Tesis de Licenciatura

"Un algoritmo *Branch & Cut* para un problema de asignación de frecuencias en redes de telefonía celular"

Alumno: Diego Delle Donne

Director: Dr. Javier Marenco

Febrero de 2009

UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Resumen

En este trabajo se estudia una manera de manejar la interferencia en modelos de optimización combinatoria que representan redes de comunicación inalámbrica. En una red típica se tiene interferencia por *co-canalidad* cuando dos antenas que solapan sus áreas de cobertura utilizan la misma frecuencia para establecer las comunicaciones. Por otra parte, se genera una interferencia de menor magnitud cuando estas antenas utilizan canales de frecuencias adyacentes.
Esto motiva la formulación del *minimum-adjacency vertex coloring problem* que, dado un grafo de interferencia G representando la potencial interferencia entre las antenas y un conjunto de colores/canales, consiste en hallar un coloreo de G minimizando la cantidad de aristas cuyos vértices reciben colores adyacentes.

Se presentan en este trabajo tres modelos de programación lineal entera y se reportan resultados computacionales para estimar la contribución práctica de cada uno de ellos. Se elije luego la mejor formulación y se realiza un estudio poliedral del polítopo asociado. Se presentan cuatro desigualdades válidas, las cuales definen facetas del mismo. Finalmente, se describe la implementación de un algoritmo *Branch & Cut* y se presentan los resultados computacionales obtenidos.

Abstract

In this work we study a particular way of dealing with interference in combinatorial optimization models representing wireless communication networks. In a typical wireless network, co-channel interference occurs whenever two overlapping antennas use the same frequency channel, and a less critical interference is generated whenever two overlapping antennas use adjacent channels.
This motivates the formulation of the *minimum-adjacency vertex coloring problem* which, given an interference graph G representing the potential interference between the antennas and a set of prespecified colors/channels, asks for a vertex coloring of G minimizing the number of edges receiving adjacent colors.

In this work, three integer programming models are presented for this problem and computational results are provided in order to assess the practical contribution of each one. The best formulation is chosen and a polyhedral study is performed on the polytope asociated. Four facet-defining inequalities are presented for this formulation. Finally, an implementation of a *Branch & Cut* algorithm is described and its computational results are presented.

Agradecimientos

Principalmente a mi director **Javier Marenco** por todo el tiempo y esfuerzo que destinó para ayudarme a realizar este trabajo. Por sus colaboraciones en cada uno de los aspectos del mismo y su constante y excelente predisposición para cualquier consulta que me pudiera surgir. Sin su ayuda, este trabajo no sería ni la mitad de lo que és.

A mi vieja, **Clementina Morbo**, por el constante apoyo de todos los días. Por darme siempre más de lo que cualquiera puede llegar a necesitar y permitirme hacer mis cosas sin tener que preocuparme por la cotidianeidad de la vida. Porque sé que siempre está, pase lo que pase, incluso más de lo que debería.

A mi viejo, **José Alberto Delle Donne**, por bancarme siempre. Por estar aunque a veces se nos complique estar.

A mi groso hermano, **Juan Delle Donne**, que siempre se preocupa por que su hermanito menor tenga todo lo que tenga que tener.

A los jurados, **Gabriela Argiroffo** y **Guillermo Durán**, por tomarse el tiempo para leer la tesis en parte de sus vacaciones y por sus comentarios y sugerencias, los cuales sirvieron para enriquecer la misma.

A mis amigos del alma[1], **Charo Bouilly**, **Mau Polanco**, **Nico Novali**, **Pedro Gilmore** y **Sole Pintos** por ser lo que son. Porque en muchas ocasiones, cada uno de ellos me ayudó más de lo que cree.

Al **Arte**, sea cual fuera la forma que tome: a la música, a la matemática, al cine, a la naturaleza... a la poesía del mundo.

[1] En orden alfabético para que no se pongan celosos :p

Índice general

CAPÍTULO 1

Introducción

Las comunicaciones hoy en día representan un aspecto fundamental de la sociedad como tal. La importancia de la transmisión y recepción de información hacen imprescindible el hecho de establecer medios de comunicación eficaces. Las redes inalámbricas, en particular, resultan ser una de las tecnologías más prometedoras y abarcativas para esta tarea y, por este motivo, han sido objeto de estudio intensivo en los años recientes. Ejemplos de esto son los sistemas de telefonía celular basada en satélites, los sistemas de radio de tipo *point-to-multipoint* y los sistemas de telefonía celular móvil terrestre (ver [1, 11]).

En este capítulo se presentará con detalle una de las tecnologías más utilizadas en el ámbito de los sistemas de redes inalámbricas para telefonía celular, a saber las redes GSM, y se planteará un problema asociado a las mismas.
Luego de comentar el estado del arte de este problema, se lo modelará matemáticamente formulando el *minimum adjacency vertex coloring problem*. Finalmente se hará un breve resumen de los contenidos de cada capítulo de este trabajo.

1.1. Redes GSM de telefonía celular

El sistema GSM (de *General System for Mobile Communications*) provee un servicio de comunicaciones inalámbricas para envío y recepción de voz y de datos. En cuanto a los servicios de envío y recepción de datos, éstos son usualmente utilizados para brindar servicio de acceso a Internet y servicios de mensajería corta (SMS).

Redes y antenas

Para que las comunicaciones puedan ser establecidas en un área determinada, es necesario cubrir la misma con una red de antenas fijas, también llamadas TRXs. Cada una de estas antenas cubre una determinada porción del área. En

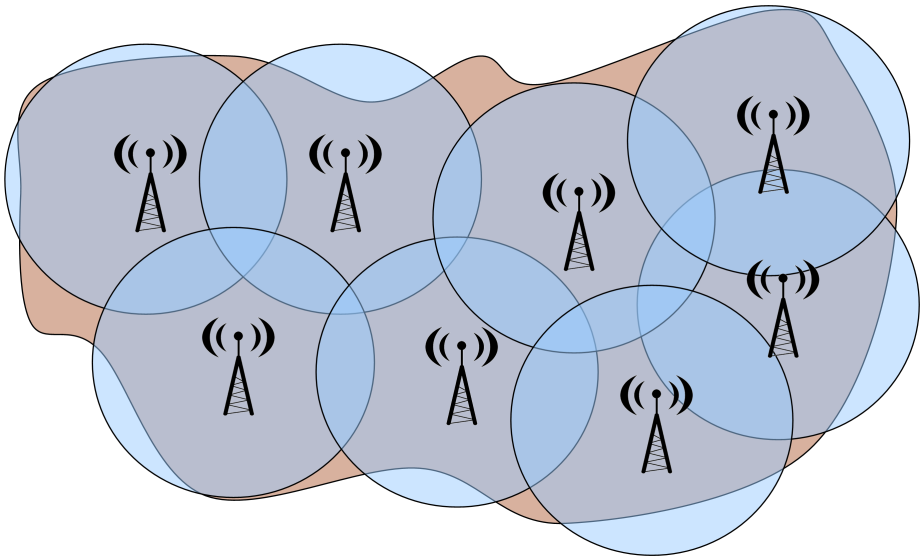


Figura 1.1: Red de antenas cubriendo un área

la Figura 1.1 se muestra un ejemplo de este tipo de redes.

Estas redes utilizan frecuencias de radio para establecer las comunicaciones. Las frecuencias utilizadas corresponden a una determinada porción del espectro electromagnético, la cual está dividida en *canales* discretos. A cada antena se le asigna uno o más canales para establecer las comunicaciones necesarias, cada una de ellas puede establecer un número fijo de comunicaciones en simultáneo utilizando técnicas de multiplexación en el tiempo. Este límite suele ser de 7 u 8 comunicaciones simultáneas en general y por este motivo en áreas de mucho tráfico de red, suelen desplegarse más de una antena para cubrir las mismas.

Comunicaciones

Cuando una persona quiere establecer una comunicación utilizando un teléfono celular, el mismo debe detectar primero las antenas que están cubriendo el lugar en la que se encuentra. Una vez detectada la antena más cercana, el teléfono utilizará el canal asignado a la misma para establecer una comunicación con ella. De ahí en adelante, mientras el móvil se mantenga dentro del área de cobertura de esta antena, la comunicación seguirá establecida utilizando el canal asignado a la misma.

El teléfono puede encontrarse en movimiento, con lo cual es probable que en algún momento llegue a una intersección entre dos áreas de cobertura de distintas antenas. En este momento el aparato detectará que la señal utilizada comienza a perder poder y que a su vez una nueva señal está ganándolo. Se produce aquí un proceso llamado *hand-over* que consiste básicamente en el pasaje

de la comunicación de una antena a la otra. Este proceso debe hacerse sin que el usuario lo note.

Interferencia en las comunicaciones

Como se mencionó previamente, existen regiones del área que se encuentran cubiertas por más de una antena, en las cuales se efectúa el proceso de *hand-over*. Sin embargo, este proceso, así como las comunicaciones en estas áreas, funciona correctamente sólo cuando las antenas en cuestión utilizan distintos canales para sus comunicaciones, pues en los casos en que éstas comparten el canal asignado, se produce una interferencia llamada *interferencia por co-canalización*, la cual impide el establecimiento de comunicaciones dentro del área.
Otro tipo de interferencia surge cuando las antenas en cuestión utilizan canales adyacentes, conocida como *interferencia por canales adyacentes*. Este tipo de interferencia no prohibe por completo las comunicaciones en el área, aunque genera problemas menores en las mismas.

***

Por estos motivos, puede verse que el problema de asignar canales a las antenas de una red, evitando este tipo de inconvenientes, puede resultar ser un problema muy dificil. En las siguientes secciones de este capítulo mencionaremos los trabajos hechos sobre estos temas y luego presentaremos el problema particular analizado en este trabajo.

1.2. Estado del arte

El problema de asignación de frecuencias en redes de telefonía celular es muy importante en la práctica, ya que todos los operadores de telefonía celular con tecnología GSM deben enfrentarse con él.
Este problema tiene muchas características que lo transforman en un problema dificil en la práctica. Además de su complejidad computacional teórica, diversos aspectos prácticos dificultan el mismo, por ejemplo:

- El tráfico en una red es un dato muy dificil de estimar *a priori*.
- Las interferencias dadas entre cada par de antenas dependen de diversos factores como la distancia entre ellas, la presencia de obstáculos (edificios, elevaciones del terreno, etc), la ubicación de otras antenas, etc., y por ello son difíciles de medir.
- En redes grandes, se dificulta estimar *a priori* el impacto que generan los cambios en las mismas.

Muchas variantes y simplificaciones de los problemas de asignación de frecuencias han sido muy estudiados utilizando heurísticas y algoritmos aproximados. Ejemplos de ello se pueden ver en [2, 3, 5, 12, 13, 14, 19].

Existen también aplicaciones de software basadas en técnicas de optimización combinatoria, las cuales se utilizan para obtener planes de asignación de frecuencias. Ejemplos de ellas son:

- Mentum Planet© (Metnum – http://www.mentum.com),
- CelPlan© (CelPlan Technologies, Inc. – http://www.celplan.com), y
- Motorola NetPlan© (Motorola - http://www.motorola.com).

Estos paquetes utilizan técnicas heurísticas y metaheurísticas (algoritmos genéticos y algoritmos evolutivos, principalmente), combinados con heurísticas *ad-hoc* para casos particulares. Actualmente, todos los operadores de telefonía celular cuentan con este tipo de paquetes de software.
Sin embargo, hay muy pocos estudios sobre métodos exactos de resolución para este tipo de problemas. La aplicación de programación lineal entera para la planificación de frecuencias en redes GSM fue propuesta por primera vez en [3] y las ideas derivadas de ese trabajo inicial se continuaron parcialmente en [6] y [7].

Los problemas de asignación de frecuencias en redes de telefonía celular están muy relacionados, como veremos más adelante, con los modelos clásicos de coloreo. Estas relaciones se estudian con detalle en [8].

1.3. El problema a estudiar

Una característica importante del problema práctico es la existencia de los dos tipos de interferencia mencionados en la Sección 1.1. Estas interferencias se pueden manejar de distintas formas de acuerdo con la definición del problema e incluso de las preferencias del operador de la red de telefonía.
Una posible situación en la planificación, si se sabe que no hay una gran cantidad de tráfico en la red[1] y hay una cantidad suficiente de canales disponibles, corresponde a prohibir ambos tipos de interferencia.
Por otro lado, una práctica habitual es penalizar fuertemente la co-canalización y admitir, con una penalización más débil, las interferencias por canales adyacentes.

Motivado por esta práctica habitual, en esta tesis se estudia un problema de optimización combinatoria que captura este tratamiento de la interferencia, prohibiendo las interferencias por co-canalidades y minimizando las interferencias por canales adyacentes sobre la estructura combinatoria resultante.

[1] Es decir, no será necesario cubrir una misma zona con más de una antena.

La idea de este problema es capturar las características de la planificación relacionadas con este doble nivel de interferencia, estudiando sus implicaciones sobre la estructura combinatoria del problema resultante.

Grafo de interferencia

Para un modelado correcto de este problema definimos el *grafo de interferencia* $G = (V, E)$ asociado a una instancia. En el mismo, para cada antena de la red existe un vértice en V, y para cada par de antenas cuyas areas de cobertura se solapen, se unen los vértices correspondientes con una arista de E.
Se define además un conjunto de colores consecutivos $C = \{c_1, \ldots, c_t\}$, con el cual se representan los canales disponibles para la asignación.
Con estas definiciones, puede verse claramente que un coloreo de G que utilice colores de C representa una asignación de frecuencias sin co-canalidades para la instancia.

Como se comentó en la Sección 1.2, las interferencias dadas entre cada par de antenas dependen de diversos factores y por lo tanto se considera a las mismas como un dato de entrada para el problema. Para cada $vw \in E$, definimos $\psi(vw) \in [0, 1]$ como el nivel de interferencia generado cuando v y w reciben colores/canales adyacentes. De esta forma, podemos definir el *minimum-adjacency vertex coloring problem* como el problema de hallar un coloreo válido de G utilizando colores de C minimizando la suma de las interferencias generadas por canales adyacentes. Formalmente,

$$\text{mín} \left\{ \sum_{\substack{vw \in E \\ |y(v)-y(w)|=1}} \psi(vw) \quad : \quad y \in \mathcal{C}(G, C) \right\}$$

donde $\mathcal{C}(G, C)$ representa el conjunto de todos los coloreos válidos de G que usan colores de C.

Es sencillo ver que este problema resuelve el problema clásico de coloreo y de esta manera se demuestra que el *minimum-adjacency vertex coloring problem* es NP-hard ([9]). Esto justifica la elección de un método de programación lineal entera.
Si bien diversos problemas de asignación de frecuencias han sido estudiados en la literatura, no estamos al tanto de estudios sobre este problema particular, es decir, sobre el *minimum-adjacency vertex coloring problem*. Más aun, no tenemos conocimiento de estudios poliedrales sobre problemas derivados de la asignación de frecuencias en redes de telefonía celular con tecnología GSM.

1.4. Contenido de la tesis

El objetivo de esta tesis es realizar un estudio poliedral del *minimum-adjacency vertex coloring problem*, e implementar un algoritmo *Branch & Cut* basado en estos resultados.

En el Capítulo 2 se presentan tres modelos de programación lineal entera para el problema propuesto y se realiza una experimentación con un algoritmo *Branch & Bound* sobre un conjunto de instancias. Se analizan los resultados obtenidos con el objetivo de elegir uno de estos modelos como punto de partida para realizar un estudio teórico sobre el poliedro asociado.

El Capítulo 3 presenta el estudio poliedral realizado al polítopo asociado al modelo elegido en el capítulo anterior. Se presentan aquí cuatro familias de desigualdades válidas para el poliedro y se demuestra que, bajo ciertas condiciones, éstas definen facetas del mismo.

En el Capítulo 4 se detalla la implementación de un algoritmo *Branch & Cut* que utiliza como planos de corte las familias propuestas en el capítulo anterior. Se presentan diversos procedimientos de separación basados en técnicas de *backtracking* y en heurísticas golosas. Se muestra también la implementación de técnicas adicionales, incluyendo técnicas de *redondeo de soluciones*, *selección de variable de branching* y *fijación de variables por implicaciones lógicas*.

En el Capítulo 5 se comenta en detalle la experimentación realizada con el algoritmo *Branch & Cut* implementado. La misma está dividida en cuatro etapas progresivas con el objetivo de ir refinando la configuración para el *Branch & Cut*. Se prueban distintas familias de desigualdades y distintos valores de parámetros, tanto para los procedimientos de separación como para parámetros propios del *Branch & Cut*. Finalmente se exponen una conclusiones y se comparan los tiempos de ejecución con los resultados obtenidos por CPLEX.

Por último, en el Capítulo 6 se reunen las conclusiones obtenidas a lo largo de esta tesis y se describen algunas líneas de trabajo futuro.

Se agrega en el Apéndice A una recopilación de conceptos básicos sobre temas de optimización combinatoria y teoría poliedral, con el objetivo de servir de referencia al lector para poder comprender correctamente todos los conceptos del presente trabajo.

CAPÍTULO 2

Modelado con programación lineal entera

En este capítulo se plantean tres formulaciones de programación lineal entera para modelar el *minimum-adjacency vertex coloring problem*; dos de ellas son adaptaciones de formulaciones preexistentes para el problema clásico de coloreo de vértices y la tercera es, a nuestro entender, un nuevo enfoque para el modelado de problemas de coloreo de grafos con técnicas de programación lineal entera. Se comenta también una cuarta formulación para problemas de coloreo tenida en cuenta, que no puede ser adaptada a este problema en forma directa.

Finalmente se presentan resultados computacionales con los tres modelos iniciales y sobre la base de esta nueva información se elije el modelo a utilizar como punto de partida para el estudio poliedral del Capítulo 3.

2.1. Stable model

Este modelo es una adaptación directa del modelo presentado en [15] para el problema clásico de coloreo de grafos. Para cada vértice $v \in V$ y cada color $c \in C$ utilizamos la *variable de asignación* binaria x_{vc}, de modo tal que $x_{vc} = 1$ si el vértice v recibe el color c y $x_{vc} = 0$ en caso contrario. También utilizamos, para cada $vw \in E$, con $v < w$, la *variable de adyacencia* binaria z_{vw}, que recibirá el valor 1 siempre que los vértices v y w reciban colores adyacentes. De esta manera, el *stable model* para el *minimum-adyacency vertex coloring problem* queda dado por:

$$\text{mín} \sum_{vw \in E} \psi(vw) \times z_{vw}$$

$$\sum_{c \in C} x_{vc} = 1 \qquad \forall v \in V \tag{2.1}$$

$$x_{vc} + x_{wc} \leq 1 \qquad \forall vw \in E, v < w, \quad \forall c \in C \tag{2.2}$$

$$x_{vc_1} + x_{wc_2} \leq 1 + z_{vw} \qquad \forall vw \in E, v < w, \quad \forall c_1, c_2 \in C, |c_1 - c_2| = 1 \tag{2.3}$$

$$x_{vc} \in \{0, 1\} \qquad \forall v \in V, \quad \forall c \in C \tag{2.4}$$

$$z_{vw} \in \{0, 1\} \qquad \forall vw \in E, v < w. \tag{2.5}$$

Las restricciones (2.1) aseguran que cada vértice reciba exactamente un color de C y las restricciones (2.2) impiden que vértices adyacentes reciban un mismo color, obteniendo así un coloreo válido para el grafo. Por otro lado, las restricciones (2.3) indican que la única manera para que dos vértices adyacentes reciban colores adyacentes es que la variable de adyacencia correspondiente tome el valor 1, pues de lo contrario la suma de las variables de asignación no podrá superar 1, es decir, ambas variables no podrán estar prendidas al mismo tiempo. Cabe notar el hecho de que estas restricciones no aseguran que la variable de adyacencia z_{vw} tome el valor 0 cuando los vértices v y w reciban colores no adyacentes, sin embargo dado que la función objetivo consiste en minimizar dichas variables, las mismas tomarán ese valor en cualquier óptimo siempre que esto sea posible[1].
El tamaño del *stable model* es de $nt + m$ variables y $n + m(2t - 1)$ restricciones.

