidencia generalmente en *grafos* donde $\hat{\omega}$ y $\hat{\chi}$ tienen gran diferencia. En vistas de este comportamiento, implementamos un *Branch-and-Cut* usando la relajación de $\mathcal{CP}'$. La mayor dificultad la encontramos en la etapa de *Branching*. Después de elegir un *vértice* del *grafo*, nuestro algoritmo abre un hijo para cada color factible entre los ya usados y un nuevo color en algunos casos. Las restricciones que eliminan soluciones simétricas en $\mathcal{CP}'$ imponen que el conjunto de *vértices* coloreados con el color j tenga cardinal mayor o igual al del conjunto de los *vértices* pintados con $j+1$. Por lo tanto, si aplicamos nuestra estrategia de *Branching* podemos estar eliminando las soluciones factibles de $\mathcal{CP}'$. Para asegurarnos de no perder las soluciones factibles, debemos abrir un hijo por cada color factible entre 1 y $\hat{\chi}$. De esta manera se construye un árbol con más ramas que las que surgen con la relajación de $\mathcal{CP}$. Esta diferencia se acentúa justamente en los *grafos* donde la relajación de $\mathcal{CP}'$ es superior a la de $\mathcal{CP}$. Nuestra experiencia nos mostró que a pesar que la relajación es mejor, no alcanza a compensar el incremento de tiempo causado por el aumento en el tamaño del árbol. Tampoco trajo buenos resultados la estrategia de dicotomoía en una variable. A pesar de los

resultados negativos de considerar a $\mathcal{CP}'$ dentro de un esquema *Branch-and-Cut*, rescatamos esta relajación para aquellos casos en los cuales buscamos sólo cotas inferiores aplicando un algoritmo de planos de corte.

7.7. *Branch-and-Bound* vs *BC-Col*

Si bien de alguna manera se desprende del análisis que hacemos sobre el *skip factor*, nos parece interesante remarcar el beneficio de incluir planos de corte. Para eso comparamos nuestro algoritmo *BC-Col* con *skip factor* igual a 1 con un algoritmo *Branch-and-Bound*. La comparación está hecha sobre la misma implementación, simplemente no llamamos a las rutinas de separación. En los Gráficos 7.8 queda evidenciada la mejor performance del algoritmo que incluye los planos de corte. Estos resultados son los promedios sobre 10 instancias para cada densidad.



	30%	50%	70%	90%
B&C	0:30:07	0:24:44	0:02:41	0:00:22
B&B	1:23:07	1:44:07	2:38:57	0:09:15

	30%	50%	70%	90%
B&C	851	889	95	17
B&B	9870	14852	22141	1358

Gráfico 7.8: *Branch-and-Bound vs BC-Col*

La reducción obtenida por *BC-Col* en los tiempos de ejecución es notable. Se logran resolver instancias que sin los planos de corte superan el límite de 2hs. Cabe señalar que para poder obtener resultados con *Branch-and-Bound* dentro de las 2 hs tuvimos que reducir el tamaño de los grafos de experimentación a 50 y 60 *vértices*. Para *grafos* con mayor cantidad de *vértices*, la inclusión de planos de corte resulta esencial para resolver el problema.

7.8. CPLEX vs *BC-Col*

Al implementar un *Branch-and-Cut* es natural plantearse la comparación contra algún algoritmo de propósito general. ¿Vale la pena todo el esfuerzo invertido en una implementación ad-hoc? ¿Por qué no usar la implementación de un buen paquete de optimización? Para responder a esta pregunta, utilizamos CPLEX [23] sobre el que corrimos instancias de 50 y 60 *vértices* con densidades 30, 50 70 y 90 %. En la Tabla 7.10 presentamos los resultados correspondientes a 10 instancias para cada densidad. Reportamos los tiempos y cantidad de nodos para cada uno de los algoritmos. Un signo ***** en la tabla indica que la instancia no pudo ser resuelta dentro del límite impuesto

de 2 hs. Las corridas con CPLEX se realizaron utilizando toda las ventajas que el paquete ofrece:

30% Densidad				50% Densidad			
BC-Col		CPLEX		BC-Col		CPLEX	
Tiempo	Nodos	Tiempo	Nodos	Tiempo	Nodos	Tiempo	Nodos
98	100	645	1236	239	247	*****	*****
306	128	*****	*****	492	348	*****	*****
45	36	303	568	310	397	6468	8134
12	7	137	324	33	103	589	1106
75	21	1046	586	155	126	*****	*****
97	41	*****	*****	3	1	4	4
9	1	7	4	56	56	*****	*****
14	16	54	128	164	191	*****	*****
86	78	322	492	172	75	*****	*****
113	27	*****	*****	62	41	*****	*****
70% Densidad				90% Densidad			
BC-Col		CPLEX		BC-Col		CPLEX	
Tiempo	Nodos	Tiempo	Nodos	Tiempo	Nodos	Tiempo	Nodos
62	116	1389	1979	8	7	22	438
44	106	71	125	7	5	10	237
31	75	85	206	29	45	*****	*****
39	49	*****	*****	12	9	678	9929
53	132	56	152	8	7	36	903
37	33	*****	*****	3	8	5	198
35	38	*****	*****	3	5	3	67
31	44	*****	*****	24	26	371	2935
20	30	7668	26277	0	4	0	1
16	28	1347	3773	2	20	1	59

Tabla 7.10: CPLEX vs BC-Col

pre-procesamiento, cortes *clique* y cortes *cover*. La estrategia de búsqueda es en profundidad y el *Branching* dicotómico en la variable *más infactible*.

La ventaja de ***BC-Col*** se acentúa en *grafos* de densidades medias, donde el CPLEX superó el tiempo límite en la mayoría de las instancias.

Si comparamos la relación tiempo-cantidad de nodos explorados, CPLEX genera mayor cantidad de nodos por unidad de tiempo. ***BC-Col*** tiene una relación mucho menor. Esto se debe a varios factores. Los algoritmos de separación y las iteraciones del algoritmo de cortes aumentan el tiempo invertido en cada subproblema. En ***BC-Col*** implementamos, gracias a cierta flexibilidad del entorno ABACUS, estructuras y procedimientos específicos para el problema. Estos facilitan la generación de cortes y la definición de las estrategias de recorrido y *Branching* pero traen como consecuencia un procedimiento de manejo del árbol mucho más lento. Sin embargo, los resultados son más que elocuentes. A pesar de la robustez y eficiencia de la implementación de CPLEX, los planos de corte y todas las estrategias específicas que desarrollamos para ***BC-Col*** conforman un algoritmo exitoso.

7.9. Resultados Finales

Después de analizados los factores que consideramos importantes en nuestra implementación, estamos en condiciones de presentar nuestro resultados finales. Para determinar la eficiencia de ***BC-Col***, comparamos contra el algoritmo **DSATUR** [10] con la modificación propuesta por Sewell [73]. Usamos el código de Mikel Trick disponible en http://mat.gsia.cmu.edu/COLOR/solvers/trick.c Los valores de los parámetros y los criterios usados en ***BC-Col*** son:

- *IPC-Iteraciones de planos de corte:* Nodo raiz:20 Otros nodos:3
- *Skip factor:* 1

- *Selección de variable de Branching:* VB2
- *Estrategia de recorrido:* O2 + Profundidad
- *Planos de corte: Clique + Anula Color +Camino Multicolor*
- *Máxima cantidad de cortes por iteración:* 300 + 100 + 300
- *Tiempo limite:* 2hs

Experimentamos con *grafos* generados al azar y la librería de instancias DIMACS. Los *grafos* generados al azar están presentes en todos los trabajos de la literatura sobre algoritmos para el *problema de coloreo de grafos.* La experiencia reportada sobre ellos los caracteriza como los *grafos* más difíciles de colorear. En general, si se trata de un algoritmo exacto, se limitan a instancias menores a 70 *vértices* acentuándose la dificultad en el caso de *grafos* de densidad media. Compartimos esta apreciación, aunque pudimos trabajar con grafos de hasta 90 *vértices.*

G(80,40)		G(70,50)		G(75,50)		G(70,70)		G(75,70)	
DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col
939	343	560		440	2135	247	85	3443	597
*****	3835	1814	742	6027	1631	767	65	5171	2978
4209	1291	125	94	866	5754	2197	449	*****	1778
216	192	481	91	*****	6393	1559	511	*****	2058
*****	698	1023	302	2037	3545	1010	362	1389	545
6447	2308	72	336	580	3294	*****	674	1443	576
720	347	383	205	5825	4414	290	34	3200	2087
*****	4953	767	203	5655	4141	1208	238	*****	1933
195	206	4065	950	995	994	286	57	*****	1914
*****	2015	268	213	2181	429	1057	186	5975	*****
G(125,10)		G(80,20)		G(80,30)		G(80,90)		G(90,90)	
DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col
48	221	2	40	308	228	30	10	24	5
5240	2767	1	19	12	99	14	6	4043	2146
1	86	3	66	152	177	284	50	162	69
61	43	18	51	3778	1187	11	2	*****	*****
30	64	8	44	15	85	1173	2764	*****	*****
4081	830	6	56	101	337	60	13	*****	5496
120	685	2	70	588	135	6291	1114	*****	395
2199	1280	1	57	152	153	353	70	1174	284
411	180	5	11	8	106	*****	1840	3890	1065
4	63	4	63	182	128	540	160	*****	3307