2.2. Orientation model

Este modelo es una adaptación directa del *orientation model* para el problema clásico de coloreo de grafos [3]. Para cada vértice $v \in V$, definimos la *variable de color* entera $x_v \in \{1, \ldots, t\}$, que contiene el color asignado a v. Definimos también, para cada arista $vw \in E$, con $v < w$, las *variables de orientación* binarias o_{vw} y o_{wv} de manera tal que $o_{vw} = 1$ si y sólo si $x_v < x_w$. Por último, incluimos también las variables de adyacencia definidas para el modelo anterior. Bajo estas definiciones, el *orientation model* para el problema de coloreo de grafos queda dado por:

$$\text{mín} \sum_{vw \in E} \psi(vw) \times z_{vw}$$

$$o_{vw} + o_{wv} = 1 \qquad \forall vw \in E, v < w \tag{2.6}$$

$$x_v - x_w \geq 2 - z_{vw} - (|C| + 1) \times o_{vw} \qquad \forall vw \in E, v < w \tag{2.7}$$

$$-x_v + x_w \geq 2 - z_{vw} - (|C| + 1) \times o_{wv} \qquad \forall vw \in E, v < w \tag{2.8}$$

$$1 \leq x_v \leq |C| \qquad \forall v \in V \tag{2.9}$$

$$x_v \in \mathbb{Z} \qquad \forall v \in V \tag{2.10}$$

$$o_{vw}, o_{wv}, z_{vw} \in \{0, 1\} \qquad \forall vw \in E, v < w. \tag{2.11}$$

[1] A lo largo de este trabajo (en particular en el Capítulo 3) se utilizará muchas veces esta característica de las variables de adyacencia.

Las restricciones (2.6) seleccionan una orientación para cada arista del grafo, y con las restricciones (2.7) y (2.8) se asegura que los colores asignados respeten este orden parcial, definiendo a su vez las variables de adyacencia con valores coherentes. Por último, las restricciones (2.9) fuerzan a los colores elegidos a estar dentro del rango de colores disponibles.

El tamaño del *orientation model* es de $n + 3m$ variables y $n + 3m$ restricciones aunque es importante aclarar que en la implementación de este modelo, es posible reducir a la mitad la cantidad de variables de orientación, ya que para cada arista, las variables correspondientes resultan ser complementarias. De esta forma, también se eliminan las restricciones (2.6) dejando el tamaño del modelo en $n + 2m$ variables y $n + 2m$ restricciones.

2.3. Distance model

El *distance model* es, a nuestro entender, un nuevo enfoque para el modelado de problemas de coloreo de grafos con técnicas de programación lineal entera, y puede ser adaptado fácilmente al problema clásico de coloreo. Para cada par de vértices $v, w \in V, v < w$, incorporamos la *variable de distancia* entera $x_{vw} \in [-(t-1), t-1]$, que registra la distancia entre los colores asignados a los vértices v y w. Por ejemplo, si el vértice v recibe el color 5 y el vértice w recibe el color 2, entonces $x_{vw} = 3$. También utilizamos en este modelo las variables de orientación y de adyacencia ya presentadas en los modelos anteriores. Con estas definiciones, el *distance model* queda dado por:

$$\text{mín} \sum_{vw \in E} \psi(vw) \times z_{vw}$$

$$o_{vw} + o_{wv} = 1 \qquad \forall vw \in E, v < w \tag{2.12}$$

$$x_{vw} = x_{vk} + x_{kw} \qquad \forall v, k, w \in V, v < k < w \tag{2.13}$$

$$x_{vw} \geq 2 - z_{vw} - (|C| + 1) \times o_{vw} \qquad \forall vw \in E, v < w \tag{2.14}$$

$$-x_{vw} \geq 2 - z_{vw} - (|C| + 1) \times o_{wv} \qquad \forall vw \in E, v < w \tag{2.15}$$

$$-(|C| - 1) \leq x_{vw} \leq |C| - 1 \qquad \forall v, w \in V, v < w \tag{2.16}$$

$$x_{vw} \in \mathbb{Z} \qquad \forall v, w \in V, v < w \tag{2.17}$$

$$o_{vw}, o_{wv}, z_{vw} \in \{0, 1\} \qquad \forall vw \in E, v < w \tag{2.18}$$

Las restricciones (2.13) aseguran la consistencia entre las distancias, mientras que las restricciones (2.14) y (2.15) validan el coloreo y definen los valores de las variables de adyacencia. Finalmente, las restricciones (2.16) limitan los valores de las distancias para garantizar que ninguna sea mayor que $(|C| - 1)$.

Es importante remarcar que las restricciones (2.13) no son linealmente independientes ya que la cantidad de restricciones es de orden cúbico en el tamaño

de V. Más aun, cualquier implementación con este tamaño de modelo sería probablemente una mala elección. El siguiente teorema caracteriza un conjunto de restricciones equivalente de orden cuadrático.

Teorema 2.3.1. *El conjunto de restricciones (2.13) es equivalente a*

$$x_{v,v+1} + x_{v+1,v+2} = x_{v,v+2} \qquad \forall v \in V, \quad 1 \leq v \leq n-2 \tag{2.19}$$

$$x_{v,w} + x_{v+1,w-1} = x_{v,w-1} + x_{v+1,w} \quad \forall v, w \in V, \quad 1 \leq v \leq n-3,\ v+3 \leq w \leq n. \tag{2.20}$$

Demostración. Demostraremos la equivalencia en dos pasos:

$\Rightarrow$) Sea $x \in \mathbb{R}^{\frac{n(n-1)}{2}}$ tal que x cumple (2.13). Dado que (2.19) es un caso particular de (2.13), entonces x la cumple trivialmente. Veamos ahora que x también cumple (2.20).
Sean $v, w \in V$ tales que $1 \leq v \leq n-3$ y $v+3 \leq w \leq n$. Entonces

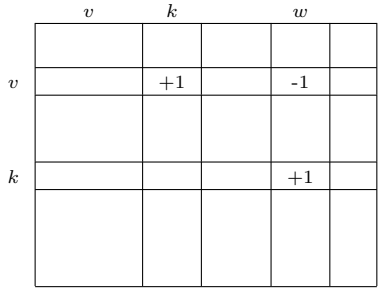
$$x_{vw} = x_{vw-1} + x_{w-1,w}$$
$$x_{v+1,w-1} + x_{w-1,w} = x_{v+1,w}$$

usando (2.13) con $(v, w-1, w)$ y $(v+1, w-1, w)$ respectivamente. Sumando estas ecuaciones se obtiene

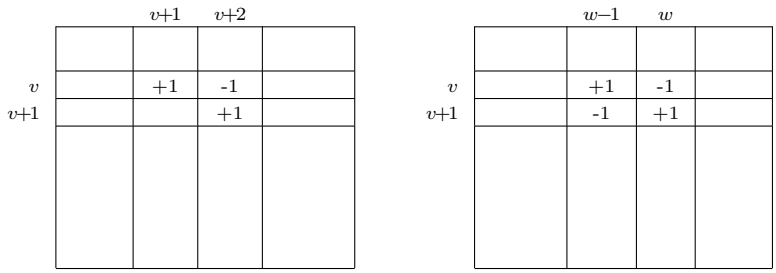
$$x_{vw} + x_{v+1,w-1} = x_{vw-1} + x_{v+1,w}.$$

Como v y w son arbitrarios, se tiene la validez de (2.20).

$\Leftarrow$) Sea $x \in \mathbb{R}^{\frac{n(n-1)}{2}}$ tal que x cumple (2.19) y (2.20). Veamos que también cumple (2.13).
Llamamos $R_1(v)$ y $R_2(v, w)$ a las ecuaciones (2.19) y (2.20) respectivamente y representamos cada ecuación por su matriz de coeficientes. Por ejemplo, las matrices de coeficientes de (2.13), $R_1(v)$ y $R_2(v, w)$ son



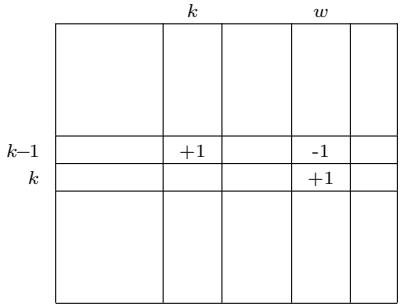
$$x_{vk} + x_{kw} = x_{vw}$$



$R_1(v)$ $R_2(v, w)$

respectivamente.

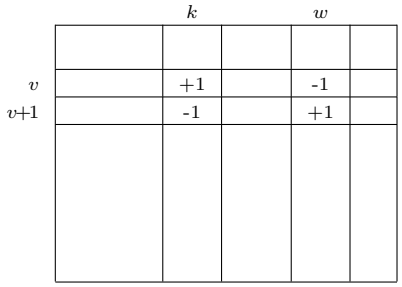
Sean entonces $v, k, w \in V$ tales que $v < k < w$. Si $k = v + 1$ y $w = k + 1$, entonces es sencillo ver que (2.19) es (2.13). Supongamos entonces que esto no sucede.
Veamos primero el caso en que $k = v + 1$ pero $w > k + 1$. En este caso, partiendo de la restricción $R_1(k-1)$, sumando las restricciones $R_2(k-1, k+2)$, $R_2(k-1, k+3)$, ..., $R_2(k-1, w)$, y cancelando las variables que aparecen a ambos lados de la ecuación resultante se obtiene la siguiente ecuación:



(a)

y dado que $v = k - 1$, entonces (a) es exactamente (2.13).

Finalmente resta ver el caso en que $k > v+1$. Aquí, partiendo de la restricción $R_2(v, k+1)$, sumándole $R_2(v, k+2)$, $R_2(v, k+3)$, ..., $R_2(v, w)$ y cancelando las variables que aparecen a ambos lados de la ecuación resultante es sencillo ver que se obtiene la siguiente matriz de coeficientes:



y si a la ecuación obtenida le sumamos ahora $R_2(v+1, k+1)$, $R_2(v+1, k+2)$, ..., $R_2(v+1, w)$, obtenemos

	k		w	
v	+1		-1	
v+2	-1		+1	

Repitiendo el procedimiento con las filas $v+2$, $v+3$, ..., $k-2$ se obtiene

	k		w	
v	+1		-1	
k–1	-1		+1	

(b)

Y por último, sumando (a) y (b) obtenemos finalmente la ecuación que buscamos[2]:

	k		w	
v	+1		-1	
k			+1	

$(a) + (b) \equiv (2.13)$

Por lo tanto, la restricción (2.13) puede escribirse como combinación de las restricciones (2.19) y (2.20). □

Es sencillo ver que la cantidad de ecuaciones de (2.19) es $n-2$, y en el caso de (2.20) la cuenta es un poco menos directa pero se puede ver que para cada

[2]Vale aclarar que la existencia de (a) no depende de la hipótesis que indica que $k = v+1$. Dicha hipótesis se utiliza sólo para ver que en ese caso, (a) es equivalente a las ecuaciones del modelo.

$v = 1, \ldots, n-3$, existen $n-2-v$ vértices w que cumplen con $v+3 \leq w \leq n$. Por lo tanto la cantidad de ecuaciones de (2.20) es

$$\begin{aligned}\sum_{v=1}^{n-3}(n-2-v) &= \sum_{v=1}^{n-3}(n-2) - \sum_{v=1}^{n-3} v \\ &= (n-3)(n-2) - \frac{(n-3)(n-2)}{2} \\ &= \frac{(n-3)(n-2)}{2}.\end{aligned}$$

De esta manera, utilizando (2.19) y (2.20) como reemplazo de (2.13), y sumando también las $3m$ restricciones provenientes de (2.12), (2.14) y (2.15), la cantidad de restricciones del *distance model* es de

$$3m + \left[(n-2) + \frac{(n-3)(n-2)}{2}\right] = 3m + \frac{(n-2)(n-1)}{2},$$

y la cantidad de variables para este modelo es $3m + \frac{n(n-1)}{2}$. Sin embargo, al igual que el *orientation model*, las variables de orientación pueden reducirse a la mitad dejando al modelo con un tamaño final de $2m + \frac{(n-2)(n-1)}{2}$ restricciones y $2m + \frac{n(n-1)}{2}$ variables.

Si bien este modelo tiene un tamaño considerablemente mayor que los otros dos, hay un aspecto interesante a destacar con respecto a la cantidad de soluciones enteras de cada uno de ellos. Puede verse que para cualquier solución entera del *orientation model* existe una única solución en el *distance model* que representa el mismo coloreo, y se obtiene de manera directa calculando las distancias entre los colores asignados a los vértices y copiando las variables de adyacencia y orientación[3]. Sin embargo, no ocurre lo mismo en el sentido inverso ya que dada una solución del *distance model* pueden existir múltiples soluciones en el *orientation model* que representan el mismo coloreo. Esto muestra que la cantidad de soluciones enteras del *distance model* resulta ser menor que la del *orientation model*.
Análogamente, puede verse fácilmente que ocurre lo mismo para el *stable model*. En este caso las variables de orientación no pueden copiarse ya que no existen en el primer modelo, pero las mismas se obtienen de manera directa una vez que se calcularon las variables de distancia para cada par de vértices.

***

Además de los tres modelos descriptos, tuvimos en cuenta el *representatives model* presentado en [4]. En este modelo se especifica un coloreo por medio de las clases de color que induce (una clase de color es el conjunto de vértices que

[3]Estas variables tienen el mismo significado en ambos modelos.

reciben un mismo color), y cada clase de color se especifica por medio de un vértice representante, que es referenciado por los restantes vértices de su clase. Si bien este modelo resulta interesante para el problema clásico de coloreo de grafos, no es posible realizar una adaptación directa al problema de coloreo de adyacencias mínimas ya que este modelo no admite una noción de distancia entre los colores asignados. Es decir, no indica qué color asignar a cada conjunto, sino que simplemente particiona el grafo en conjuntos independientes.
Una posible adaptación de este modelo para el *minimum-adyacency vertex coloring problem* sería mediante el agregado de variables para representar el color asignado a cada clase de color, pero de esta forma se perdería la estructura particular de este modelo.

2.4. Resultados computacionales

En esta sección se reportan resultados computacionales sobre instancias reales, con el objetivo de determinar los tiempos de resolución y la calidad de las soluciones obtenidas por cada uno de los modelos presentados en este capítulo.

2.4.1. Instancias de prueba

La experimentación fue realizada sobre las instancias de CELAR (Centre d´Electronique de l´ARmement France) del proyecto EUCLID CALMA (ver [11]). Dado que el principal interés recae sobre los tiempos de ejecución necesarios para lograr optimalidad, se extrajeron subgrafos conexos inducidos de las instancias con el objetivo de controlar los tamaños y densidades de las mismas para los experimentos.

El procedimiento de generación de subgrafos a partir de las instancias está basado en la idea de la extracción de "vecindades" del grafo. En cada paso del algoritmo se agrega un nuevo vértice escogido de manera aleatoria entre los vecinos de los vértices ya escogidos previamente, manteniendo un grafo conexo. Además, al agregar cada nuevo vértice, el mismo es conectado con todos los vértices ya escogidos con los cuales estaba conectado en el grafo original.
A continuación se muestra un pseudocódigo del procedimiento de extracción de subgrafos, que recibe como parámetros de entrada el grafo en cuestión y la cantidad de vértices para el grafo resultante:

Una posibilidad interesante sería ponderar la elección del nuevo vértice en cada iteración en función de la cantidad de vértices ya elegidos que sean adyacentes a él. De esta manera se esperaría que la vecindad extraída sea más compacta y menos proliferada.

Algoritmo 1 Extraer subgrafo

Entrada: $G = (V, E)$, k
1: $v \leftarrow \texttt{verticeAleatorio}(V)$
2: $H \leftarrow \{v\}$
3: $vecinos \leftarrow \texttt{vecinos}(G, v)$
4: **mientras** $|H| < k$ **hacer**
5: $\quad v \leftarrow \texttt{verticeAleatorio}(vecinos)$
6: $\quad H \leftarrow H \cup \{v\}$
7: $\quad vecinos \leftarrow vecinos \setminus \{v\} \cup \texttt{vecinos}(G, v)$
8: **fin mientras**
9: **devolver** $\texttt{subgrafoInducido}(G, H)$

2.4.2. Resultados computacionales

En la Tabla 2.1 presentamos los resultados computacionales para 165 instancias generadas, agrupadas por su cantidad de vértices y densidad. Mostramos, para cada grupo, la cantidad de instancias resueltas de manera óptima con el tiempo de ejecución promedio y el desvío estándar para cada uno de los tres modelos. Para las instancias no resueltas luego de un límite de tiempo de una hora, mostramos el *gap* de optimalidad promedio. Los experimentos fueron ejecutados usando CPLEX 10.0 con parámetros por defecto en una computadora Silicon Graphics Origin 200 con 1024 MB de memoria RAM y cuatro microprocesadores R12000, corriendo a 400 MHz cada uno.

			Resueltas			Resueltas/Total			Tiempo prom. (seg)			Devío std.			No resueltas		
															GAP prom.	
$\|V\|$	$\|C\|$	Densidad	Sta	Ori	Dis	Sta	Ori	Dis	Sta	Ori	Dis	Sta	Ori	Dis	Sta	Ori	Dis
10	7	0 % - 50 %	11/11	11/11	11/11	0,36	0,36	0,54	0,48	0,64	0,49						
		50 % - 60 %	12/12	12/12	12/12	1,08	2,16	3,5	1,18	1,51	2,66						
		60 % - 100 %	11/11	11/11	11/11	1	14	31,54	0,42	12,64	34,56						
12	8	0 % - 50 %	10/10	10/10	10/10	0,8	4,7	7,1	0,74	7,79	11,11						
		50 % - 60 %	7/7	7/7	7/7	4,28	52,85	77,28	2,05	74,96	95,97						
		60 % - 100 %	6/6	6/6	6/6	4,66	103,1	277,8	2,92	82,32	229,7						
14	9	0 % - 50 %	18/18	18/18	18/18	14,33	20,77	186,1	26,93	33,16	518,6						
		50 % - 60 %	7/7	6/7	6/7	50,85	325,5	893,6	38,62	145,7	357,7				55 %	55 %	
		60 % - 100 %	3/3	1/3	1/3	131	226	167	86,82	0	0				47 %	59 %	
16	11	0 % - 50 %	13/13	13/13	12/13	43,61	159,8	166,2	58,02	287,8	305,6						33 %
		50 % - 60 %	10/12	2/12	2/12	1152	1183	1624	477,2	742,5	895	36 %	66 %				82 %
		60 % - 100 %	3/4	0/4	0/4	1718			558,6			57 %	80 %				93 %
18	13	0 % - 50 %	20/21	18/21	17/21	141,5	214,8	257	243,3	455,3	502,9	31 %	79 %				75 %
		50 % - 60 %	1/4	1/4	1/4	2	1	50	0	0	0	41 %	100 %				100 %
		60 % - 100 %	0/5	0/5	0/5							43 %	100 %				100 %
20	13	0 % - 50 %	6/11	6/11	4/11	186,3	426,5	498,5	197,9	729,2	372,8	31 %	100 %				100 %
		50 % - 60 %	1/5	1/5	1/5	43	1	48	0	0	0	67 %	100 %				100 %
		60 % - 100 %	2/5	0/5	0/5	1902			139			54 %	100 %				100 %

Tabla 2.1: Resultados computacionales para subgrafos de las instancias de CELAR.