Tabla 7.11: BC-Col en *Grafos* al Azar

Los resultados muestran que, en general, los *grafos* al azar de densidad baja (menor al 30 %) se resuelven más rápidamente con un esquema de enumeración completa. Esto no sólo se debe a la buena performance de DSATUR, sino también al bajo impacto que tienen los planos de corte en *BC-Col.* Ya hemos visto el rol fundamental de los cortes *Clique* para el mejoramiento de la cota inferior. En grafos al azar de baja densidad, el impacto en la evolución de la cota disminuye por varias razones: las cotas inferior y superior iniciales no tienen una marcada diferencia, las *cliques* no son de gran tamaño y $\hat{\omega}$ difiere poco de $\chi(G)$. Esto trae como consecuencia que el proceso de *Bound* no sea tan efectivo y el árbol de búsqueda no pueda ser podado significativamente.

Los grafos de media densidad (entre el 30 % y 70 %) son los más difíciles para DSATUR. En estos grafos se presentan mayores diferencias entre las cotas iniciales y entre $\chi(G)$ y $\hat{\omega}$. La búsqueda en el árbol de enumeración es más exhaustiva y el número de subproblemas explorados aumenta exponencialmente. Con esta densidad, instancias que no pueden ser resueltas dentro del tiempo límite de 2hs, surgen en grafos de *apenas* 70 *vértices.* Esta medida nos da una clara idea de la dificultad de

esta clase de *grafos*. Por el contrario, ***BC-Col*** muestra muy buena performance, pudiendo resolver instancias que con DSATUR no fue posible. Esta situación es más frecuente a medida que aumenta la densidad del *grafo*.

Los grafos de alta densidad (más de 70 %) son los que muestran un comportamiento más errático, encontrándose instancias que se resuelven en pocos segundos y otras que superan las 2hs. No encontramos una explicación convincente de este fenómeno. La performance de ***BC-Col*** fue muy buena, incluso resolviendo instancias a nivel del nodo raíz.

A modo de resumen, en la Tabla 7.12 mostramos el tiempo promedio sobre las instancias resueltas por los dos algoritmos y el porcentaje de instancias que superaron las 2hs de CPU para cada uno de los algorimtos. Cabe destacar el bajo porcentaje de instancias no resueltas por ***BC-Col*** comparado con DSATUR y la buena performance reflejada en los tiempos promedio.

G(80,40)		G(70,50)		G(75,50)		G(70,70)		G(75,70)	
Tiempo Promedio									
DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col
2121	866	956	348	2734	2926	958	221	3437	1357
Porcentaje de instancias no resueltas									
40	0	0	0	10	0	10	0	40	10

G(125,10)		G(80,20)		G(80,30)		G(80,90)		G(90,90)	
Tiempo Promedio									
DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col	DSATUR	BC-Col
606	1220	7	48	530	264	973	465	1859	714
Porcentaje de instancias no resueltas									
0	0	0	0	0	0	10	0	50	20

Tabla 7.12: Tiempos Promedio sobre *Grafos* al Azar

DIMACS es un centro de investigación producto de la colaboración de Rutgers University, Princeton University, AT&T Labs Research, Bell Labs and NEC Research Institute. Entre sus objetivos se encuentra fomentar el análisis experimental de algoritmos. En particular, en 1993, se realizó un encuentro-desafío sobre algoritmos para el *problema de clique máxima*, *coloreo* y *satisfactibilidad* [29]. Con este propósito, se reunió una colección de instancias de prueba provenientes de diversas fuentes. En el reciente *2002 Computational Symposium: Graph Coloring and its Generalizations*, este conjunto de instancias se actualizó y se usó para la comparación de los distintos algoritmos. En dicho congreso presentamos parcialmente los resultados computacionales de nuestro algoritmo, siendo éstos muy competitivos [64].

Dividimos nuestra presentación en dos grupos. La Tabla 7.13 corresponde a las instancias que pudieron ser resueltas dentro del límite de 2 horas de CPU. Las primeras 10 se resolvieron en el nodo raíz, sin necesidad de ingresar en un proceso de *Branching*. El resto de las instancias requirió de una exploración del árbol. La Tabla 7.14 corresponde a instancias en las que no pudo alcanzarse la optimalidad dentro de las 2 horas de CPU. En estos casos, reportamos la mejor cota inferior y

superior obtenida dentro de ese tiempo por cada uno de los algoritmos.