2.4.3. Cantidad de instancias resueltas

Un primer aspecto a evaluar es la cantidad de instancias que cada modelo resuelve en forma óptima. Puede verse que para los grupos de instancias pequeñas (10 y 12 vértices por grafo) ninguno de los tres modelos encuentra problemas para resolver las instancias generadas.
Con instancias levemente más grandes (14 vértices), los tres modelos siguen resolviendo las instancias con baja densidad, pero para densidades mayores, sólo el *stable model* logra resolver todas las instancias en forma óptima, mientras que los otros dos modelos comienzan, en este punto, a encontrar problemas para lograr un óptimo en algunas de ellas.
Al pasar al grupo de grafos de 16 vértices, si bien ninguno de los tres modelos logra resolver todas las instancias propuestas, es claro que la cantidad de instancias resueltas por el *stable model* es muy superior a la de los otros dos modelos. En los últimos dos grupos de instancias si bien se mantiene esta diferencia entre el *stable model* y los otros dos modelos, la misma no es demasiado marcada ya que incluso el primer modelo deja muchas instancias sin resolver. Tal vez la diferencia más interesante sea la que se da en las instancias de mayor densidad del último grupo donde puede verse que el *stable model* continúa resolviendo algunas instancias de manera óptima a diferencia de los otros dos modelos que dejan la serie entera sin resolver.

Una aclaración importante es el hecho de que ninguno de los tres métodos logran resolver instancias de tamaño mayor que 20 vértices, y por lo tanto éstas no se muestran en la Tabla 2.1. Si bien el *minimum-adjacency vertex coloring problem* no puede compararse en forma directa con el problema clásico de coloreo, vale la pena mencionar que, para este último, existen algoritmos en la literatura que lo resuelven de manera exacta en instancias de hasta 90 vértices, para grafos aleatorios. Un ejemplo de esto puede verse en el *Branch & Cut* presentado en [16].

2.4.4. Tiempos de resolución

La Tabla 2.1 muestra, para las intancias resueltas en forma óptima, los tiempos de resolución promedio y el desvío estándar para cada grupo y cada densidad.

En cada uno de los grupos de instancias y para cada modelo puede verse notablemente cómo el tiempo de ejecución se incrementa considerablemente al aumentar la densidad de los grafos.
También puede notarse, para cada modelo, el aumento en el tiempo de ejecución a medida que el tamaño de las instancias se hace más grande, pudiendo verse que con leves aumentos en el tamaño[4] el incremento en los tiempos de ejecución resulta ser muy grande. Este comportamiento no llama específicamente la

[4]Los grupos aumentan de a 2 vértices.

atención ya que como dijimos en la Sección 1.3, el problema es NP-Hard.

La comparación más interesante surge al comparar los tiempos de los modelos entre sí. Puede verse cómo en casi todos los grupos de instancias, el *stable model* logra tiempos de resolución significativamente mejores que los otros dos modelos. Si bien hay algunos grupos en los que esto parece no cumplirse, en un análisis un poco más detallado puede deducirse el porqué.
Uno de estos casos es el grupo de instancias de 16 vértices y densidad media; se ve en este grupo que el tiempo promedio de los tres modelos es muy similar, sin embargo mirando la cantidad de instancias resueltas por cada modelo puede notarse que el *stable model* resuelve 10 instancias mientras que los otros dos modelos logran resolver solamente 2. Profundizando el análisis sobre los tiempos individuales de este grupo (presentados en la Tabla 2.2), puede verse que sólo una de las 10 instancias resueltas por el *stable model* es resuelta también por los otros dos modelos y que los tiempos obtenidos por el *stable* y el *orientation model* para la misma son relativamente bajos comparados con los de las demás instancias. Esto beneficia claramente al promedio obtenido por el *orientation model*, pues las demás instancias (no resueltas) no se tienen en cuenta para el cálculo del mismo.

Tiempo (seg.)		
Sta	Ori	Dis
822	***	***
1692	***	***
1687	***	***
***	***	***
1690	***	***
1694	***	***
***	1926	729
927	***	***
563	***	***
1072	***	***
394	441	2519
981	***	***

***: No resuelta

Tabla 2.2: Tiempos para 16 vértices y densidad media.

Por otro lado, en la tabla puede notarse también otro caso particularmente interesante, en el cual el *stable model* no logra resolver la instancia y los otros dos modelos sí[5]. En este caso el mejor tiempo es obtenido por el *distance model*, mejorando levemente el promedio final del mismo. De todas maneras, si bien los tiempos promedio en este grupo resultan ser similares para los tres modelos, puede verse en la tabla que en la mayoría de los casos el mejor resultado es

[5]No se analizó en profundidad esta instancia, pero probablemente este hecho tenga relación con alguna característica de la estructura del grafo.

obtenido por el *stable model*.

Otros dos casos particulares se dan en los grupos de 18 y 20 vértices, de densidad media en ambos casos. Los tres modelos resolvieron una sola instancia de cada uno de estos grupos y puede verse que el *orientation model* obtiene el mejor de los tiempos en ambas. Sin embargo, los tres modelos obtienen tiempos de resolución extremadamente bajos en comparación con los tiempos para el resto de las instancias de estos grupos y por este motivo puede considerarse a estas dos instancias como casos particulares que probablemente sean instancias muy sencillas de resolver.

2.4.5. Calidad de las soluciones

Para las instancias que no pudieron ser resueltas en forma óptima la tabla muestra el promedio de los *gap* de optimalidad obtenidos.
Se puede observar claramente que ya desde el grupo de instancias de 14 vértices, tanto el *orientation model* como el *distance model* comienzan a tener problemas para resolver instancias y por lo tanto en la tabla se muestra el *gap* obtenido en estos casos, obteniendo el *orientation model* resultados levemente mejores.
En los siguientes grupos de instancias, ya los tres modelos generan *gaps* no nulos con el límite de tiempo impuesto, sin embargo en todos los casos el *stable model* parece comportarse mejor que los otros dos modelos, logrando siempre *gaps* más ajustados.

Ya en el último grupo de instancias puede verse claramente que la calidad de las soluciones obtenidas por el *stable model* con un límite de tiempo de una hora es muy superior a las obtenidas por los otros dos modelos.

2.4.6. Conclusiones

El objetivo de la experimentación con los modelos propuestos es el de contar con la mayor información posible para poder elegir uno de ellos como punto de partida para la realización de un estudio teórico del poliedro asociado al mismo. Por lo expuesto en las secciones anteriores, el modelo elegido es el *stable model*, ya que éste demostró un desempeño mucho mejor que los otros modelos en las instancias evaluadas y por ello parecería ser el mejor punto de partida para este estudio.

El estudio poliedral mencionado se verá en detalle en el capítulo siguiente, en el cual se presentan varias desigualdades válidas fuertes para la cápsula convexa del poliedro. Luego, en el Capítulo 4, se describirá una implementación de un algoritmo *Branch & Cut*, en el cual se utilizarán como planos de corte las familias de desigualdes válidas halladas. Para ello se implementaron distintos procedimientos de separación, los cuales se presentan en el mismo capítulo. Finalmente veremos, en el Capítulo 5, resultados computacionales del *Branch & Cut*.

CAPÍTULO 3

Estudio poliedral

En este capítulo se presenta un estudio poliedral de la cápsula convexa de las soluciones factibles del *stable model*. Al comienzo se analiza la misma con el objetivo de determinar su dimensión, y luego se presentan algunas desigualdades válidas encontradas y, en la mayoría de los casos, se demuestra que las mismas definen facetas del poliedro asociado.

3.1. Estudio de la cápsula convexa

Dado un grafo $G = (V, E)$ y un conjunto de colores consecutivos C, definimos $\mathcal{PS}(G, C) \subseteq \mathbb{R}^{nt+m}$ como la cápsula convexa de los vectores de incidencia de todas las soluciones factibles del *stable model* formulado para el grafo G y el conjunto de colores C.

Damos ahora una definición importante para este capítulo. Dados un poliedro $\mathcal{P} \subseteq \mathbb{R}^k$, una matriz $A \in \mathbb{R}^{h \times k}$ y un vector $b \in \mathbb{R}^h$, si $Ax = b$ es un sistema de ecuaciones linealmente independientes, entonces se dice que es un *sistema de ecuaciones minimal* para $\mathcal{P}$ si todo punto $x \in \mathcal{P}$ cumple $Ax = b$. Se demuestra en [18] que, en este caso, la dimensión del poliedro es $dim(\mathcal{P}) = k - h$.

Teorema 3.1.1. *Si $|C| > \chi(G)$ entonces las ecuaciones (2.1) definen un sistema minimal de ecuaciones para $\mathcal{PS}(G, C)$.*

Demostración. Sean $\lambda \in \mathbb{R}^{nt+m}$ y $\lambda_0 \in \mathbb{R}$ tales que $\lambda y = \lambda_0$ para toda solución $y = (x, z) \in \mathcal{PS}(G, C)$. Veremos que (λ, λ_0) es una combinación lineal de los vectores de coeficientes de las ecuaciones (2.1).

Dado que $|C| > \chi(G)$, entonces existe una solución $y = (x, z)$ en la cual al menos un color no se utiliza. Sea $v \in V$ un vértice arbitrario, y llamemos $c \in C$ al color asignado a v en y. Sea $c' \in C$ el color no utilizado en y, y sea $y' = (x', z')$

la solución que se obtiene cambiando en y el color de v a c' y dejando el resto de las asignaciones sin modificar. La solución y' es factible, ya que v no comparte el color c' con ningún otro vértice, pues c' no estaba utilizado en y.

Es fácil ver que las soluciones y e y' coinciden en los valores de todas las variables salvo en x_{vc} y $x_{vc'}$, para las cuales tenemos

$$x_{vc} = x'_{vc'} = 1$$

$$x_{vc'} = x'_{vc} = 0.$$

Sabemos además que $\lambda y = \lambda_0$ y $\lambda y' = \lambda_0$ con lo cual

$$\lambda y = \lambda y'. \tag{3.1}$$

Si cancelamos en (3.1) los términos que sabemos que son iguales, obtenemos

$$\lambda_{x_{vc}} x_{vc} + \lambda_{x_{vc'}} x_{vc'} = \lambda_{x_{vc}} x'_{vc} + \lambda_{x_{vc'}} x'_{vc'}$$

y reemplazando las variables por los valores correspondientes podemos ver que

$$\lambda_{x_{vc}} = \lambda_{x_{vc'}}. \tag{3.2}$$

Permutando los colores de y se obtienen distintas soluciones en las cuales c y c' pueden ser colores cualesquiera de C, con lo cual la ecuación (3.2) es válida para cualquier par de colores distintos de C. Además, como el vértice v es arbitrario, tenemos

$$\lambda_{x_{vc}} = \lambda_{x_{vc'}} \qquad \forall v \in V, \forall c, c' \in C$$

Veremos ahora que, para cada arista $vw \in E$, el coeficiente de λ para la variable z_{vw} (es decir, $\lambda_{z_{vw}}$) es nulo. Tomemos una solución $y = (x, z)$ en la cual los colores asignados a v y a w no son adyacentes. Esta solución puede hallarse permutando los colores asignados de ser necesario, pues $|C| \geq 3$ (ya que $|C| > \chi(G)$ y, como $E \neq \emptyset$, entonces $\chi(G) \geq 2$). Podemos asumir que en esta solución la variable z_{vw} toma el valor 0. Además, podemos asumir que existe una solución con estas características, en la cual para toda arista $v'w'$ distinta de vw, la variable $z_{v'w'}$ vale 1 (sin importar que los colores asignados a los extremos de dicha arista sean adyacentes).
A partir de la solución y, podemos construir la solución $y' = (x', z')$ cambiando el valor de la variable z_{vw} a 1, y de esta manera ambas soluciones coinciden en todos los valores salvo en esta variable.

Finalmente, podemos ver que (λ, λ_0) es una combinación lineal de los vectores de coeficientes de las ecuaciones (2.1) ya que

$$\lambda x = \sum_{v \in V} \beta_v \left[\sum_{c \in C} x_{vc} \right]$$

donde β_v es el valor que toman todos los $\lambda_{x_{vc}}$, para un determinado vértice $v \in V$ y cualquiera de los colores de C. □

Corolario 3.1.1. *Si $|C| > \chi(G)$, la dimensión de $\mathcal{PS}(G, C)$ es $n(t-1)+m$.*

El Corolario 3.1.1 caracteriza la dimensión de $\mathcal{PS}(G, C)$ cuando $|C| > \chi(G)$. La caracterización de la dimensión de este polítopo para el caso $|C| = \chi(G)$ es un problema abierto para esta y otras formulaciones del problema de coloreo, dado que este parámetro está fuertemente influenciado por la estructura de G.

3.2. Desigualdades válidas

En esta sección presentamos algunas desigualdades válidas encontradas para $\mathcal{PS}(G, C)$. En la mayoría de los casos, las desigualdades propuestas definen, bajo determinadas circunstancias, facetas de este polítopo.

3.2.1. Consecutive colors clique (CCK) inequality

Esta desigualdad válida impone una cota inferior para las variables de adyacencia correspondientes a las aristas de una clique de G cuyos extremos reciban colores dentro de un conjunto de colores consecutivos. Antes de dar la definición formal de la misma, se comenta a continuación el origen de la desigualdad.

Consideremos una clique dada por un conjunto de vértices $K \subseteq V$, y consideremos también un conjunto de colores consecutivos $Q = \{c_1, \ldots, c_q\} \subseteq C$. En el caso en que se utilicen todos los colores de Q para pintar vértices de K, se generarían $q-1$ adyacencias de color entre los vértices de la clique, pues éstos son todos adyacentes entre sí. En este caso se podría afirmar que

$$\sum_{v,w \in K} z_{vw} \geq q - 1.$$

Sin embargo, por cada color de Q no usado, la cantidad de adyacencias generadas por estos colores disminuye, aunque cuánto disminuye depende de cuál sea el color que no se utilice. Si el color no utilizado es el primero o el último (es decir, c_1 o c_q, resp.), entonces la cantidad de adyacencias debería disminuir en 1. Podemos entonces cambiar la desigualdad para reflejar esta posibilidad de la siguiente manera:

$$\sum_{v,w \in K} z_{vw} \geq (q-1) - \left(1 - \sum_{v \in K} x_{vc_1}\right) - \left(1 - \sum_{v \in K} x_{vc_q}\right),$$

ya que $\sum_{v \in K} x_{vc_1}$ y $\sum_{v \in K} x_{vc_q}$ valen 1 si c_1 y c_q están asignados a vértices de K, y valen 0 si no, respectivamente. En el caso en que el color no utilizado esté entre c_2 y c_{q-1} inclusive, entonces la cantidad de adyacencias disminuiría en 2, y

análogamente podemos representar esto en la desigualdad como

$$\sum_{v,w\in K} z_{vw} \geq (q-1) - \left(1 - \sum_{v\in K} x_{vc_1}\right) - \left(1 - \sum_{v\in K} x_{vc_q}\right) - \sum_{c\in Q\setminus\{c_1,c_q\}} 2\left(1 - \sum_{v\in K} x_{vc}\right). \tag{3.3}$$

La desigualdad definida a continuación es una reescritura de esta última.

Definición 3.2.1. *Sea $K \subseteq V$ un conjunto de vértices que inducen una clique de G, y sea $Q = \{c_1, \ldots, c_q\} \subseteq C$ un conjunto de colores consecutivos tal que $c_{i+1} = c_i + 1$, para $i = 1, \ldots, q-1$. Definimos la* consecutive colors clique inequality *asociada a K y Q como*

$$\sum_{v\in K} \left[x_{vc_1} + x_{vc_q} + \sum_{c\in Q\setminus\{c_1,c_q\}} 2x_{vc} \right] \leq (q-1) + \sum_{v,w\in K} z_{vw}. \tag{3.4}$$

Antes de demostrar la validez de estas desigualdades definimos la noción de *distribución de colores*. Sea $Q \subseteq C$ un conjunto de colores consecutivos y sea $W \subseteq V$ un conjunto de vértices del grafo. Dado un coloreo de G con colores de C, decimos que los conjuntos disjuntos de colores consecutivos $Q_1, Q_2, \ldots, Q_r \subseteq Q$ representan una *distribución de colores* de Q en W si $\forall c \in Q$,

$c \in Q_i$ para algún $i \in \{1, \ldots, r\} \Leftrightarrow c$ está asignado a algún vértice $v \in W$.

Decimos que la distribución de colores es *minimal* si para todo $i = 1, \ldots, r-1$, los conjuntos Q_i y Q_{i+1} son no contiguos. Podemos visualizar esto con el siguiente ejemplo:

$Q =$

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}
	x	x	x		x				x	x	x

donde una x en el color c_j indica que c_j está asignado a algún vértice de W, y así la distribución minimal de colores para este ejemplo estaría dada por $Q_1 = \{c_2, c_3, c_4\}$, $Q_2 = \{c_6\}$ y $Q_3 = \{c_{10}, c_{11}, c_{12}\}$. Es sencillo ver que la distribución minimal de colores es única en cualquier caso.

Análogamente se define la *distribución inversa de colores* de Q en W, teniendo en cuenta en este caso los colores de Q **no** utilizados por vértices de W. Es decir, $\overline{Q}_1, \ldots, \overline{Q}_{\overline{r}}$ es una *distribución inversa de colores* de Q en W si sucede lo siguiente:

$c \in \overline{Q}_i$ para algún $i \in \{1, \ldots, \overline{r}\} \Leftrightarrow c$ **no** está asignado a ningún vértice $v \in W$.

La noción de minimalidad es análoga. En el ejemplo anterior, la *distribución minimal inversa* estaría dada por $\overline{Q}_1 = \{c_1\}$, $\overline{Q}_2 = \{c_5\}$ y $\overline{Q}_3 = \{c_7, c_8, c_9\}$.

Proposición 3.2.1. *Las* consecutive colors clique inequalities *son válidas para* $\mathcal{PS}(G, C)$.

Demostración. Dada una solución $y = (x, z) \in \mathcal{PS}(G, C) \cap \mathbb{Z}^{nt+m}$, queremos probar que la desigualdad (3.4) es cumplida por y. Definimos, para simplicidad de la demostración, $\delta_j \in \{0, 1\}$ de manera tal que

$$\delta_j = \begin{cases} 1 & \text{si el color } c_j \text{ está asignado a algún vértice de } K, \\ 0 & \text{si no.} \end{cases}$$

Sea $Q_1, Q_2, \ldots, Q_r$ la distribución minimal de colores de Q en K. Es posible reescribir el lado izquierdo de la desigualdad (3.4) como

$$\sum_{v \in K} \left[x_{vc_1} + x_{vc_q} + \sum_{c \in Q \setminus \{c_1, c_q\}} 2x_{vc} \right] = \left[\sum_{i=1}^{r} 2|Q_i| \right] - \delta_1 - \delta_q \tag{3.5}$$

pues las únicas variables activas entre las variables x son las que corresponden a colores pertenecientes a la distribución de colores y, cada uno de estos colores está asignado exactamente a un sólo vértice de K, ya que K es una clique. Además, la resta de δ_1 y δ_q asegura que, en el caso de que c_1 o c_q estén usados en la distribución, cada uno de ellos sume sólo 1 (en lugar de 2), tal como lo hacen en el lado izquierdo de (3.4).
Por otro lado, podemos acotar el lado derecho de (3.4) con

$$(q-1) + \sum_{v,w \in K} z_{vw} \geq (q-1) + \sum_{i=1}^{r} (|Q_i| - 1) = (q-1) - r + \sum_{i=1}^{r} |Q_i| \tag{3.6}$$

ya que sabemos que para cada Q_i, se tienen $|Q_i| - 1$ variables de adyacencia activas.
Dicho esto, sólo resta probar que

$$\left[\sum_{i=1}^{r} 2|Q_i| \right] - \delta_1 - \delta_q \leq (q-1) - r + \sum_{i=1}^{r} |Q_i| \tag{3.7}$$

y con ello, usando (3.5) y (3.6), podemos demostrar la validez de (3.4).

Hasta este momento sabemos lo siguiente:

1. La cantidad de colores de Q **asignados** a vértices de K está dada por

$$\sum_{i=1}^{r} |Q_i|$$

2. Entre cada par de conjuntos Q_i y Q_{i+1}, dado que la distribución es minimal, existe al menos un color de Q que no está asignado a ningún vértice de K. Por lo tanto existen al menos $r - 1$ colores de Q **no asignados** a vértices de K.

3. La cantidad de colores de Q **asignados** a vértices de K más la cantidad de colores de Q **no asignados** a vértices de K es igual a la cantidad de colores de Q en total (trivial).

Por lo tanto podemos afirmar que

$$\sum_{i=1}^{r} |Q_i| + (r-1) \leq q \tag{3.8}$$

Más aun, si c_1 y c_q no están asignados a ningún vértice de K entonces estos colores no se están contando en ninguno de los dos sumandos del lado izquierdo de (3.8). Por lo tanto, podemos agregarlos a (3.8) de la siguiente manera:

$$\sum_{i=1}^{r} |Q_i| + (r-1) + (1-\delta_1) + (1-\delta_q) \leq q \tag{3.9}$$

ya que $(1-\delta_j) = 1$ sólo cuando c_j no está asignado a ningún vértice de K.

Finalmente, reescribiendo (3.9) y sumando $\sum_{i=1}^{r} |Q_i|$ a cada lado se obtiene

$$\left[\sum_{i=1}^{r} 2|Q_i|\right] - \delta_1 - \delta_q \leq (q-1) - r + \sum_{i=1}^{r} |Q_i|$$

lo que prueba (3.7) y por lo tanto la desigualdad (3.4) es satisfecha por la solución $y = (x, z)$. Como esta solución es arbitraria entonces (3.4) es válida para $\mathcal{PS}(G, C)$. □

Veremos que bajo ciertas condiciones estas desigualdades definen facetas de $\mathcal{PS}(G, C)$, pero antes caracterizaremos las soluciones $y = (x, z) \in \mathcal{PS}(G, C) \cap \mathbb{Z}^{nt+m}$ que cumplen por igualdad las *CCK inequalities* con el siguiente lema:

Lema 3.2.1. *Si $|C| > 2|K| - |Q| + 1$, entonces una solución $y = (x, z) \in \mathcal{PS}(G, C) \cap \mathbb{Z}^{nt+m}$ cumple por igualdad la* CCK inequality *asociada a K y a Q si y sólo si se cumple que:*

- *para cada par de colores consecutivos de Q, al menos uno de ellos es utilizado en la solución por un vértice de K,*
- *no existen variables de adyacencia $z_{vw} = 1$, con $v, w \in K$ tal que v y/o w reciben colores fuera de Q, y*
- *no existen variables de adyacencia $z_{vw} = 1$ con $v, w \in K$ tal que v y w reciben colores no adyacentes.*

Demostración. Supongamos que existen en y variables de adyacencia $z_{vw} = 1$ con $v, w \in K$ tal que v y/o w no reciben colores adyacentes. Si se fija la variable

en el valor cero, el lado derecho de (3.4) se hace más pequeño y entonces y no cumple (3.4) por igualdad, con locual esto no puede ocurrir.

Supongamos ahora que existen en y variables de adyacencia $z_{vw} = 1$ con $v, w \in K$ tal que v y/o w reciben colores fuera de Q. Si existiera algún color de Q no utilizado por vértices de K, sería posible permutar ese color con el asignado a v (o a w), para poder poner $z_{vw} = 0$. De esta manera se hace más estrecha la diferencia entre el lado izquierdo y el derecho de *MCCK*, y por lo tanto la solución y no cumplía (3.4) por igualdad.
En el caso en que todos los colores de Q están usados por vértices de la clique, permutando los colores fuera de Q es posible hacer que esta adyacencia desaparezca, reduciendo así nuevamente la diferencia entre el lado derecho y el lado izquierdo de (3.4) (pues este último se mantiene igual), y probando así que la solución y no cumplía por igualdad (3.4). Para que estas permutaciones sean posibles hace falta disponer de suficientes colores fuera de Q para poder pintar sin adyacencias los colores de K que no se hayan pintado con colores de Q. Además, no se puede utilizar tampoco el color inmediatamente anterior a c_1 ni el inmeditamente posterior a c_q (pues los mismos generarían adyacencias). Es decir,

$$\begin{aligned} |C \setminus (Q \cup \{c_1 - 1, c_q + 1\})| \geq 2(|K| - |Q|) - 1 \\ \iff \\ |C| - |Q| - 2 \geq 2|K| - 2|Q| - 1 \\ \iff \\ |C| \geq 2|K| - |Q| + 1, \end{aligned}$$

lo cual está garantizado por las hipótesis del lema.

Asumimos entonces que en cualquier solución que cumple por igualdad (3.4), no existen variables de adyacencia $z_{vw} = 1$ con $v, w \in K$ tal que v y w reciben colores no adyacentes y tampoco tal que v y/o w reciben colores fuera de Q.

Sea $Q_1, \ldots, Q_r$ la distribución minimal de colores de Q en K, y $\overline{Q}_1, \ldots, \overline{Q}_{\overline{r}}$ la distribución inversa minimal de colores de Q en K.
Dado que las únicas variables activas entre las variables x de (3.4) son las que corresponden a colores pertenecientes a la distribución de colores y que cada uno de estos colores está asignado exactamente a un solo vértice de K, es posible reescribir el lado izquierdo de la desigualdad como

$$\sum_{v \in K} \left[x_{vc_1} + x_{vc_q} + \sum_{c \in Q \setminus \{c_1, c_q\}} 2x_{vc} \right] = \left[\sum_{i=1}^{r} 2|Q_i| \right] - \delta_1 - \delta_q \qquad (3.10)$$

donde δ_j es igual a 1 si c_j está asignado a un vértice de K y es igual a 0 si no. La resta de δ_1 y δ_q asegura que, en el caso de que c_1 o c_q estén usados en la distribución, cada uno de ellos sume sólo 1 (en lugar de 2), tal como lo hacen en el lado izquierdo de (3.4).

Además, sabemos que con los colores de la distribución y de la distribución inversa se cubren todos los colores de Q, es decir

$$|Q| = q = \sum_{i=1}^{r} |Q_i| + \sum_{j=1}^{\overline{r}} |\overline{Q}_j|, \tag{3.11}$$

y por otro lado, sabemos que las adyacencias de color en las aristas de la clique provienen sólo de los colores utilizados en la distribución de colores, con lo cual

$$\sum_{v,w \in K} z_{vw} = \sum_{i=1}^{r} (|Q_i| - 1). \tag{3.12}$$

De esta manera, utilizando (3.10), (3.11) y (3.12) para reescribir (3.4), se puede ver que la misma se cunple por igualdad si y sólo si

$$\left[\sum_{i=1}^{r} 2|Q_i|\right] - \delta_1 - \delta_q = \left(\sum_{i=1}^{r} |Q_i| + \sum_{j=1}^{\overline{r}} |\overline{Q}_j|\right) - 1 + \sum_{i=1}^{r} (|Q_i| - 1), \tag{3.13}$$

y se cumplen las condiciones del lema. Cancelando los términos que aparecen a ambos lados de (3.13) podemos reescribirla como

$$\sum_{j=1}^{\overline{r}} |\overline{Q}_j| = r + 1 - \delta_1 - \delta_q = (r-1) + (1-\delta_1) + (1-\delta_q). \tag{3.14}$$

Es decir, que la cantidad de colores de Q no utilizados en vértices de K, además de eventualmente c_1 y c_q, son exactamente $r-1$, y esto ocurre si y sólo si hay solamente un color entre cada par de conjuntos Q_i y Q_{i+1} correspondientes a la distribución de colores. □

Teorema 3.2.1. *Si se cumplen* $|C| > \chi(G)$, $|C| > |Q|$, $|C| \geq |K| + 5$, $|C| \geq 2|K| - |Q| + 3$, *y* $|K| \geq \frac{|Q|}{2} + 1$, *entonces las* consecutive colors clique inequalities *definen facetas de* $\mathcal{PS}(G, C)$.

Demostración. Sea $\pi \in \mathbb{R}^{nt+m}$ el vector de coeficientes de (3.4) y $\pi_0 \in \mathbb{R}$ el término independiente de la misma. La cara de $\mathcal{PS}(G, C)$ definida por esta desigualdad es entonces

$$F = \{y \in \mathbb{R}^{nt+m} : \pi y = \pi_0\} \cap \mathcal{PS}(G, C).$$

Sean $\lambda \in \mathbb{R}^{nt+m}$ y $\lambda_0 \in \mathbb{R}$ tales que $\lambda y = \lambda_0, \forall y \in F$. Veremos que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo, con lo cual F es una faceta de $\mathcal{PS}(G, C)$.

Para demostrar que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo veremos que los coeficientes de λ

cumplen con ciertas condiciones, las cuales se van demostrando a continuación.

Dada una arista $vw \in E$, tal que $v, w \notin K$, veremos que $\lambda_{z_{vw}} = 0$. Tomemos una solución $y = (x, z) \in \mathcal{PS}(G, C)$ que cumpla (3.4) por igualdad en la cual los vértices v y w reciban colores no adyacentes y $z_{vw} = 0$. Esta solución puede construirse permutando el color asignado a uno de estos vértices para lograr la no adyacencia. Luego, construimos la solución $y' = (x', z') \in \mathcal{PS}(G, C)$ a partir de la solución y pero poniendo $z_{vw} = 1$. Esta solución sigue siendo válida, y dado que z_{vw} no participa de (3.4), la misma sigue cumpliéndose por igualdad. Dado que tanto y como y' pertenecen a F, sabemos que $\lambda y = \lambda_0$ y $\lambda y' = \lambda_0$ y por lo tanto $\lambda y = \lambda y'$. Sabemos además que y coincide con y' en todos sus elementos salvo en z_{vw}, con lo cual se deduce que

$$\lambda_{z_{vw}} z_{vw} = \lambda_{z_{vw}} z'_{vw},$$

y por lo tanto

$$\textit{[Condición 1]} \qquad \lambda_{z_{vw}} = 0 \quad \forall vw \in E \text{ tal que } v, w \notin K. \tag{3.15}$$

Tomemos ahora un vértice $v \notin K$. Veremos que $\lambda_{x_{vc}} = \lambda_{x_{vc'}}$ para cualquier par de colores $c, c' \in C$. Dado que $|C| > \chi(G)$, es posible construir una solución en la cual el color c' no esté utilizado por ningún vértice del grafo. Consideremos entonces una solución $y = (x, z) \in \mathcal{PS}(G, C)$ que cumple (3.4) por igualdad en la cual c' esté libre de vértices y en la cual el vértice v reciba el color c (esto puede conseguirse permutando los colores de la solución). Luego, construimos una solución $y' = (x', z')$ a partir de y, en la cual el vértice v recibe el color c'. Podemos visualizar la diferencia entre estas dos soluciones con el siguiente diagrama:

		c		c'	
$y \rightarrow$		v			
$y' \rightarrow$				v	

Esta solución es válida pues este color no estaba ultilizado por ningún otro vértice del grafo, y además esta solución pertenece a la cara F ya que, como $v \notin K$, sigue cumpliendo (3.4) por igualdad.
Dado que tanto y como y' pertenecen a F, sabemos que $\lambda y = \lambda_0$ y $\lambda y' = \lambda_0$ y por lo tanto $\lambda y = \lambda y'$. Sabemos además que y coincide con y' en todos sus elementos salvo en x_{vc}, $x_{vc'}$ y en algunas variables de adyacencia incidentes a v. Sin embargo, por (3.15) sabemos que $\lambda_{z_{vw}} = 0, \forall vw \in E$, y entonces podemos deducir que

$$\lambda_{x_{vc}} x_{vc} + \lambda_{x_{vc'}} x_{vc'} = \lambda_{x_{vc}} x'_{vc} + \lambda_{x_{vc'}} x'_{vc'},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\textit{[Condición 2]} \qquad \lambda_{x_{vc}} = \lambda_{x_{vc'}} \quad \forall v \notin K \quad \forall c, c' \in C. \tag{3.16}$$

A continuación, consideremos un vértice $v \in K$. Veremos en esta ocasión que $\lambda_{x_{vc_1}} = \lambda_{x_{vc_q}} = \lambda_{x_{vc}} - \lambda_{z_{vw}}$ para algún color $c \notin Q$ y algún vértice $w \in K$. Veremos primero lo que ocurre para $\lambda_{x_{vc_1}}$.
Tomemos una solución $y = (x, z) \in \mathcal{PS}(G, C)$ que cumpla (3.4) por igualdad en la cual el vértice v utilice el color c_1 y en la cual un vértice $w \in K$ utilice el color c_2. Además, pediremos que haya un color $c \notin Q$ libre de vértices (esto puede construirse gracias a que $|C| > \chi(G)$) y los colores a ambos lados de c no estén asignados a vértices de la clique (para esto es necesario que $|C| \geq 2|K| - |Q| + 3$, lo cual está garantizado por las hipótesis del teorema). Luego, construimos una segunda solución $y' = (x', z')$ a partir de y pero pintando en este caso el vértice v con el color c.

		c			c_1	c_2	$\dots$	c_q	
$y \rightarrow$	$\emptyset_K$	$\emptyset$	$\emptyset_K$	$\emptyset_K$	v	w			
$y' \rightarrow$	$\emptyset_K$	v	$\emptyset_K$	$\emptyset_K$		w			

Esta solución sigue siendo válida ya que c no estaba utilizado en ningún vértice. Además, por el Lema 3.2.1 la solución sigue cumpliendo (3.4) por igualdad, con lo cual pertenece a la cara F y por lo tanto, al igual que en casos anteriores, se puede ver que $\lambda y = \lambda y'$. En este caso, y coincide con y' en todas sus variables salvo en x_{vc_1}, x_{vc}, z_{vw} y otras variables de adyacencia incidentes a v. Estas últimas variables sin embargo quedarán anuladas pues, por (3.15), los coeficientes de λ para esas variables son nulos. Se puede decir entonces que

$$\lambda_{x_{vc_1}} x_{vc_1} + \lambda_{x_{vc}} x_{vc} + \lambda_{z_{vw}} z_{vw} = \lambda_{x_{vc_1}} x'_{vc_1} + \lambda_{x_{vc}} x'_{vc} + \lambda_{z_{vw}} z'_{vw},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\lambda_{x_{vc_1}} + \lambda_{z_{vw}} = \lambda_{x_{vc}}. \tag{3.17}$$

Analogamente, utilizando los colores c_q y c_{q-1} para v y w, respectivamente se prueba que

$$\lambda_{x_{vc_q}} + \lambda_{z_{vw}} = \lambda_{x_{vc}}, \tag{3.18}$$

y por lo tanto, de (3.17) y (3.18) decimos que

$$\textit{[Condición 3]} \qquad \lambda_{x_{vc_1}} = \lambda_{x_{vc_q}} = \lambda_{x_{vc}} - \lambda_{z_{vw}} \quad \forall v \in K, \tag{3.19}$$
$$\text{para algún } c \notin Q \text{ y algún } w \in K.$$

Vale aclarar que el color c puede ser, permutando los colores, cualquier color

fuera de Q (para el caso de $c = c_1 - 1$ o $c = c_q + 1$ se puede utilizar la igualdad $\lambda_{x_{vc_1}} = \lambda_{x_{vc_q}}$ como prueba). Con esto, y teniendo en cuenta que el vértice w es arbitrario, es fácil ver que

$$\textit{[Condición 4]} \qquad \lambda_{x_{wc}} = \lambda_{x_{wc'}} \quad \forall w \in K \quad \forall c, c' \notin Q. \tag{3.20}$$

Consideremos nuevamente un vértice $v \in K$ y ahora también un color $c \in Q \setminus \{c_1, c_q\}$. Veremos que $\lambda_{x_{vc}} = \lambda_{x_{vc'}} - \lambda_{z_{vw_1}} - \lambda_{z_{vw_2}}$ para algún color $c' \notin Q$ y algún par de vértices $w_1, w_2 \in K$.
Tomemos una solución $y = (x, z) \in \mathcal{PS}(G, C)$ que cumpla (3.4) por igualdad en la cual el vértice v utilice el color c y en la cual dos vértices $w_1, w_2 \in K$ utilicen los colores adyacentes a c. Además, pediremos que haya un color $c' \notin Q$ libre de vértices (esto puede construirse gracias a que $|C| > \chi(G)$) y los colores a ambos lados de c' no estén asignados a vértices de la clique (para esto es necesario que $|C| \geq 2|K| - |Q| + 3$, lo cual está garantizado por las hipótesis del teorema). Luego, construimos una segunda solución $y' = (x', z')$ a partir de y pero pintando en este caso el vértice v con el color c'.

		c'		c_1	$\dots$		c		$\dots$	c_q	
$y \rightarrow$	$\emptyset_K$	$\emptyset$	$\emptyset_K$			w_1	v	w_2			
$y' \rightarrow$	$\emptyset_K$	v	$\emptyset_K$			w_1		w_2			