Los *grafos* sobre los cuales *BC-Col* no pudo obtener ningún resultado tienen alguna de las siguientes propiedades:

- Grafos al azar con más de 500 *vértices*: Son instancias muy difíciles de resolver y que presentaron problemas de memoria. Están pensadas para experimentar con heurísticas.
- Grafos con más de 1000 *vértices*: para experimentación con heurísticas.
- Grafos Reina: son grafos muy regulares de densidad media (ver referencia en Tabla 7.2. No hemos encontrado en la literatura algoritmos exactos que resuelvan el problema para más de 80 *vértices* .

Problema	n	m	n_cli	$\hat{\chi}$	χ	BC-Col	DSATUR
DSJC125_1	125	736	4	5	5	2.00	0.10
fpsol2_i_1	496	11654	55	65	65	29.00	0.30
fpsol2_i_2	451	8691	29	30	30	9.00	0.20
fpsol2_i_3	425	8688	29	30	30	9.00	0.20
miles1000	128	3216	41	42	42	0.02	0.10
miles1500	128	5198	71	73	73	0.14	0.10
ash331GPIA	662	4185	3	4	4	2719.0	1.70
3-FullIns_3	80	346	5	6	6	1.00	*****
4-FullIns_3	114	541	6	7	7	3.00	*****
5-FullIns_3	154	792	7	8	8	3.00	*****
mug88_1	88	146	3	4	4	485.00	*****
mug88_25	88	146	3	4	4	1690.0	*****
mug100_1	100	166	3	4	4	4029.0	*****
mug100_25	100	166	3	4	4	5498.0	*****
3-Insertions_3	56	110	2	4	4	10.00	12.70
1-FullIns_4	93	593	3	5	5	703.00	*****
2-FullIns_3	52	201	4	5	5	3.00	2558
queen8_8	64	728	8	10	9	96.00	46.00

Tabla 7.13: Instancias DIMACS: Solución Óptima

Problema	n	m	n_cli	$\hat{\chi}$	χ	BC-Col		DSATUR	
						Inf	Super	Inf	Super
DSJC125_5	125	3891	9	20	?	13	20	9	19
DSJC125_9	125	6961	32	47	?	42	47	29	45
DSJC250_1	250	3218	4	9	?	5	9	4	9
DSJC250_5	250	15668	11	36	?	13	36	9	35
DSJC250_9	250	27897	38	88	?	47	88	34	85
DSJR500_1c	500	121275	72	87	?	76	88	70	88
queen9_9	81	2112	9	11	10	9	10	9	10
myciel6	95	755	2	7	7	5	7	2	7
myciel7	191	2360	2	8	8	5	8	2	8
1-Insertions_5	202	1227	2	6	?	4	6	2	6
1-Insertions_6	607	6337	2	7	?	4	7	2	7
2-Insertions_4	149	541	2	5	?	4	5	2	5
2-Insertions_5	597	3936	2	6	?	3	6	2	6
3-Insertions_4	281	1046	2	5	?	3	5	2	5
3-Insertions_5	1406	9695	2	6	?	3	6	2	6
4-Insertions_3	79	156	2	4	4	3	4	2	4
4-Insertions_4	475	1795	2	5	?	3	5	2	5
1-FullIns_5	282	3247	3	6	?	4	6	3	6
2-FullIns_4	212	1621	4	6	?	5	6	4	6
2-FullIns_5	852	12201	4	7	?	5	7	4	7
3-FullIns_4	405	3524	5	7	?	6	7	5	7
3-FullIns_5	2030	33751	5	8	?	6	8	5	8
4-FullIns_4	690	6650	6	8	?	7	8	6	8

Tabla 7.14: Instancias DIMACS: Mejoramiento de Cotas

En este Capítulo hemos presentado nuestra experiencia computacional con ***BC-Col*** que demuestra su efectividad tanto en *grafos* generados al azar como en instancias provenientes de aplicaciones de la vida real. La mayoría de los algoritmos usados para el *problema de coloreo de grafos* son procedimientos heurísticos. Estos brindan cotas superiores del número cromático. La cota inferior que suele considerarse es el tamaño de una *clique* maximal. En muchos casos, la diferencia entre las cotas es grande y no sirve como criterio para saber si la heurística es buena. ***BC-Col*** es un algoritmo exacto que mejora las cotas inferior y superior iniciales. De esta manera, aunque no alcance la optimalidad dentro de un tiempo límite permite reducir el intervalo donde se encuentra el valor óptimo. Esto da una herramienta más eficiente para medir la calidad de la cotas.