Esta solución sigue siendo válida ya que c no estaba utilizado en ningún vértice. Además, por el Lema 3.2.1 la solución sigue cumpliendo (3.4) por igualdad, con lo cual pertenece a la cara F y por lo tanto se puede ver que $\lambda y = \lambda y'$. En este caso, y coincide con y' en todas sus variables salvo en x_{vc}, $x_{vc'}$, z_{vw_1}, z_{vw_2} y otras variables de adyacencia incidentes a v que no participan en la desigualdad y por lo tanto, por (3.15), los coeficientes de λ para ellas son nulos. Se puede decir entonces que

$$\begin{aligned}\lambda_{x_{vc}} x_{vc} + \lambda_{x_{vc'}} x_{vc'} + \lambda_{z_{vw_1}} z_{vw_1} + \lambda_{z_{vw_2}} z_{vw_2} = \\ = \lambda_{x_{vc}} x'_{vc} + \lambda_{x_{vc'}} x'_{vc'} + \lambda_{z_{vw_1}} z'_{vw_1} + \lambda_{z_{vw_2}} z'_{vw_2},\end{aligned}$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\textit{[Condición 5]} \qquad \lambda_{x_{vc}} = \lambda_{x_{vc'}} - \lambda_{z_{vw_1}} - \lambda_{z_{vw_2}} \quad \forall v \in K, \forall c \in Q \setminus \{c_1, c_q\}, \tag{3.21}$$

para algún $c' \notin Q$ y algún par de vértices $w_1, w_2 \in K$.

y dado que los vértices w_1 y w_2 son arbitrarios, la condición anterior implica que

$$\lambda_{z_{vw_1}} = \lambda_{z_{vw_2}} \quad \forall v, w_1, w_2 \in K,$$

lo cual a su vez implica lo siguiente:

$$\textit{[Condición 6]} \qquad \lambda_{z_{v_1 w_1}} = \lambda_{z_{v_2 w_2}} \quad \forall v_1, v_2, w_1, w_2 \in K. \tag{3.22}$$

Con esto finaliza la demostración de condiciones particulares sobre los coeficientes de λ. Veremos ahora cómo estas condiciones implican el hecho de que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo.
La condición (3.16) indica que dado un vértice v fuera de la clique, el coeficiente de λ para la variable x_{vc} es el mismo para cualquier color c. Llamemos entonces, β_v a este valor. Es decir, dado $v \notin K$

$$\lambda_{x_{vc}} = \beta_v \quad \forall c \in C.$$

Por otro lado, la condición (3.20) indica que dado un vértice $v \in K$, el coeficiente de λ para la variable x_{vc} es el mismo para cualquier color $c \in C \setminus Q$. Llamemos entonces, β_v a este valor[1], o sea

$$\lambda_{x_{vc}} = \beta_v \quad \forall c \in C \setminus Q.$$

En la condición (3.22) se ve que $\lambda_{z_{v_1 w_1}} = \lambda_{z_{v_2 w_2}}$ para todo $v_1, v_2, w_1, w_2 \in K$. Si llamamos α a este valor obtenemos que

$$\lambda_{z_{vw}} = \alpha \quad \forall v, w \in K.$$

Definidos α y β_v para cada $v \in V$, podemos ver qué valores toman el resto de los coeficientes de λ en función de éstos.
Tomando la condición (3.19), podemos ver que

$$\lambda_{x_{vc_1}} = \lambda_{x_{vc_q}} = \beta_v - \alpha \quad \forall v \in K,$$

y utilizando la condición (3.21), se ve que

$$\lambda_{x_{vc}} = \beta_v - 2\alpha \quad \forall v \in K, \forall c \in Q \setminus \{c_1, c_q\}.$$

De esta manera, teniendo en cuenta además la condición (3.15), se puede ver que λ se puede formar como la combinación lineal

$$\sum_{v \in V} \beta_v e^{(v)} - \alpha\pi,$$

donde $e^{(v)}$ representa el vector de coeficientes de la ecuación de (2.1) correspondiente al vértice v. Por lo tanto, las *consecutive-colors clique inequalitites* definen facetas de $\mathcal{PS}(G, C)$. □

[1] Vale aclarar que, dado que β_v depende de v, estos pueden ser distintos para cada vértice.

Un aspecto interesante de esta familia de desigualdades son las desigualdades que surgen al utilizar $K = \{v, w\}$ para alguna arista $vw \in E$, y $Q = \{c_1, c_2\}$, donde c_1 y c_2 son colores consecutivos de C. La desigualdad en este caso es

$$(x_{vc_1} + x_{vc_2}) + (x_{wc_1} + x_{wc_2}) \leq 1 + z_{vw},$$

y puede verse que la misma domina a las desigualdades (2.3) del modelo. Así, en una implementación de este modelo, sería conveniente reemplazar (2.3) por las *CCK inequalities* dadas para cada arista del grafo y cada par ordenado de colores consecutivos de C. De esta manera, además de utilizar restricciones más fuertes, se reduce la cantidad de restricciones ya que para cada par de colores consecutivos sólo se necesita una restricción por arista en lugar de dos (como se da en el modelo original).

3.2.2. Multi-consecutive colors clique (MCCK) inequality

La siguiente desigualdad válida es en realidad una generalización de la *CCK inequality*, y la idea de la misma consiste en utilizar más de un conjunto de colores consecutivos para acotar las variables de adyacencia. Antes de dar la definición formal, se la discute brevemente.

Consideremos nuevamente una clique dada por el conjunto de vértices $K \subseteq V$ y, en esta ocasión, consideremos p conjuntos no contiguos de colores consecutivos $Q^1, Q^2, \ldots, Q^p \subseteq C$. Sabemos que la *CCK inequality* definida para K y cualquiera de estos conjuntos es válida para cualquier solución $y = (x, z) \in \mathcal{PS}(G, C)$. Por otro lado, se puede ver que cada una de las aristas de K puede ser relevante para la desigualdad sólo si ambos extremos de la misma reciben colores dentro del mismo subconjunto Q_i, pues de no ser así los colores asignados no serían adyacentes[2]. Usando estas dos ideas podemos ver que la cota para la suma de las variables de adyacencia de la clique puede estar dada por la suma de las cotas sobre todos los subconjuntos dados. De esta manera llegamos a la siguiente generalización de la *CCK*.

Definición 3.2.2. *Sea $K \subseteq V$ un conjunto de vértices que inducen un subgrafo completo de G, y sean $Q^1 = \{c^1_1, \ldots, c^1_{q_1}\}, Q^2 = \{c^2_1, \ldots, c^2_{q_2}\}, \ldots, Q^p = \{c^p_1, \ldots, c^p_{q_p}\} \subseteq C$ p conjuntos no contiguos de colores consecutivos tales que para todo $h \in \{1, \ldots, p\}$, se cumple $c^h_{i+1} = c^h_i + 1$ para $i = 1, \ldots, q_h - 1$. Definimos la* multi-consecutive colors clique inequality *asociada a K, $Q^1, \ldots, Q^{p-1}$ y Q^p como*

$$\sum_{h=1}^{p} \sum_{v \in K} \left[x_{vc^h_1} + x_{vc^h_{q_h}} + \sum_{c \in Q^h \setminus \{c^h_1, c^h_{q_h}\}} 2x_{vc} \right] \leq \sum_{h=1}^{p} (q_h - 1) + \sum_{v,w \in K} z_{vw}. \quad (3.23)$$

Vale mencionar que (3.23) no es simplemente la suma de las desigualdades CCK correspondientes a $Q^1, \ldots, Q^{p-1}$ y Q^p. Esto puede verse sencillamente

[2]Nótese que los conjuntos de colores son no contiguos.

viendo que, si bien el lado izquierdo de (3.23) y parte de su lado derecho sí lo son, la sumatoria de las variables de adyacencia aparece una sóla vez en (3.23).

Proposición 3.2.2. *Las* multi-consecutive colors clique inequalities *son válidas para* $PS(G, C)$.

Demostración. La idea de esta demostración es similar a la demostración de validez de las *CCK*. Para $h = 1, \dots, p$ y $j = 1, \dots, q_h$ definimos $\delta_j^h \in \{0, 1\}$ como

$$\delta_j^h = \begin{cases} 1 & \text{si el color } c_j^h \text{ está asignado a algún vértice de } K, \\ 0 & \text{si no.} \end{cases}$$

Para cada $h = 1, \dots, p$, sea $Q_1^h, Q_2^h, \dots, Q_{r_h}^h$ la distribución minimal de colores de Q^h en K. Se puede ver que

$$\sum_{v \in K} \left[x_{vc_1^h} + x_{vc_q^h} + \sum_{c \in Q^h \setminus \{c_1^h, c_{q_h}\}} 2x_{vc} \right] = \left[\sum_{i=1}^{r_h} 2|Q_i^h| \right] - \delta_1^h - \delta_q^h \tag{3.24}$$

pues las únicas variables activas entre las variables x son las que corresponden a colores pertenecientes a la distribución de colores y, cada uno de estos colores está asignado exactamente a un sólo vértice de K, ya que K es una clique. Además, la resta de δ_1^h y δ_q^h asegura que, en el caso de que c_1^h o c_q^h estén usados en la distribución, cada uno de ellos sume sólo 1 (en lugar de 2), tal como lo hacen en el lado izquierdo de la ecuación.
Sumando las ecuaciones (3.24) para cada $h = 1, \dots, p$ obtenemos

$$\sum_{h=1}^{p} \left(\sum_{v \in K} \left[x_{vc_1^h} + x_{vc_q^h} + \sum_{c \in Q^h \setminus \{c_1^h, c_{q_h}\}} 2x_{vc} \right] \right) = \sum_{h=1}^{p} \left(\left[\sum_{i=1}^{r_h} 2|Q_i^h| \right] - \delta_1^h - \delta_q^h \right) \tag{3.25}$$

Por otro lado, podemos acotar el lado derecho de (3.23) con

$$\begin{aligned} \sum_{h=1}^{p} (q_h - 1) + \sum_{v,w \in K} z_{vw} &\geq \sum_{h=1}^{p} (q_h - 1) + \sum_{h=1}^{p} \left(\sum_{i=1}^{r_h} (|Q_i^h| - 1) \right) \\ &= \sum_{h=1}^{p} \left((q_h - 1) - r_h + \sum_{i=1}^{r_h} |Q_i^h| \right) \end{aligned} \tag{3.26}$$

ya que sabemos que al menos para cada Q_i^h, se tienen $|Q_i^h| - 1$ variables de adyacencia activas. Dicho esto, sólo resta probar que

$$\sum_{h=1}^{p} \left(\left[\sum_{i=1}^{r_h} 2|Q_i^h| \right] - \delta_1^h - \delta_q^h \right) \leq \sum_{h=1}^{p} \left((q_h - 1) - r_h + \sum_{i=1}^{r_h} |Q_i^h| \right) \tag{3.27}$$

y con ello, usando (3.25) y (3.26), podemos demostrar la validez de (3.23).

Hasta este momento podemos afirmar lo siguiente para cada Q^h:

1. La cantidad de colores de Q^h **asignados** a vértices de K está dada por
$$\sum_{i=1}^{r_h} |Q_i^h|$$
2. Entre cada par de conjuntos Q_i^h y Q_{i+1}^h, dado que la distribución es minimal, existe al menos un color de Q^h que no está asignado a ningún vértice de K. Por lo tanto existen al menos $r_h - 1$ colores de Q^h **no asignados** a vértices de K.
3. La cantidad de colores de Q^h **asignados** a vértices de K más la cantidad de colores de Q^h **no asignados** a vértices de K es igual a la cantidad de colores de Q^h en total (trivial).

Por lo tanto podemos afirmar que

$$\sum_{i=1}^{r_h} |Q_i^h| + (r_h - 1) \leq q_h, \tag{3.28}$$

para todo $h = 1, \ldots, p$. Más aun, si c_1^h y c_q^h no están asignados a ningún vértice de K entonces estos colores no se están contando en ninguno de los dos sumandos de (3.28). Por lo tanto, podemos agregarlos a (3.27) de la siguiente manera:

$$\sum_{i=1}^{r_h} |Q_i^h| + (r_h - 1) + (1 - \delta_1^h) + (1 - \delta_q^h) \leq q_h, \tag{3.29}$$

ya que $(1 - \delta_j^h) = 1$ sólo cuando c_j^h no está asignado a ningún vértice de K.

Reescribiendo (3.29) y sumando $\sum_{i=1}^{r_h} |Q_i^h|$ a cada lado se obtiene

$$\left[\sum_{i=1}^{r_h} 2|Q_i^h|\right] - \delta_1^h - \delta_q^h \leq (q_h - 1) - r_h + \sum_{i=1}^{r_h} |Q_i^h|, \tag{3.30}$$

y finalmente sumando (3.30) sobre todos los $h = 1, \ldots, p$ obtenemos

$$\sum_{h=1}^{p} \left(\left[\sum_{i=1}^{r_h} 2|Q_i^h|\right] - \delta_1^h - \delta_q^h \right) \leq \sum_{h=1}^{p} \left((q_h - 1) - r_h + \sum_{i=1}^{r_h} |Q_i^h| \right),$$

lo que prueba (3.27) y por lo tanto la desigualdad (3.23) es válida para la solución $y = (x, z)$. □

Conjeturamos que bajo hipótesis similares a las de la *CCK*, las *MCCK inequalities* definen facetas de $\mathcal{PS}(G, C)$. Si bien la idea de esta demostración es muy similar a la del Teorema 3.2.1, las hipótesis necesarias para la misma deben detallarse con mayor cuidado. Dado que la demostración de este teorema no se verificó con este nivel de detalle, preferimos no enunciarlo en este trabajo.

3.2.3. 3-colors inner clique (3CIK) inequality

La siguiente desigualdad válida acota, al igual que las anteriores, las variables de adyacencia de las aristas de una clique, pero en este caso sólo se tienen en cuenta las aristas incidentes a un vértice distinguido de la clique. También se utiliza un conjunto de colores consecutivos, pero en esta ocasión es un conjunto de sólo tres colores.

Consideremos entonces una clique dada por un conjunto de vértices $K \subseteq V$, un vértice distinguido $k \in K$ y un conjunto de tres colores consecutivos $Q = \{c_1, c_2, c_3\} \subseteq C$. Supongamos que se utiliza c_1 para colorear el vértice k, entonces si se utiliza c_2 para colorear algún otro vértice de la clique se obtendrá una adyacencia para la arista que une k con dicho vértice. Lo mismo se dará si en lugar de pintar k con c_1 se lo pinta con c_3. Es decir,

$$(x_{kc_1} + x_{kc_3}) = 1 \quad \wedge \quad \sum_{v \in K \setminus \{k\}} x_{vc_2} = 1 \quad \Rightarrow \sum_{v \in K \setminus \{k\}} z_{vk} \geq 1.$$

Esta implicación puede escribirse como una desigualdad lineal de la siguiente manera:

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq (x_{kc_1} + x_{kc_3}) \quad + \sum_{v \in K \setminus \{k\}} x_{vc_2} \quad - \quad 1. \tag{3.31}$$

Como se mostró, la desigualdad (3.31) se deduce lógicamente de ciertas características del problema en cuestión, que involucran a las variables mencionadas en la misma. Sin embargo, es posible fortalecer la desigualdad mediante un procedimiento conocido como *lifting*. Este procedimiento consiste en tomar una variable no utilizada en la desigualdad e incorporarla a la misma. Lo que se intenta es incorporarla con el coeficiente que mejor ajuste la desigualdad, manteniendo por supuesto la validez de la misma. Un aspecto interesante de este procedimiento es que si la desigualdad original definía facetas del poliedro resultante de fijar en cero la variable a incorporar, entonces la nueva desigualdad define facetas del poliedro original. La demostración de este teorema puede encontrarse en [18] junto con una excelente explicación de este tipo de procedimientos.

La siguiente definición surge de un proceso de *lifting* de la variable x_{kc_2} en (3.31), con el objetivo de fortalecer esta desigualdad válida.

Definición 3.2.3. *Sea $K \subseteq V$ un conjunto de vértices que inducen un subgrafo completo de G, $k \in K$ un vértice distinguido y $Q = \{c_1, c_2, c_3\} \subseteq C$ un conjunto de tres colores consecutivos tal que $c_{i+1} = c_i + 1$, para $i = 1, 2$. Definimos la* 3-colors inner clique inequality *asociada a K, k y Q como*

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq (x_{kc_1} + x_{kc_2} + x_{kc_3}) \quad + \sum_{v \in K \setminus \{k\}} x_{vc_2} \quad - \quad 1. \tag{3.32}$$

Proposición 3.2.3. *Las* 3-colors inner clique inequalities *son válidas para $\mathcal{PS}(G, C)$.*

Demostración. Es sencillo ver que si el vértice k no se pinta con alguno de los tres colores de Q, entonces la desigualdad se cumple trivialmente ya que, por un lado $(x_{kc_1} + x_{kc_2} + x_{kc_3}) = 0$ y por otro, dado que K representa una clique, se puede ver que

$$\sum_{v \in K \setminus \{k\}} x_{vc_2} \leq 1,$$

lo cual deja al lado derecho de la restricción con un valor nulo o negativo.
Por este motivo basta concentrar la atención en los casos en que al vértice k se le asigna algún color de Q. En estos casos $(x_{kc_1} + x_{kc_2} + x_{kc_3}) = 1$, y por lo tanto deberíamos demostrar que

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} x_{vc_2}. \tag{3.33}$$

Si el color asignado al vértice k fuese c_2, entonces ningún otro vértice de K podría tener el mismo color asignado, con lo cual (3.33) se cumple trivialmente, ya que su lado derecho es cero. Si, en cambio, el color asignado a k fuese c_1, entonces podría haber otro vértice $v \in K$ tal que $x_{vc_2} = 1$, pero en ese caso la variable de adyacencia z_{vk} tomará también el valor 1 y (3.33) se sguirá cumpliendo. Análogamente, ocurre lo mismo si el color asignado a k es c_3 en lugar de c_1. □

Es importante notar, en particular para la implementación y el uso de esta desigualdad válida, que tanto c_1 como c_3 pueden no existir y la desigualdad seguirá siendo válida. Es decir, si se toman sólo los dos primeros colores de C como c_2 y c_3 respectivamente, la desigualdad sigue siendo válida frente a la ausencia de x_{vc_1}. Lo mismo ocurre si se toman los últimos dos colores de C para c_1 y c_2, respectivamente, y en este caso la variable ausente sería x_{vc_3}. La validez de estos hechos queda implicada por la dominancia de (3.32) sobre la misma desigualdad con estas variables ausentes.

Teorema 3.2.2. *Si* $|C| > \chi(G)$ *y* $|C| \geq |K| + 5$*, entonces las* 3-colors inner clique inequalities *definen facetas de* $\mathcal{PS}(G, C)$*.*

Demostración. Sea $\pi \in \mathbb{R}^{nt+m}$ el vector de coeficientes de (3.32) y $\pi_0 \in \mathbb{R}$ el término independiente de la misma. La cara de $\mathcal{PS}(G, C)$ definida por esta desigualdad es entonces

$$F = \{y \in \mathbb{R}^{nt+m} : \pi y = \pi_0\} \cap \mathcal{PS}(G, C).$$

Sean $\lambda \in \mathbb{R}^{nt+m}$ y $\lambda_0 \in \mathbb{R}$ tales que $\lambda y = \lambda_0, \forall y \in F$. Veremos que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo, con lo cual F es una faceta de $\mathcal{PS}(G, C)$.

Antes de comenzar la demostración caracterizaremos las soluciones enteras $y = (x, z) \in F$ en tres tipos:

Tipo (I) El vértice k no utiliza ningún color de Q y algún otro vértice de la clique utiliza c_2. Además, ningún vértice de la clique utiliza un color adyacente al utilizado por k. Es decir,

- $x_{kc_1} = x_{kc_2} = x_{kc_3} = 0$,
- $x_{vc_2} = 1$ para algún $v \in K \setminus \{k\}$ y
- $z_{kv} = 0$ para todo $v \in K \setminus \{k\}$.

Tipo (II) El vértice k utiliza c_2. En este caso para que se cumpla la igualdad los demás vértices de la clique no deben utilizar c_1 o c_3 para no generar adyacencias de colores, y tampoco c_2 pues lo utiliza k. Es decir,

- $x_{kc_2} = 1$ y
- $x_{vc_1} = x_{vc_2} = x_{vc_3} = 0$ para todo $v \in K \setminus \{k\}$.

Tipo (III) El vértice k utiliza c_1 o c_3, y las únicas adyacencias de color dadas entre k y los demás vértices de la clique están dadas por el uso de c_2 por parte de los demás vértices. Es decir,

- $x_{kc_1} = 1$ o $x_{kc_3} = 1$, y
- $z_{kv} = 1 \Leftrightarrow x_{vc_2} = 1$ para todo $v \in K \setminus \{k\}$.

Llamaremos tipo (IIIa) y tipo (IIIb) a las soluciones de este tipo en las cuales k utilice c_1 y c_3, respectivamente.

A continuación se muestran gráficamente los tipos de soluciones descriptos. Los espacios en blanco de la tabla indican que no hay restricciones para los vértices que pueden recibir el color en cuestión y el símbolo $\emptyset_K$ indica que el mismo no está asignado a ningún vértice de la clique. Tanto v como w representan colores de la clique.

		c		c_1	c_2	c_3	
$(I) \rightarrow$	$\emptyset_K$	k	$\emptyset_K$		v		
$(II) \rightarrow$				$\emptyset_K$	k	$\emptyset_K$	
$(IIIa) \rightarrow$			$\emptyset_K$	k			
$(IIIb) \rightarrow$						k	$\emptyset_K$

Para demostrar que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo veremos que los coeficientes de λ cumplen con ciertas condiciones, las cuales se van demostrando a continuación.

Dada una arista $vw \in E$, tal que $vw \notin \{kj : j \in K\}$, veremos que $\lambda_{z_{vw}} = 0$. Tomemos una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (II) en la cual los vértices v y w reciban colores no adyacentes y $z_{vw} = 0$. Esta solución puede construirse permutando el color asignado a uno de estos vértices para lograr la no adyacencia. Luego, construimos la solución $y' = (x', z') \in \mathcal{PS}(G, C)$ a partir de la

solución y pero poniendo $z'_{vw} = 1$. Esta solución sigue siendo válida, y dado que z_{vw} no participa de (3.32), la misma sigue cumpliéndose por igualdad.
Dado que tanto y como y' pertenecen a F, sabemos que $\lambda y = \lambda_0$ y $\lambda y' = \lambda_0$ y por lo tanto $\lambda y = \lambda y'$. Sabemos además que y coincide con y' en todos sus elementos salvo en z_{vw}, con lo cual se deduce que

$$\lambda_{z_{vw}} z_{vw} = \lambda_{z_{vw}} z'_{vw},$$

y por lo tanto

$$\textit{[Condición 1]} \qquad \lambda_{z_{vw}} = 0 \quad \forall vw \in E \text{ tal que } vw \notin \{kj : j \in K\}. \tag{3.34}$$

Tomemos ahora un vértice $v \notin K$. Veremos que $\lambda_{x_{vc}} = \lambda_{x_{vc'}}$ para cualquier par de colores $c, c' \in C$. Dado que $|C| > \chi(G)$, es posible construir una solución en la cual el color c' no esté utilizado por ningún vértice del grafo. Consideremos entonces una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (I) en la cual c' esté libre de vértices[3] y en la cual el vértice v reciba el color c (esto último puede conseguirse permutando el color de v, asumiendo, sin perder generalidad, que el color asignado a v es distinto de c_2). Luego, construimos una solución $y' = (x', z')$ a partir de y, en la cual el vértice v recibe el color c'. Podemos visualizar la diferencia entre estas dos soluciones con el siguiente diagrama:

		c		c'	
$y \rightarrow$		v			
$y' \rightarrow$				v	

Esta solución es válida pues este color no estaba ultilizado por ningún otro vértice del grafo, y además esta solución pertenece a la cara F pues es también una solución de tipo (I).
Dado que tanto y como y' pertenecen a F, sabemos que $\lambda y = \lambda_0$ y $\lambda y' = \lambda_0$ y por lo tanto $\lambda y = \lambda y'$. Sabemos además que y coincide con y' en todos sus elementos salvo en x_{vc}, $x_{vc'}$ y en algunas variables de adyacencia incidentes a v. Sin embargo, por (3.34) sabemos que $\lambda_{z_{vw}} = 0, \forall vw \in E$, y entonces podemos deducir que

$$\lambda_{x_{vc}} x_{vc} + \lambda_{x_{vc'}} x_{vc'} = \lambda_{x_{vc}} x'_{vc} + \lambda_{x_{vc'}} x'_{vc'},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\lambda_{x_{vc}} = \lambda_{x_{vc'}}.$$

Para el caso en que $c' = c_2$, no existen en F soluciones de tipo (I) de manera tal que no se utilice c', sin embargo, se puede demostrar esta igualdad bajo el mismo razonamiento previo utilizando soluciones de tipo (III). Así,

$$\textit{[Condición 2]} \qquad \lambda_{x_{vc}} = \lambda_{x_{vc'}} \quad \forall v \notin K \quad \forall c, c' \in C. \tag{3.35}$$

[3] Para que exista esta solución, asumimos que $c' \neq c_2$; luego veremos este caso particular.

A continuación, consideremos un vértice $v \in K \setminus \{k\}$. Veremos que $\lambda_{x_{vc_2}} = \lambda_{x_{vc}} - \lambda_{z_{kv}}$ para algún color $c \in C \setminus \{c_2\}$. Tomemos en este caso una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (IIIa) en la cual el vértice v utilice el color c_2, y además haya un color $c \in C \setminus \{c_2\}$ libre de vértices (esto puede construirse gracias a que $|C| > \chi(G)$), el cual asumimos por ahora que es distinto de c_1 y de c_3[4]. Luego, construimos una segunda solución $y' = (x', z')$ a partir de y pero pintando en este caso el vértice v con el color c.

	c		c_1	c_2	c_3	
$y \rightarrow$	$\emptyset$		k	v		
$y' \rightarrow$	v		k			

Esta solución sigue siendo válida ya que c no estaba utilizado en ningún vértice. Además, esta solución sigue siendo una solución de tipo (IIIa) con lo cual pertenece a la cara F y por lo tanto, al igual que en casos anteriores, se puede ver que $\lambda y = \lambda y'$. En este caso, y coincide con y' en todas sus variables salvo en x_{vc_2}, x_{vc}, z_{kv} y otras variables de adyacencia para v. Estas últimas variables sin embargo quedarán anuladas pues, por (3.34), los coeficientes de λ para esas variables son nulos. Se puede decir entonces que

$$\lambda_{x_{vc_2}} x_{vc_2} + \lambda_{x_{vc}} x_{vc} + \lambda_{z_{kv}} z_{kv} = \lambda_{x_{vc_2}} x'_{vc_2} + \lambda_{x_{vc}} x'_{vc} + \lambda_{z_{kv}} z'_{kv},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\textit{[Condición 3]} \qquad \lambda_{x_{vc_2}} + \lambda_{z_{kv}} = \lambda_{x_{vc}} \quad \forall v \in K \setminus \{k\}, \text{ para algún } c \in C \setminus \{c_2\}. \tag{3.36}$$

Vale aclarar que el color c puede ser, permutando los colores, cualquier color salvo c_2 (para el caso de $c = c_1$ la demostración es análoga a la anterior pero partiendo de una solución de tipo (IIIb) en lugar de una de tipo (IIIa)). Con esto, es fácil ver que

$$\textit{[Condición 4]} \qquad \lambda_{x_{vc}} = \lambda_{x_{vc'}} \quad \forall v \in K \setminus \{k\} \quad \forall c, c' \in C \setminus \{c_2\}. \tag{3.37}$$

Veremos a continuación que $\lambda_{x_{kc_1}} = \lambda_{x_{kc_2}} = \lambda_{x_{kc_3}} = \lambda_{x_{kc}} - \lambda_{z_{kv}}$ para algún $c \in C \setminus \{c_1, c_2, c_3\}$ y algún $v \in K \setminus \{k\}$. Vemos primero lo que ocurre con $\lambda_{x_{kc1}}$. Para ello tomemos una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (IIIa) en la cual un vértice $v \in K \setminus \{k\}$ utilice el color c_2, y además haya un color $c \in C \setminus \{c_1, c_2, c_3\}$ libre de vértices (esto puede construirse gracias a que

[4] Veremos más adelante cómo puede demostrarse lo mismo para estos casos.

$|C| > \chi(G)$). Pedimos también que los dos colores adyacentes a c no estén asignados a ningún vértice de K; esto es factible si $|C \setminus \{c\}| \geq |K|+2$, y esto último está garantizado por las hipótesis del teorema. Luego, construimos una segunda solución $y' = (x', z')$ a partir de y pero pintando en este caso el vértice k con el color c.

			c			c_1	c_2	c_3	
$y \rightarrow$		$\emptyset_K$	$\emptyset$	$\emptyset_K$		k	v		
$y' \rightarrow$		$\emptyset_K$	k	$\emptyset_K$			v		

Esta solución sigue siendo válida ya que c no estaba utilizado en ningún vértice. Además, esta solución pertenece a la cara F ya que representa una solución de tipo (I) y por lo tanto, al igual que en casos anteriores, se puede ver que $\lambda y = \lambda y'$. En este caso, y coincide con y' en todas sus variables salvo en x_{kc_1}, x_{kc}, z_{kv} y eventualmente algunas otras variables de adyacencia para k, aunque estas últimas variables quedarán anuladas pues, por (3.34), los coeficientes de λ para esas variables son nulos. Se puede decir entonces que

$$\lambda_{x_{kc_1}} x_{kc_1} + \lambda_{x_{kc}} x_{kc} + \lambda_{z_{kv}} z_{kv} = \lambda_{x_{kc_1}} x'_{kc_1} + \lambda_{x_{kc}} x'_{kc} + \lambda_{z_{kv}} z'_{kv},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\lambda_{x_{kc_1}} + \lambda_{z_{kv}} = \lambda_{x_{kc}}, \tag{3.38}$$

para algún $v \in K \setminus \{k\}$ y algún $c \in C \setminus \{c_1, c_2, c_3\}$.
Análogamente, partiendo de una solución de tipo (IIIb) en lugar de una de tipo (IIIa) se demuestra lo mismo para $\lambda_{x_{kc_3}}$. Es decir,

$$\lambda_{x_{kc_3}} + \lambda_{z_{kv}} = \lambda_{x_{kc}}, \tag{3.39}$$

para algún $v \in K \setminus \{k\}$ y algún $c \in C \setminus \{c_1, c_2, c_3\}$.
Veremos ahora que $\lambda_{x_{vc2}} = \lambda_{x_{vc3}}$ para de esa manera completar la quinta condición que debe cumplir λ. Tomemos entonces una solución $y = (x, z)$ de tipo (II) en la cual el color c_3 no esté utilizado. Pedimos además que el color que le sigue a c_3, llamémoslo $c = c_3 + 1$, no esté asignado a ningún vértice de la clique (nuevamente esto es factible ya que $|C| > \chi(G)$ y $|C \setminus \{c_3\}| \geq |K| + 1$). Luego, construimos otra solución $y' = (x', z')$ a partir de y en la cual pintamos k utilizando c_3.

	c_1	c_2	c_3		
$y \rightarrow$		k	$\emptyset$	$\emptyset_K$	
$y' \rightarrow$			k	$\emptyset_K$	

Esta solución es una solución factible pues c_3 estaba libre de vértices. Además es una solución perteneciente a F pues representa una solución de tipo (IIIb). De nuevo, tenemos que $\lambda y = \lambda y'$ y en este caso y coincide con y' en todas

sus variables salvo en x_{kc_2}, x_{kc_3} y eventualmente algunas otras variables de adyacencia para k, aunque estas últimas quedarán anuladas pues los coeficientes de λ para ellas son nulos. Se puede decir entonces que

$$\lambda_{x_{kc_2}} x_{kc_2} + \lambda_{x_{kc_3}} x_{kc_3} = \lambda_{x_{kc_2}} x'_{kc_2} + \lambda_{x_{kc_3}} x'_{kc_3},$$

y reemplazando las variables por los valores que toman en ambas soluciones podemos ver que

$$\lambda_{x_{kc_2}} = \lambda_{x_{kc_3}}. \tag{3.40}$$

Finalmente, combinando (3.38), (3.39) y (3.40) llegamos a

$$\textit{[Condición 5]} \quad \lambda_{x_{kc_1}} = \lambda_{x_{kc_2}} = \lambda_{x_{kc_3}} = \lambda_{x_{kc}} - \lambda_{z_{kv}}, \tag{3.41}$$
$$\text{para algún } v \in K \setminus \{k\}, \text{ y algún } c \in C \setminus \{c_1, c_2, c_3\}.$$

Vale aclarar que el vértice v se tomó de manera arbitraria y puede usarse cualquier vértice $v \in K \setminus \{k\}$, con lo cual utilizando (3.41) variando el vértice elegido se deduce que

$$\textit{[Condición 6]} \quad \lambda_{z_{kv}} = \lambda_{z_{kw}} \quad \forall v, w \in K \setminus \{k\}. \tag{3.42}$$

La última condición a demostrar es el hecho de que $\lambda_{x_{kc}} = \lambda_{x_{kc'}}$ para todo $c, c' \in C \setminus \{c_1, c_2, c_3\}$. Tomemos entonces una solución $y = (x, z)$ de tipo (I) en la cual el vértice k está pintado con el color c; es factible construir esta solución permutando los colores asignados. Además, el color c' debe estar libre de vértices; nuevamente la hipótesis $|C| > \chi(G)$ permite tomar esta solución. Y por último, pediremos que en esta solución tanto los colores adyacentes a c como los adyacentes a c' no estén asignados a ningún vértice de K; esto es factible si $|C \setminus \{c\}| \geq |K| + 4$, lo cual está implicado por las hipótesis. Luego, construimos una solución $y' = (x', z')$ a partir de y en la cual el vértice k utiliza el color c'.

			c				c'		
$y \rightarrow$		$\emptyset_K$	k	$\emptyset_K$		$\emptyset_K$	$\emptyset$	$\emptyset_K$	
$y' \rightarrow$		$\emptyset_K$		$\emptyset_K$		$\emptyset_K$	k	$\emptyset_K$	

Esta solución es una solución válida ya que el color c' no estaba utilizado por ningún vértice. Además, esta solución pertenece a la cara F ya que, al igual que y, es una solución de tipo (I).
Estas dos soluciones coinciden en todos sus valores salvo en los valores de x_{kc}, $x_{kc'}$ y eventualmente algunas variables de adyacencia entre k y vértices fuera de la clique (pues aseguramos que los colores adyacentes a c y a c' no están

asignados a vértices de K). Estas últimas variables resultan anuladas por sus correspondientes coeficientes en λ, con lo cual se puede ver que

$$\lambda_{x_{kc}} x_{kc} + \lambda_{x_{kc'}} x_{kc'} = \lambda_{x_{kc}} x'_{kc} + \lambda_{x_{kc'}} x'_{kc'},$$

y reemplazando las variables por los valores que toman en ambas soluciones se obtiene

$$\textit{[Condición 7]} \quad \lambda_{x_{kc}} = \lambda_{x_{kc'}} \quad \forall c, c' \in C \setminus \{c_1, c_2, c_3\}. \tag{3.43}$$

Terminamos con esto de demostrar las condiciones particulares sobre los coeficientes de λ. Veremos ahora cómo estas condiciones implican el hecho de que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo.
La condición (3.35) indica que dado un vértice v fuera de la clique, el coeficiente de λ para la variable x_{vc} es el mismo para cualquier color c. Llamemos entonces, β_v a este valor. Es decir, dado $v \notin K$

$$\lambda_{x_{vc}} = \beta_v \quad \forall c \in C.$$

Por otro lado, la condición (3.37) indica que dado un vértice $v \in K \setminus \{k\}$, el coeficiente de λ para la variable x_{vc} es el mismo para cualquier color c, salvo para c_2. Llamemos entonces, β_v a este valor[5], o sea

$$\lambda_{x_{vc}} = \beta_v \quad \forall c \in C \setminus \{c_2\}.$$

De manera similar, la condición (3.43) indica que el coeficiente para la variable x_{kc} es el mismo para cualquier color c, salvo para c_1, c_2 y c_3. Llamemos entonces, β_k a este valor, es decir

$$\lambda_{x_{kc}} = \beta_k \quad \forall c \in C \setminus \{c_1, c_2, c_3\}.$$

En la condición (3.42) se ve que $\lambda_{z_{kv}} = \lambda_{z_{kw}}$ para todo par de vértices $v, w \in K \setminus \{k\}$. Si llamamos α a este valor obtenemos que

$$\lambda_{z_{kv}} = \alpha \quad \forall v \in K \setminus \{k\}.$$

Definidos α y β_v para cada $v \in V$, podemos ver qué valores toman el resto de los coeficientes de λ en función de éstos.
Tomando la condición (3.36), podemos ver que

$$\lambda_{x_{vc_2}} = \beta_v - \alpha \quad \forall v \in K \setminus \{k\},$$

y utilizando la condición (3.41), se ve que

$$\lambda_{x_{kc_1}} = \lambda_{x_{kc_2}} = \lambda_{x_{kc_3}} = \beta_k - \alpha.$$

[5]Vale aclarar que, dado que β_v depende de v, estos pueden ser distintos para cada vértice.

De esta manera, teniendo en cuenta además la condición (3.34), se puede ver que λ se puede formar como la combinación lineal

$$\sum_{v \in V} \beta_v e^{(v)} - \alpha\pi,$$

donde $e^{(v)}$ representa el vector de coeficientes de la ecuación de (2.1) correspondiente al vértice v. Por lo tanto, las *3-colors inner clique inequalitites* definen facetas de $\mathcal{PS}(G, C)$. □

Un aspecto interesante de esta familia de desigualdades son las desigualdades que surgen al utilizar $K = \{v, w\}$ para alguna arista $vw \in E$, y $k = v$. La desigualdad en este caso es

$$z_{vw} \geq (x_{vc1} + x_{vc2} + x_{vc3}) + x_{wc2} - 1,$$

y puede verse que la misma domina a las desigualdades (2.3) del modelo. Análogamente, si se utiliza $k = w$ se obtiene otra desigualdad también dominante para (2.3).

3.2.4. 3-colors outer clique (3COK) inequality

Esta desigualdad es muy similar a la anterior pero, en esta ocasión, se parte de suponer que el color asignado al vértice distinguido k es c_2, dejando los colores *externos* de Q para pintar otros vértices de la clique.