Capítulo 8

Conclusiones

En esta tesis abordamos la resolución del *problema de coloreo de grafos* utilizando modelos de programación lineal entera binaria. Nuestro principal objetivo fue superar una importante dificultad que tiene este tipo de enfoque para muchos problemas combinatorios: la simetría en el conjunto de soluciones factibles.

Debido a la indistinguibilidad de los colores, hay un gran número de coloreos del grafo que son equivalentes o simétricos, es decir utilizan la misma cantidad de colores. Si todos estos coloreos se encuentran representados en el conjunto de soluciones factibles del modelo de programación lineal entera, esta simetría se traslada al modelo. Esta propiedad influye negativamente en la performance de la principal herramienta para resolver estos modelos: los algoritmos *Branch-and-Cut*. Con el propósito de mejorar esta situación trabajamos en varios puntos: modelaje, estudio poliedral y desarrollo de un algoritmo *Branch-and-Cut* que denominamos *BC-Col*.

Propusimos nuevos modelos que, con distintos criterios, eliminan parcialmente soluciones simétricas. Analizamos sus ventajas y desventajas tanto desde un punto de vista poliedral como algorítmico. De los tres modelos propuestos en esta tesis, uno de ellos resultó ser el más conveniente para realizar un estudio poliedral. Los otros dos modelos tienen asociados poliedros muy complejos que dependen de algunas propiedades del *grafo* que hacen muy difícil su caracterización. Gran parte de este trabajo estuvo abocado a realizar este estudio. Encontramos varias familias de desigualdades válidas que bajo ciertas condiciones definen facetas del poliedro. Claramente no hemos logrado una caracterización completa. Debido a la complejidad del problema, lejos estaba éste de ser nuestro objetivo.

Una de las desigualdades obtenidas que tiene un rol fundamental desde el punto de vista algorítmico es la desigualdad *Clique*. Estas desigualdades son válidas para todos los poliedros, incluso para el asociado al modelo clásico. Esto nos permitió comparar las relajaciones lineales usando un algoritmo de planos de corte. La evolución de la cota inferior en los nuevos modelos es mejor y más rápida.

La segunda parte del trabajo estuvo centrada en el desarrollo del algoritmo *BC-Col*. Es un algoritmo *Branch-and-Cut* en el que tuvimos en cuenta factores que consideramos esenciales para

una buena performance.

- Una etapa inicial de preprocesamiento que reduce el número de *vértices* del *grafo*, permitiendo resolver *grafos* de mayor tamaño.
- Heurísticas iniciales para el cálculo de cotas inferiores y superiores que reducen el espacio de búsqueda y la cantidad de variables y restricciones del modelo.
- Disminución de la cantidad de restricciones de la relajación lineal, reeemplazando las desigualdades de adyacencia por una versión relajada de las desigualdades de *Vecindad*. De esta manera se reducen los tiempos de resolución de las relajaciones en cada nodo del árbol.
- Procedimientos de separación rápidos y eficientes para varias de las familias de desigualdades válidas obtenidas de nuestro estudio poliedral.
- Estrategias de selección de variable de *branching* y recorrido del árbol que guian la búsqueda evitando explorar sobre soluciones simétricas.

Para cada uno de estos factores consideramos distintas alternativas. Mediante la experimentación con grafos generados al azar y de la recopilación de instancias DIMACS, buscamos identificar aquellas alternativas que brindan la mejor performance. En algunos casos surgió claramente cual es la mejor opción. En otros, no pudimos llegar a una conclusión determinante. Esta situación es comprensible por la diversidad de estructuras que tienen los grafos. Ciertas propiedades del *grafo* favorecen la buena performance del algoritmo y otras lo dificultan. Las desigualdades válidas surgidas del estudio poliedral constituyen un factor decisivo en la performance del algoritmo implementado. La incorporación de las mismas en la etapa de *cutting* permite mejorar sustancialmente el valor de la cota inferior y reducir el tamaño del árbol de búsqueda. La comparación que hicimos en el Capítulo 7 con *Branch-and-Bound* da muestras más que evidentes de esta afirmación donde observamos una reducción notable en los tiempos de ejecución. Las estrategias propuestas para selección de variable de *Branching* y recorrido son claves para la generación de un árbol de menor tamaño. La experimentación con los criterios clásicos de dicotomía mostraron muy mala performance.

BC-Col es un algoritmo exacto que tiene la característica de ir mejorando las cotas inferior y superior durante el tiempo de ejecución. De esta manera, aún en el caso que no logremos alcanzar el óptimo dentro del tiempo límite establecido, tenemos un intervalo donde tenemos garantía que se encuentra el número cromático. Esto es muy importante para evaluar la calidad de una solución obtenida por uan heurística.