Consideremos entonces nuevamente el conjunto de vértices que definen una clique K, el vértice distinguido $k \in K$ y el conjunto de tres colores consecutivos $Q = \{c_1, c_2, c_3\} \subseteq C$. Supongamos ahora que se utiliza el c_2 para pintar el vértice k, entonces si se asigna c_1 a algún vértice $v \in K$ se obtendrá una adyacencia de colores dada por la variable z_{vk}. Lo mismo pasa si se utiliza el color c_3, en lugar de c_1, para pintar v.
Es preciso notar que ambos colores (c_1 y c_3) pueden estar utilizados al mismo tiempo sobre dos vértices distintos $v, w \in K \setminus \{k\}$, generando así dos adyacencias. De hecho, la cantidad de adyacencias entre k y otros vértices de la clique estará dada por la cantidad de estos últimos que utilicen los colores c_1 y c_3. Es decir

$$x_{kc_2} = 1 \quad \Rightarrow \sum_{v \in K \setminus \{k\}} z_{vk} = \sum_{v \in K \setminus \{k\}} (x_{vc_1} + x_{vc_3}),$$

pero la cantidad de vértices de $K \setminus \{k\}$ usando c_1 o c_3 es obviamente menor o igual que 2.

Se puede escribir esta implicación como una desigualdad lineal de la siguiente manera:

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \; 2x_{kc_2} + \sum_{v \in K \setminus \{k\}} (x_{vc_1} + x_{vc_3}) \; - \; 2. \tag{3.44}$$

Nuevamente utilizamos aquí un procedimiento de *lifting* con el objetivo de fortalecer esta desigualdad válida, en este caso para introducir las variables x_{kc_1} y x_{kc_3} en (3.44), y obtener la siguiente definición.

Definición 3.2.4. *Sea $K \subseteq V$ un conjunto de vértices que inducen un subgrafo completo de G, $k \in K$ un vértice distinguido y $Q = \{c_1, c_2, c_3\} \subseteq C$ un conjunto de tres colores consecutivos tal que $c_{i+1} = c_i + 1$, para $i = 1, 2$. Definimos la* 3-colors outer clique inequality *asociada a K, k y Q como*

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq (x_{kc_1} + 2x_{kc_2} + x_{kc_3}) + \sum_{v \in K \setminus \{k\}} (x_{vc_1} + x_{vc_3}) \quad - \quad 2. \tag{3.45}$$

Proposición 3.2.4. *Las* 3-colors outer clique inequalities *son válidas para $\mathcal{PS}(G, C)$.*

Demostración. Es fácil ver que si no se asigna ningún color de Q al vértice k, entonces la desigualdad se cumple trivialmente ya que, por un lado $(x_{kc_1} + 2x_{kc_2} + x_{kc_3}) = 0$ y por otro, como K representa una clique, dados dos colores distintos, se puede pintar a lo sumo dos vértices de la clique con ellos, es decir,

$$\sum_{v \in K \setminus \{k\}} (x_{vc_1} + x_{vc_3}) \leq 2,$$

lo cual deja un valor nulo o negativo en el lado derecho de (3.45). Por este motivo basta demostrar que la desigualdad es válida cuando al vértice k se le asigna algún color de Q.
Veamos entonces qué sucede si k se pinta con el color c_1. En este caso ocurre $(x_{kc_1} + 2x_{kc_2} + x_{kc_3}) = 1$, y por lo tanto deberíamos probar que

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} (x_{vc_1} + x_{vc_3}) \quad - \quad 1.$$

Pero dado que k pertenece a la clique, el hecho de estar pintado con c_1 impide que otros vértices de la clique estén pintados con el mismo color, con lo cual sólo deberíamos demostrar que

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} x_{vc_3} \quad - \quad 1, \tag{3.46}$$

pero esto es trivial, pues el lado derecho de (3.46) es a lo sumo cero. Analogamente, se demuestra la validez si el color asignado a k es c_3.

Si en cambio el color asignado al vértice k fuese c_2, tendríamos entonces $(x_{kc_1} + 2x_{kc_2} + x_{kc_3}) = 2$, y por lo tanto habría que demostrar que

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} (x_{vc_1} + x_{vc_3}). \tag{3.47}$$

Pero en esta ocasión las únicas adyacencias de color que puede tener k con otros vértices de la clique estarán dadas únicamente con los vértices que reciban colores adyacentes a c_2, es decir c_1 y c_3, y por lo tanto por cada vértice de la clique que reciba uno de esos colores la correspondiente variable de adyacencia con respecto a k deberá valer 1. □

Es importante notar, al igual que para las *3CIK inequalities*, que tanto c_1 como c_3 pueden no existir y la desigualdad seguirá siendo válida. Es decir, si se toman sólo los dos primeros colores de C o sólo los dos últimos y se escribe la desigualdad sin las variables correspondientes al color faltante, la misma seguirá siendo válida. La validez de estos hechos queda implicada nuevamente por la dominancia de (3.45) sobre la misma desigualdad con estas variables ausentes.

Teorema 3.2.3. *Si* $|C| > \chi(G)$ *y* $|C| \geq |K| + 5$*, entonces las* 3-colors outer clique inequalities *definen facetas de* $\mathcal{PS}(G, C)$*.*

La demostración de este teorema tiene sólo leves diferencias con la del teorema 3.2.2.

3.2.5. 4-colors vertex clique (4CVK) inequality

Esta desigualdad surge de analizar un caso similar a los dos anteriores. Nuevamente tenemos una clique K y un vértice distinguido $k \in K$, aunque en este caso comenzamos considerando sólo dos colores consecutivos, llamemos a estos colores c_2 y c_3[6].
Supongamos que asignamos al vértice k, el color c_2. Luego, si cualquier vértice de la clique recibe el color c_3, se obtendrá una adyacencia de color entre estos dos vértices. Analogamente ocurre lo mismo si el color asignado a k es c_3 y el asignado al otro vértice de la clique es c_2. Es decir

$$(x_{kc_2} + x_{kc_3}) = 1 \quad \wedge \quad \sum_{v \in K \setminus \{k\}} (x_{vc_2} + x_{vc_3}) = 1 \quad \Rightarrow \sum_{v \in K \setminus \{k\}} z_{vk} \geq 1.$$

Podemos representar esta situación por medio de una desigualdad lineal como

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \quad (2x_{kc_2} + 2x_{kc_3}) \quad + \sum_{v \in K \setminus \{k\}} (x_{vc_2} + x_{vc_3}) \quad - \quad 2. \tag{3.48}$$

Si llamamos c_1 y c_4 a los colores inmediatamente anterior y posterior a c_2 y c_3, respectivamente, es posible utilizar un proceso de *lifting* para incorporar las variables x_{kc_1} y x_{kc_4} a la desigualdad y obtener con eso una desigualdad válida más fuerte. La misma se define a continuación.

Definición 3.2.5. *Sea* $K \subseteq V$ *un conjunto de vértices que inducen un subgrafo completo de* G*,* $k \in K$ *un vértice distinguido y* $Q = \{c_1, c_2, c_3, c_4\} \subseteq C$ *un*

[6]Esta nomenclatura cobra sentido una vez que la desigualdad quede totalmente definida.

conjunto de cuatro colores consecutivos tal que $c_{i+1} = c_i + 1$*, para* $i = 1, 2, 3$*. Definimos la* 4-colors vertex clique inequality *asociada a* K*,* k *y* Q *como*

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq (x_{kc_1} + 2x_{kc_2} + 2x_{kc_3} + x_{kc_4}) + \sum_{v \in K \setminus \{k\}} (x_{vc_2} + x_{vc_3}) \ - \ 2. \quad (3.49)$$

Proposición 3.2.5. *Las* 4-colors vertex clique inequalities *son válidas para* $\mathcal{PS}(G, C)$*.*

Demostración. En el caso en que el vértice k no utiliza ningún color de Q, es sencillo probar la validez de (3.49) ya que $(x_{kc_1} + 2x_{kc_2} + 2x_{kc_3} + x_{kc_4}) = 0$ y, como K representa una clique, dados dos colores distintos, se puede pintar a lo sumo dos vértices de la clique con ellos, es decir,

$$\sum_{v \in K \setminus \{k\}} (x_{vc_2} + x_{vc_3}) \leq 2,$$

y por ello el lado derecho de (3.49) resulta nulo o negativo. Luego, basta demostrar que la desigualdad es válida cuando al vértice k se le asigna algún color de Q.

Veamos entonces qué sucede si k se pinta con el color c_1. En este caso ocurre $(x_{kc_1} + 2x_{kc_2} + 2x_{kc_3} + x_{kc_4}) = 1$, y por lo tanto deberíamos probar que

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} (x_{vc_2} + x_{vc_3}) \ - \ 1. \quad (3.50)$$

Dado que k está pintado con c_1, sabemos que cualquier vértice de la clique que utilice c_2 generará una adyacencia de color con k, es decir,

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} x_{vc_2}, \quad (3.51)$$

y además, al ser K una clique sabemos también que

$$\sum_{v \in K \setminus \{k\}} x_{vc_3} \leq 1. \quad (3.52)$$

Por lo tanto, usando (3.51) y (3.52) se demuestra (3.50). Analogamente, se prueba la validez de (3.49) si el color utilizado por k es c_4, en lugar de c_1.

Restaría ver entonces qué ocurre si el color asigando a k es c_2 o bien c_3. Supongamos entonces que se asignó c_2 al vértice k. En este caso se puede ver que $(x_{kc_1} + 2x_{kc_2} + 2x_{kc_3} + x_{kc_4}) = 2$, y por lo tanto lo que hay que probar es que

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} (x_{vc_2} + x_{vc_3}), \quad (3.53)$$

aunque en realidad sabemos que ningún otro vértice de la clique, además de k, puede recibir el color c_2, con lo cual (3.53) se reduce a

$$\sum_{v \in K \setminus \{k\}} z_{vk} \geq \sum_{v \in K \setminus \{k\}} x_{vc_3}, \tag{3.54}$$

y es sencillo ver que cualquier vértice de la clique que reciba el color c_3 generará una adyacencia de color con el vértice k (pues este último está pintado con c_2) lo que demuestra (3.54). Analogamente, se prueba la validez en el caso en que k reciba el color c_3 en lugar de c_2. □

Al igual que en las últimas dos desigualdades válidas, vale aclarar que los colores de los extremos de Q pueden no existir, pudiendo utilizarse entonces los tres primeros colores de C para representar c_2, c_3 y c_4, o bien los tres últimos colores de C para c_1, c_2 y c_3. La validez, al igual que antes, queda implicada por la dominancia de (3.49) sobre la misma desigualdad con estas variables ausentes.

Teorema 3.2.4. *Si* $|C| > \chi(G)$ *y* $|C| \geq |K| + 5$*, entonces las* 4-colors vertex clique inequalities *definen facetas de* $\mathcal{PS}(G, C)$*.*

Demostración. Sea $\pi \in \mathbb{R}^{nt+m}$ el vector de coeficientes de (3.49) y $\pi_0 \in \mathbb{R}$ el término independiente de la misma. La cara de $\mathcal{PS}(G, C)$ definida por esta desigualdad es entonces

$$F = \{y \in \mathbb{R}^{nt+m} : \pi y = \pi_0\} \cap \mathcal{PS}(G, C).$$

Sean $\lambda \in \mathbb{R}^{nt+m}$ y $\lambda_0 \in \mathbb{R}$ tales que $\lambda y = \lambda_0, \forall y \in F$. Veremos que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo, con lo cual F es una faceta de $\mathcal{PS}(G, C)$.

Antes de comenzar la demostración caracterizaremos las soluciones enteras $y = (x, z) \in F$ en tres tipos:

Tipo (I) El vértice k no utiliza ningún color de Q y existen dos vértices de la clique que utilizan c_2 y c_3. Además, ningún vértice de la clique utiliza un color adyacente al utilizado por k. Es decir,

- $x_{kc_1} = x_{kc_2} = x_{kc_3} = x_{kc_4} = 0$,
- $x_{vc_2} = 1$ y $x_{wc_3} = 1$ para algunos $v, w \in K \setminus \{k\}$ y
- $z_{kv} = 0$ para todo $v \in K \setminus \{k\}$.

Tipo (II) El vértice k utiliza c_1 o c_4 y al menos algún otro vértice de la clique utiliza c_3 o c_2, respectivamente. Además, ningún vértice de la clique utiliza el color adyacente al utilizado por k que está fuera de Q (es decir $c_1 - 1$ o c_4+1, respectivamente). Vale aclarar que el resto de los colores pueden o no ser usados por vértices de la clique y la desigualdad seguirá cumpliéndose por igualdad. Llamaremos tipo (IIa) al caso en que k utiliza c_1 y (IIb) que utiliza c_4. Es decir, en las de tipo (IIa)

- $x_{kc_1} = 1$,
- $x_{vc_3} = 1$ para algún $v \in K \setminus \{k\}$ y
- si existe $c_1 - 1$, entonces $x_{vc_1-1} = 0$ para todo $v \in K \setminus \{k\}$.

y en las de tipo (IIb)

- $x_{kc_4} = 1$,
- $x_{vc_2} = 1$ para algún $v \in K \setminus \{k\}$ y
- si existe $c_4 + 1$, entonces $x_{vc_4+1} = 0$ para todo $v \in K \setminus \{k\}$.

Tipo (III) El vértice k utiliza c_2 o c_3 y ningún otro vértice de la clique utiliza el c_1 o c_4, respectivamente. Aclaramos también en este caso que el resto de los colores pueden o no ser usados por vértices de la clique y la desigualdad seguirá cumpliéndose por igualdad. Llamaremos tipo (IIIa) al caso en que k utiliza c_2 y (IIIb) que utiliza c_3. Es decir, en las de tipo (IIIa)

- $x_{kc_2} = 1$ y
- $x_{vc_1} = 0$ para todo $v \in K \setminus \{k\}$.

y en las de tipo (IIIb)

- $x_{kc_3} = 1$ y
- $x_{vc_4} = 0$ para todo $v \in K \setminus \{k\}$.

A continuación se muestran gráficamente los tipos de soluciones descriptos. Los espacios en blanco de la tabla indican que no hay restricciones para los vértices que pueden recibir el color en cuestión y el símbolo $\emptyset_K$ indica que el mismo no está asignado a ningún vértice de la clique. Tanto v como w representan colores de la clique.

		c		c_1	c_2	c_3	c_4	
$(I) \rightarrow$	$\emptyset_K$	k	$\emptyset_K$		v	w		
$(IIa) \rightarrow$			$\emptyset_K$	k		v		
$(IIb) \rightarrow$					v		k	$\emptyset_K$
$(IIIa) \rightarrow$				$\emptyset_K$	k			
$(IIIb) \rightarrow$						k	$\emptyset_K$	

Para demostrar que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo veremos que los coeficientes de λ cumplen con ciertas condiciones, las cuales se van demostrando a continuación.

Dada una arista $vw \in E$, tal que $vw \notin \{kj : j \in K\}$, veremos que $\lambda_{z_{vw}} = 0$. Tomemos una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (I) en la cual los vértices v y w reciban colores no adyacentes y $z_{vw} = 0$. Esta solución puede construirse permutando el color asignado a uno de estos vértices para lograr la no adyacencia. Luego, construimos la solución $y' = (x', z') \in \mathcal{PS}(G, C)$ a partir de la

solución y pero poniendo $z'_{vw} = 1$. Esta solución sigue siendo válida, y dado que z_{vw} no participa de (3.49), la misma sigue cumpliéndose por igualdad.
Dado que tanto y como y' pertenecen a F, sabemos que $\lambda y = \lambda_0$ y $\lambda y' = \lambda_0$ y por lo tanto $\lambda y = \lambda y'$. Sabemos además que y coincide con y' en todos sus elementos salvo en z_{vw}, con lo cual se deduce que

$$\lambda_{z_{vw}} z_{vw} = \lambda_{z_{vw}} z'_{vw},$$

y por lo tanto

$$\textit{[Condición 1]} \qquad \lambda_{z_{vw}} = 0 \quad \forall vw \in E \text{ tal que } vw \notin \{kj : j \in K\}. \tag{3.55}$$

Tomemos ahora un vértice $v \notin K$. Veremos que $\lambda_{x_{vc}} = \lambda_{x_{vc'}}$ para cualquier par de colores $c, c' \in C$. Dado que $|C| > \chi(G)$, es posible construir una solución en la cual el color c' no esté utilizado por ningún vértice del grafo. Consideremos entonces una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (I) en la cual c' esté libre de vértices[7] y en la cual el vértice v reciba el color c (esto último puede conseguirse permutando el color de v, asumiendo, sin perder generalidad, que el color asignado a v no es ni c_2 ni c_3). Luego, construimos una solución $y' = (x', z')$ a partir de y, en la cual el vértice v recibe el color c'. Podemos visualizar la diferencia entre estas dos soluciones con el siguiente diagrama:

		c		c'	
$y \rightarrow$		v			
$y' \rightarrow$				v	

Esta solución es válida pues este color no estaba ultilizado por ningún otro vértice del grafo, y además esta solución pertenece a la cara F pues es también una solución de tipo (I).
Dado que tanto y como y' pertenecen a F, sabemos que $\lambda y = \lambda_0$ y $\lambda y' = \lambda_0$ y por lo tanto $\lambda y = \lambda y'$. Sabemos además que y coincide con y' en todos sus elementos salvo en x_{vc}, $x_{vc'}$ y en algunas variables de adyacencia incidentes a v. Sin embargo, por (3.55) sabemos que $\lambda_{z_{vw}} = 0, \forall vw \in E$, y entonces podemos deducir que

$$\lambda_{x_{vc}} x_{vc} + \lambda_{x_{vc'}} x_{vc'} = \lambda_{x_{vc}} x'_{vc} + \lambda_{x_{vc'}} x'_{vc'},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\lambda_{x_{vc}} = \lambda_{x_{vc'}}.$$

Para el caso en que $c' = c_2$ o $c' = c_3$, no existen en F soluciones de tipo (I) de manera tal que no se utilice c', sin embargo, se puede demostrar esta igualdad

[7] Para que exista esta solución, asumimos que $c' \notin \{c_2, c_3\}$; luego veremos estos casos particulares.

bajo el mismo razonamiento previo utilizando soluciones de tipo (IIIb) y (IIIa), respectivamente. Así,

$$\textit{[Condición 2]} \qquad \lambda_{x_{vc}} = \lambda_{x_{vc'}} \quad \forall v \notin K \quad \forall c, c' \in C. \tag{3.56}$$

A continuación, consideremos un vértice $v \in K \setminus \{k\}$. Veremos que $\lambda_{x_{vc_2}} = \lambda_{x_{vc_3}} = \lambda_{x_{vc}} - \lambda_{z_{kv}}$ para algún color $c \in C \setminus \{c_2, c_3\}$. Tomemos en este caso una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (IIIa) en la cual el vértice v utilice el color c_3, y además haya un color $c \in C \setminus \{c_2, c_3\}$ libre de vértices (esto puede construirse gracias a que $|C| > \chi(G)$), el cual asumimos por ahora que es distinto de c_1[8]. Luego, construimos una segunda solución $y' = (x', z')$ a partir de y pero pintando en este caso el vértice v con el color c.

	c		c_1	c_2	c_3	c_4	
$y \rightarrow$	$\emptyset$		$\emptyset_K$	k	v		
$y' \rightarrow$	v		$\emptyset_K$	k			