Nuestro trabajo sobre el estudio poliedral y el algoritmo ***BC-Col*** deja espacio para futuros estudios:

- Usando ZeroOne [59], un programa que enumera las soluciones 0-1 de un poliedro y PORTA [15], un programa que calcula la cápsula convexa de un conjunto de puntos de coordenadas 0-1, hemos obtenido la caracterización completa del poliedro para *grafos* muy pequeños. Los resultados nos confirman que aún hay muchas familias de facetas para estudiar. También hemos encontrado instancias para las cuales la descripción con una o varias de las desigualdades obtenidas es completa. Queda abierta la identificación de familias de *grafos* con esta propiedad.

- Nuestro análisis del algoritmo evidencia el rol fundamental de los planos de corte. El desarrollo de algoritmos de separación para otras desigualdades ó incluso mejorar los propuestos puede lograr una mejor performance de ***BC-Col***.

- En nuestra experimentación hemos encontrado instancias más difíciles de colorear que otras. ¿Qué es lo que provoca la marcada diferencia de performance de *BC-Col* entre una instancia y otra? ¿Por qué los *grafos* generados al azar son tan difíciles? y los *Reina*? Sería muy útil entender las causas de las dificultades para mejorar el algoritmo.

- Los modelos de programación lineal entera y el estudio poliedral de esta tesis pueden servir como base para problemas de coloreo generalizado. En [62] definimos el *problema de k-i coloreo de grafos*. Un $k-icoloreo$ asigna a cada *vértice* del grafo k colores de manera tal que no comparte más de i colores con cada uno de sus *vértices* adyacentes. El *problema de k-i coloreo* es determinar la mínima cantidad de colores necesarias para un *k-i colorear* un grafo. En [62] presentamos algunas propiedades teóricas del problema y un modelo de programación entera. No hay en la literatura un estudio poliedral del modelo y tampoco existe un algoritmo *Branch-and-Cut* para resolverlo. Consideramos que los resultados de esta tesis facilitarán el estudio del problema y el desarrollo e implementación de un algoritmo.

El *problema de coloreo de grafos* ha sido tratado por muchos investigadores y día a día aparecen nuevos resultados que aumentan el conocimiento del problema. Esperamos haber contribuido a despertar el interés para trabajar en este área y estimular la discusión de las líneas de investigación presentes y futuras.

Bibliografía

[1] K. Aardal, A. Hipolito, C. van Hoesel, and B. Jansen, *A branch-and-cut algorithm for the frequency assignment problem*, Tech. report, Maastricht University, 1996.

[2] K. Appel and W. Haken, *Every planar map is four colorable, part 1: Discharging*, Illinois J. Math **21** (1977), 429–490.

[3] K. Appel, W. Haken, and J. Koch, *Every planar map is four colorable, part 2: Reducibility*, Illinois J. Math **21** (1977), 491–567.

[4] A. Atamturk, G.Ñemhauser, and M. Savelsbergh, *Conflict graphs in integer programming*, manuscript, March 1998.

[5] E. Balas, *An additive algorithm for solving linear programs with zero-one variables*, Operations Research **13** (1965), 517–546.

[6] E. Balas, S. Ceria, and G. Cornuejols, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming **58** (1993), 295–324.

[7] I. Baybars, *Optimal assignment of broadcasting frequencies*, European Journal of Operations Research **9** (1982), 257–263.

[8] J. Beasley, *Lagrangean relaxation, ch. 6*, Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, 1993.

[9] R. Borndörfer, *Aspects of set packing, partitioning and covering*, Ph.D. thesis, Technischen Universitat Berlin, 1999.

[10] D. Brélaz, *New methods to color vertices of a graph*, Communications of the ACM **22** (1979), 251–256.

[11] P. Briggs, K. Cooper, K. Kennedy, and L. Torczon, *Coloring heuristics for register allocation*, ASCM Conference on Program Language Design and Implementation, 1989, pp. 275–284.

[12] G. Chaitin, M. Auslander, A. Chandra, J. Cocke, M. Hopkins, and P.Markstein, *Register allocation via coloring*, Computer Languages **6** (1981), 47–57.

[13] M. Chams, A.Hertz, and D. de Werra, *Some experiments with simulated annealing for coloring graphs*, European Journal of Operational Research **32** (1987), 260–266.

[14] F. Chow and J. Hennessy, *The priority-based coloring approach to register allocation*, ACM Transactions on Programming Languages and Systems **12** (1990), 501–536.

[15] T. Christof, A. Löbel, and M. Stoer, *Porta-apolyhedron representation transformation algorithm. version 1.3.2*, 1997, Konrad-Zuse-Zentrum für Infortionstechnik (ZIB), Berlin.

[16] N. Christofides, *An algorithm for the chromatic number of a graph*, Computer Journal **14** (1971), 38–39.

[17] ———, *Graph theory: An algorithm approach*, Academic Press, London, 1975.

[18] P. Coll, J. Marenco, I. Méndez-Díaz, and P. Zabala, *An integer programming model for the graph coloring problem*, Annals of Operations Research Letters **116** (2002), 79–90.

[19] COLORLIB, *Dimacs benchmark graphs*, http://mat.gsia.cmu.edu/COLOR02/.

[20] S. A. Cook, *The complexity of theorem-proving procedures*, 3rd Annual ACM Symposium on Theory of Computing Machinery, 1971, pp. 151–158.

[21] D. Corneil and B. Graham, *An algorithm for determinig the chromatic number of a graph*, SIAM Journal on Computing **2** (1973), 311–318.

[22] D. Costa, A.Hertz, and O. Dubuis, *Embedding of a sequential procedure within an evolutionary algorithm for coloring problems in graphs*, Journal of Heuristics **1** (1995), 105–128.

[23] CPLEX, *Linear optimization 6.0 with mixed integer and barrier solvers, ilog*, 1997-1998.

[24] H. Crowder, E. Johnson, and M. Padberg, *Solving large scale zero-one linear programming problems*, Operations Research **31** (1983), 803–834.

[25] R. Dakin, *A tree search algorithm for mixed-integer programming problems*, Computer Journal **8** (1965), 250–255.

[26] G. Dantzig, D. Fulkerson, and S. Johnson, *Solution of a large scale traveling salesman problem*, Operations Research **2** (1954), 393–410.

[27] D. de Werra, *An introduction to timetabling*, European Journal of Operations Research **19** (1985), 151–162.

[28] J. Desroisiers, Y. Dumas, M. Solomon, and F. Soumis, *Time constrained routing and scheduling*, Handbook in Operations Research and Management Science American Mathematics Society. Network Routing (C. Monma M. Ball, T. Magnanti and G.Ñemhauser, eds.), vol. 8, Elsevier, Amsterdam, 1995, pp. 35–140.

[29] DIMACS, *Center for discrete mathematics and theoretical computer science*, http://dimacs.rutgers.edu.

[30] R. Dorne and J. Hao, *Tabu search for graph coloring, t-coloring ans set t-coloring*, Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization (S., S. Martello, I. Osman, and C. Roucairol, eds.), Kluwer, Boston, 1999, pp. 33–47.

[31] T. Feo and M. Resende, *A probabilistic heuristic for a computationally difficult set covering problem*, Operations Research Letters **8** (1989), 67–71.

[32] ———, *Greedy randomized adaptive search procedures*, Journal of Global Optimization **2** (1995), 1–27.

[33] C. Fleurent and J. Ferland, *Genetic and hybrid algorithms for graph coloring*, Annals of Operations Research **63** (1996), 437–464.

[34] M. Garey and D. Johnson, *Computers and intractability - a guide to the theory np-completeness*, W. H. Freeman, 1979.

[35] P. Gilmore and R. Gomory, *A linear programming approach to the cutting stock problem*, Operations Research 9 (1961), 849–859.

[36] F. Glover, *Tabu search, part 1*, ORSA-Journal on Computing **1** (1989), 190–206.

[37] R. Gomory, *Outline of an algorithm for integer solution to linear programs*, Bulletin Amaerican Mathematical Society **64** (1958), 275–278.

[38] J. Gonzalez-Velarde and M. Laguna, *Tabu search with simple ejection chains for coloring graphs*, manuscript, February 2002.

[39] M. Grötschel, M. Junger, and G. Reinelt, *A cutting plane algorithm for the linear ordering problem*, Operations Research **32** (1984), 1195–1220.

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

[41] M. Grötschel, M. Lovasz, and G. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer Verlag, Berlin, 1988.

[42] M. Grötschel and M. Padberg, *On the symmetric traveling salesman problem i: inequalities*, Mathematical Programming **16** (1979), 265–280.

[43] W. Hale, *Frequency assignment: Theory and applications*, Proceedings of the IEEE, vol. 68, 1980, pp. 1497–1514.

[44] P. Heawood, *Map-colour theorem*, Quart. J. Pure Appl. Math **24** (1890), 332–338.

[45] A. Hertz and D. de Werra, *Using tabu search techniques for graph coloring*, Computing **39** (1988), 345–351.

[46] K. Hoffman and M. Padberg, *Solving airline crew-scheduling problems by branch-and-cut*, Management Science **39** (1993), 667–682.

[47] J. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[48] T. Jensen and B. Toft, *Graph coloring problems*, John Wiley & Sons, Inc, 1995.

[49] D. Johnson, C. Aragon, L. Mcgeoch, and C. Schevon, *Optimization by simulated annealing: An experimental evaluation - part ii (graph coloring and number partitioning)*, Operations Research **31** (1991), 378–406.

[50] N. Karmarkar, *A new polynonial-time algorithm for linear programming*, Combinatorica 4 (1984), 373–395.

[51] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (J. W. Thatcher and R. E. Miller, eds.), Plenum Press, 1972, pp. 85–103.

[52] A. Kempe, *On the geographical problem of the four colours*, American Journal of Mathematics 2 (1879), 193–200.

[53] S. Kirkpatrick, C. Gelatt, and M. Vecchi, *Optimization by simulated annealing*, Science **220** (1983), 671–680.

[54] A. Koster, *Frequency assignment*, Ph.D. thesis, Universiteit Maastricht, 1999.

[55] M. Laguna and R. Marti, *A grasp for coloring sparse graphs*, Computational Optimization and Applications 19 (2001), 165–178.

[56] A. Land and A. Doig, *An automatic method for solving discrete programming problems*, Econometrica **28** (1960), 497–520.

[57] F.T. Leighton, *A graph coloring algorithm for large scheduling problems*, Journal of Research of the National Bureau of Standards **84** (1979), 489–506.

[58] J. Linderoth and M. Savelsbergh, *A computational study of search strategies for mixed integer programming*, Tech. report, School of Industrial and Systems Engineering - Georgia Institute of Technology, 1997.

[59] M. Lübbecke, *Algorithmen zur enumeration aller ecken und facetten konvexer polyeder*, Master's thesis, Technical University of Braunschweig, Departement of Mathematical Optimization, Braunschweig, 1996.

[60] D. Matula, G. Marble, and J. Isaacson, *Graph coloring algorithms*, Graph Theory and Computing (1972), 109–122.

[61] A. Mehrotra and M. Trick, *A column generation approach for graph coloring*, INFORMS Journal on Computing **8** (1996), 344–353.

[62] I. Méndez-Díaz and P. Zabala, *A generalization of the graph coloring problem*, Revista Latinoamericana de Investigación Operativa 8 (1999), 167–184.

[63] ———, *A polyhedral approach for graph coloring*, Electronic Notes of Discrete Mathematics (2001).

[64] ———, *A branch-and-cut algorithm for graph coloring*, COLOR02-Computational Symposium of Graph Coloring and its Generalizations, 2002.

[65] B. Metzger, *Frequency assignment: Theory and applications*, Presentation at 3 National ORSA meeting, 1970.

[66] G.Ñemhauser and L. Wolsey, *Integer programming and combinatorial optimization*, John Wiley and Sons, 1988.

[67] M. Padberg, *On the facial structure of a set packing polyhedra*, Mathematical Programming 5 (1973), 199–216.

[68] M. Padberg and G. Rinaldi, *A branch-and-cut algorithm for the resolution of a large scale symmetric traveling salesman problems*, SIAM review **33** (1991), 60–100.

[69] C. Papadimitriou and K. Steiglitz, *Combinatorial optimization: Algorithms and complexity*, Prentice Hall, 1985.

[70] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, *The four color therorem*, J. of Comb. Theory Serie B **70** (1977), 2–44.

[71] F. Rossi and S. Smriglio, *A branch-and-cut algorithm for the cardinality stable set problem*, Operations Research Letters **28** (2001), 63–74.

[72] T. Sager and S. Lin, *A pruning procedure for exact graph coloring*, ORSA Journal on Computing **3** (1991), 226–230.

[73] E. Sewell, *An improved algortihm for exact graph coloring*, Cliques, Coloring and Satisfiability (D. Johnson ans M. Trick, ed.), vol. 26, DIMACS-Series in Discrete Mathematics and Theoretical Computer Science, 1993, pp. 359–373.

[74] H. Sherali and W. Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM Journal of Discrte Mathematics **3** (1990), 411–430.

[75] S. Thienel, *Abacus a branch-and-cut system*, Ph.D. thesis, Universiteit zu Koln, 1995.

[76] C. Wang, *An algorithm for the chromatic number of a graph*, Journal of A.C.M **21** (1974), 385.

[77] D. Welsh and B. Powell, *An upper bound for the chromatic number of a graph and its application to timetabling problems*, Comput. J. **10** (1979), 85–86.

[78] A. Zykov, *On some properties of linear complexes*, American Mathematics Society. **79** (1952), 81.