Esta solución sigue siendo válida ya que c no estaba utilizado en ningún vértice. Además, esta solución sigue siendo una solución de tipo (IIIb) con lo cual pertenece a la cara F y por lo tanto, al igual que en casos anteriores, se puede ver que $\lambda y = \lambda y'$. En este caso, y coincide con y' en todas sus variables salvo en x_{vc_3}, x_{vc}, z_{kv} y otras variables de adyacencia para v. Estas últimas variables sin embargo quedarán anuladas pues, por (3.55), los coeficientes de λ para esas variables son nulos. Se puede decir entonces que

$$\lambda_{x_{vc_3}} x_{vc_3} + \lambda_{x_{vc}} x_{vc} + \lambda_{z_{kv}} z_{kv} = \lambda_{x_{vc_3}} x'_{vc_3} + \lambda_{x_{vc}} x'_{vc} + \lambda_{z_{kv}} z'_{kv},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\lambda_{x_{vc_3}} + \lambda_{z_{kv}} = \lambda_{x_{vc}} \tag{3.57}$$

Análogamente, utilizando una solución de tipo (IIIb) en lugar de (IIIa) y partiendo de una solución en la cual el vértice v utilice el color c_2, se demuestra, asumiendo que $c \neq c_4$, que

$$\lambda_{x_{vc_2}} + \lambda_{z_{kv}} = \lambda_{x_{vc}} \tag{3.58}$$

y por lo tanto, de (3.57) y (3.58) decimos que

$$\textit{[Condición 3]} \qquad \lambda_{x_{vc_2}} = \lambda_{x_{vc_3}} = \lambda_{x_{vc}} - \lambda_{z_{kv}} \quad \forall v \in K \setminus \{k\}, \text{ para algún } c \in C \setminus \{c_2, c_3\}. \tag{3.59}$$

Vale aclarar que el color c puede ser, permutando los colores, cualquier color

[8] Veremos más adelante cómo puede demostrarse lo mismo para este caso.

salvo c_2 y c_3 (para los casos obviados en los que $c = c_1$ y $c = c_4$, se puede realizar la demostración análogamente utilizando soluciones de tipo (II) en lugar de tipo (III)). Con esto, se puede ver que

$$\textit{[Condición 4]} \qquad \lambda_{x_{vc}} = \lambda_{x_{vc'}} \quad \forall v \in K \setminus \{k\} \quad \forall c, c' \in C \setminus \{c_2, c_3\}. \tag{3.60}$$

Veremos a continuación que $\lambda_{x_{kc_1}} = \lambda_{x_{kc_4}} = \lambda_{x_{kc}} - \lambda_{z_{kv}}$ para algún $c \in C \setminus Q$ y algún $v \in K \setminus \{k\}$. Vemos primero lo que ocurre con $\lambda_{x_{kc1}}$.
Para ello tomemos una solución $y = (x, z) \in \mathcal{PS}(G, C)$ de tipo (IIa) en la cual dos vértices $v, w \in K \setminus \{k\}$ utilicen los colores c_2 y c_3, respectivamente, y además haya un color $c \in C \setminus Q$ libre de vértices (esto puede construirse gracias a que $|C| > \chi(G)$). Pedimos también que los dos colores adyacentes a c no estén asignados a ningún vértice de K; esto es factible si $|C \setminus \{c\}| \geq |K| + 3$, y esto último está garantizado por las hipótesis del teorema. Luego, construimos una segunda solución $y' = (x', z')$ a partir de y pero pintando en este caso el vértice k con el color c.

			c			c_1	c_2	c_3	c_4	
$y \rightarrow$		$\emptyset_K$	$\emptyset$	$\emptyset_K$		$\emptyset_K$	k	v	w		
$y' \rightarrow$		$\emptyset_K$	k	$\emptyset_K$		$\emptyset_K$		v	w		

Esta solución sigue siendo válida ya que c no estaba utilizado en ningún vértice. Además, esta solución pertenece a la cara F ya que representa una solución de tipo (I) y por lo tanto, al igual que en casos anteriores, se puede ver que $\lambda y = \lambda y'$. En este caso, y coincide con y' en todas sus variables salvo en x_{kc_1}, x_{kc}, z_{kv} y eventualmente algunas otras variables de adyacencia para k, aunque estas últimas variables quedarán anuladas pues, por (3.55), los coeficientes de λ para esas variables son nulos. Se puede decir entonces que

$$\lambda_{x_{kc_1}} x_{kc_1} + \lambda_{x_{kc}} x_{kc} + \lambda_{z_{kv}} z_{kv} = \lambda_{x_{kc_1}} x'_{kc_1} + \lambda_{x_{kc}} x'_{kc} + \lambda_{z_{kv}} z'_{kv},$$

y reemplazando las variables por los valores que toman en ambas soluciones se desprende que

$$\lambda_{x_{kc_1}} + \lambda_{z_{kv}} = \lambda_{x_{kc}}, \tag{3.61}$$

para algún $v \in K \setminus \{k\}$ y algún $c \in C \setminus Q$.
Análogamente, partiendo de una solución de tipo (IIb) en lugar de una de tipo (IIa), en la cual el vértice v utilice en este caso el color c_3, se demuestra lo mismo para $\lambda_{x_{kc_4}}$. Es decir,

$$\lambda_{x_{kc_4}} + \lambda_{z_{kv}} = \lambda_{x_{kc}}, \tag{3.62}$$

para algún $v \in K \setminus \{k\}$ y algún $c \in C \setminus Q$. Finalmente, combinando (3.61) y (3.62) llegamos a

$$\textit{[Condición 5]} \quad \lambda_{x_{kc_1}} = \lambda_{x_{kc_4}} = \lambda_{x_{kc}} - \lambda_{z_{kv}}, \tag{3.63}$$
$$\text{para algún } v \in K \setminus \{k\}, \text{ y algún } c \in C \setminus Q.$$

Vale aclarar que el vértice v se tomó de manera arbitraria y puede usarse cualquier vértice $v \in K \setminus \{k\}$, con lo cual utilizando (3.41) variando el vértice elegido se deduce que

$$\textit{[Condición 6]} \quad \lambda_{z_{kv}} = \lambda_{z_{kw}} \quad \forall v, w \in K \setminus \{k\}. \tag{3.64}$$

Demostraremos ahora que $\lambda_{x_{kc}} = \lambda_{x_{kc'}}$ para todo $c, c' \in C \setminus Q$. Tomemos una solución $y = (x, z)$ de tipo (I) en la cual el vértice k está pintado con el color c; es factible construir esta solución permutando los colores asignados. Además, el color c' debe estar libre de vértices; nuevamente la hipótesis $|C| > \chi(G)$ permite construir esta solución. Y por último, pediremos que en esta solución tanto los colores adyacentes a c como los adyacentes a c' no estén asignados a ningún vértice de K; esto es factible si $|C \setminus \{c\}| \geq |K| + 4$, lo cual está implicado por las hipótesis. Luego, construimos una solución $y' = (x', z')$ a partir de y en la cual el vértice k utiliza el color c'.

			c				c'		
$y \rightarrow$		$\emptyset_K$	k	$\emptyset_K$		$\emptyset_K$	$\emptyset$	$\emptyset_K$	
$y' \rightarrow$		$\emptyset_K$		$\emptyset_K$		$\emptyset_K$	k	$\emptyset_K$	

Esta solución es una solución válida ya que el color c' no estaba utilizado por ningún vértice. Además, esta solución pertenece a la cara F ya que, al igual que y, es una solución de tipo (I).
Estas dos soluciones coinciden en todos sus valores salvo en los valores de x_{kc}, $x_{kc'}$ y eventualmente algunas variables de adyacencia entre k y vértices fuera de la clique (pues aseguramos que los colores adyacentes a c y a c' no están asignados a vértices de K). Estas últimas variables resultan anuladas por sus correspondientes coeficientes en λ, con lo cual se puede ver que

$$\lambda_{x_{kc}} x_{kc} + \lambda_{x_{kc'}} x_{kc'} = \lambda_{x_{kc}} x'_{kc} + \lambda_{x_{kc'}} x'_{kc'},$$

y reemplazando las variables por los valores que toman en ambas soluciones se obtiene

$$\textit{[Condición 7]} \quad \lambda_{x_{kc}} = \lambda_{x_{kc'}} \quad \forall c, c' \in C \setminus Q. \tag{3.65}$$

La última condición a demostrar es el hecho de que $\lambda_{x_{kc_2}} = \lambda_{x_{kc_3}} = \lambda_{x_{kc_1}} - \lambda_{z_{kv}}$ para algún $v \in K \setminus \{k\}$. Tomemos entonces una solución $y = (x, z)$ de tipo (IIIa) en la cual algún vértice $v \in K \setminus \{k\}$ esté pintado con el color c_3; es factible construir esta solución permutando los colores asignados. Además, el color c_1 debe estar libre de vértices; nuevamente la hipótesis $|C| > \chi(G)$ permite tomar esta solución. Por último, pediremos que en esta solución el color

anterior a c_1, de existir, no esté asignado a ningún vértice de K; esto es factible si $|C \setminus \{c_1 - 1\}| \geq |K| + 1$, lo cual está implicado por las hipótesis. Luego, construimos una solución $y' = (x', z')$ a partir de y en la cual el vértice k utiliza el color c_1.

			c_1	c_2	c_3	c_4	
$y \rightarrow$		$\emptyset_K$	$\emptyset$	k	v		
$y' \rightarrow$		$\emptyset_K$	k		v		

Esta solución es una solución válida ya que el color c_1 no estaba utilizado por ningún vértice. Además, esta solución pertenece a la cara F ya que corresponde a una solución de tipo (IIa).
Estas dos soluciones coinciden en todos sus valores salvo en los valores de x_{kc_2}, x_{kc_1}, z_{kv} y eventualmente algunas variables de adyacencia entre k y vértices fuera de la clique. Estas últimas variables resultan anuladas por sus correspondientes coeficientes en λ, con lo cual se puede ver que

$$\lambda_{x_{kc_1}} x_{kc_1} + \lambda_{x_{kc_2}} x_{kc_2} + \lambda_{z_{kv}} z_{kv} = \lambda_{x_{kc_1}} x'_{kc_1} + \lambda_{x_{kc_2}} x'_{kc_2} + \lambda_{z_{kv}} z'_{kv},$$

y reemplazando las variables por los valores que toman en ambas soluciones se obtiene

$$\lambda_{x_{kc_1}} = \lambda_{x_{kc_2}} + \lambda_{z_{kv}}. \tag{3.66}$$

Análogamente, utilizando soluciones de tipo (IIIb) y (IIb), respectivamente, se puede demostrar que

$$\lambda_{x_{kc_4}} = \lambda_{x_{kc_3}} + \lambda_{z_{kv}}, \tag{3.67}$$

y dado que, por (3.63), $\lambda_{x_{kc_1}} = \lambda_{x_{kc_4}}$ se demuestra que

$$\textit{[Condición 8]} \quad \lambda_{x_{kc_2}} = \lambda_{x_{kc_3}} = \lambda_{x_{kc_1}} - \lambda_{z_{kv}}, \text{ para algún } v \in K \setminus \{k\}. \tag{3.68}$$

Terminamos con esto de demostrar las condiciones particulares sobre los coeficientes de λ. Veremos ahora cómo estas condiciones implican el hecho de que λ es una combinación lineal de π y los vectores de coeficientes de las ecuaciones (2.1) del modelo.
La condición (3.56) indica que dado un vértice v fuera de la clique, el coeficiente de λ para la variable x_{vc} es el mismo para cualquier color c. Llamemos entonces, β_v a este valor. Es decir, dado $v \notin K$

$$\lambda_{x_{vc}} = \beta_v \quad \forall c \in C.$$

Por otro lado, la condición (3.60) indica que dado un vértice $v \in K \setminus \{k\}$, el coeficiente de λ para la variable x_{vc} es el mismo para cualquier color c, salvo

para c_2 y c_3. Llamemos entonces, β_v a este valor[9], o sea

$$\lambda_{x_{vc}} = \beta_v \quad \forall c \in C \setminus \{c_2, c_3\}.$$

De manera similar, la condición (3.65) indica que el coeficiente para la variable x_{kc} es el mismo para cualquier color c, salvo para c_1, c_2, c_3 y c_4. Llamemos entonces, β_k a este valor, es decir

$$\lambda_{x_{kc}} = \beta_k \quad \forall c \in C \setminus \{c_1, c_2, c_3, c_4\}.$$

En la condición (3.64) se ve que $\lambda_{z_{kv}} = \lambda_{z_{kw}}$ para todo par de vértices $v, w \in K \setminus \{k\}$. Si llamamos α a este valor obtenemos que

$$\lambda_{z_{kv}} = \alpha \quad \forall v \in K \setminus \{k\}.$$

Definidos α y β_v para cada $v \in V$, podemos ver qué valores toman el resto de los coeficientes de λ en función de éstos.
Tomando la condición (3.59), podemos ver que

$$\lambda_{x_{vc_2}} = \lambda_{x_{vc_3}} = \beta_v - \alpha \quad \forall v \in K \setminus \{k\}.$$

Para el vértice k, la condición (3.63) implica que

$$\lambda_{x_{kc_1}} = \lambda_{x_{kc_4}} = \beta_k - \alpha,$$

y entonces según la condición (3.68)

$$\lambda_{x_{kc_2}} = \lambda_{x_{kc_3}} = \beta_k - 2\alpha.$$

De esta manera, teniendo en cuenta además la condición (3.55), se puede ver que λ se puede formar como la combinación lineal

$$\sum_{v \in V} \beta_v e^{(v)} - \alpha\pi,$$

donde $e^{(v)}$ representa el vector de coeficientes de la ecuación de (2.1) correspondiente al vértice v. Por lo tanto, las *4-colors vertex clique inequalitites* definen facetas de $\mathcal{PS}(G, C)$. □

[9]Vale aclarar que, dado que β_v depende de v, estos pueden ser distintos para cada vértice.

CAPÍTULO 4

Branch & Cut

En este capítulo se presenta la implementación de un algoritmo *Branch & Cut* para el *minimum-adjacency vertex coloring problem*. El algoritmo utiliza como planos de corte las desigualdades válidas descriptas en el Capítulo 3 de este trabajo e introduce además una desigualdad válida adicional ya existente en la bibliografía.

En primera instancia se comentan los procedimientos de separación implementados para la búsqueda de cortes para las soluciones fraccionarias en la fase de planos de corte. Para cada familia de desigualdades se implementaron dos tipos de procedimientos para la separación: uno de ellos es un algoritmo de tipo *bactracking* mientras que el otro es una heurística golosa. Ambos procedimientos son ampliamente parametrizables, con la idea de que su desempeño pueda ser evaluado y comparado en la etapa de experimentación. Los resultados de esta experimentación son comentados en el Capítulo 5 de este trabajo.

Luego de describir los procedimientos de separación se comentarán otras técnicas también utilizadas en el *Branch & Cut*. Estas técnicas incluyen la obtención de cotas primales por medio de redondeos de las soluciones, la reducción de los subproblemas mediante la fijación de variables por implicaciones lógicas y la selección de la variable de *branching*.

4.1. *Clique inequalities*

Presentamos a continuación una desigualdad válida de la literatura que fue utilizada en el algoritmo *Branch & Cut* del presente trabajo. Esta desigualdad es una adaptación de la *clique inequality* presentada en [15], ya que esta última está definida para el poliedro asociado al *stable model* para el problema clásico de coloreo y utiliza variables que no existen en el modelo adaptado al *minimum-adjacency vertex coloring problem*. En el modelo orignal, la variable binaria w_c

indica si el color c es utilizado en el coloreo o no. La adaptación de las *clique inequalities* consiste en reemplazar esas variables por 1.

La desigualdad surge de la idea de que dada una clique de G y un color de C, a lo sumo un sólo vértice de la clique puede utilizar este color, ya que de haber más de un vértice de la clique asignado a este color, el coloreo no sería válido. Damos a continuación la definición formal de la misma.

Definición 4.1.1. *Sea $K \subseteq V$ un conjunto de vértices que inducen una clique de G, y sea $c \in C$ un color cualquiera. Se define la* clique inequality *asociada a K y c como*

$$\sum_{v \in K} x_{vc} \leq 1. \tag{4.1}$$

Se demuestra en [15] que, si la clique es maximal, la desigualdad define facetas del poliedro asociado al problema clásico de coloreo. Dado que (4.1) no involucra a las variables de adyacencia, conjeturamos que bajo hipótesis similares, también define facetas de $\mathcal{PS}(G, C)$, ya que la demostración de esto parece ser similar a la presentada en [15]. Sin embargo, las hipótesis necesarias para esto deben ser verificadas con mayor cuidado, y por lo tanto no se enuncia, en este trabajo, el teorema de facetitud de (4.1) sobre $\mathcal{PS}(G, C)$.

4.2. Algoritmos de separación

Una de las etapas más importantes en un algoritmo de tipo *Branch & Cut* es la etapa de separación. Luego de la resolución de cada relajación lineal, se obtiene una solución posiblemente fraccionaria y por lo tanto no la solución óptima. Se comineza entonces con la fase de planos de corte.
En esta etapa se buscan, dentro de una o más familias de desigualdades válidas, uno o más planos de corte que sirvan para "cortar" esta solución fraccionaria hallada, es decir, desigualdades válidas para $\mathcal{PS}(G, C)$ que no sean cumplidas por la solución en cuestión. Estos planos de corte se agregan a la formulación actual del modelo como restricciones de la misma y de esta manera la solución encontrada previamente deja de ser factible para este nuevo poliedro.
Dado que las desigualdades son válidas para $\mathcal{PS}(G, C)$ (y por lo tanto para toda solución entera dentro del poliedro), no se corre riesgo de estar perdiendo el óptimo entero del problema.

Veremos en esta sección los procedimientos de separación implementados para la búsqueda de planos de corte utilizando las siguientes familias de desigualdades válidas:

- 3-colors inner clique inequalities (3CIK),
- 3-colors outer clique inequalities (3COK),
- 4-colors vertex clique inequalities (4CVK),

- multi-consecutive colors clique inequalities (MCCK) y
- clique inequalities (K).

Todas estas familias requieren una clique del grafo y por este motivo fue posible generalizar los procedimientos para poder utilizar los mismos en varios casos. Para las desigualdades 3CIK, 3COK, 4CVK y K se utiliza la misma implementación variando en cada caso determinados parámetros. En el caso de las desigualdades MCCK, si bien los algoritmos comparten la idea general, fue necesario particularizarlos levemente y por lo tanto esta desigualdad cuenta con un algoritmo distinto del resto. Detallamos estas dos implementaciones en las secciones siguientes.
También, como se ha comentado en la introducción de este capítulo, se han implementado dos procedimientos de distinto tipo en cada caso, siendo el primero un algoritmo de *backtracking* y el segundo una heurística golosa. Estos dos algoritmos se describen en cada una de las secciones siguientes.

4.2.1. Busqueda genérica de cliques

Para explicar la generalización de este procedimiento, se utilizará como ejemplo la desigualdad 3CIK.
Supongamos entonces que $\hat{y} = (\hat{x}, \hat{z})$ es la solución fraccionaria hallada. Por lo tanto, queremos hallar una desigualdad de la familia de las 3CIK que sea violada por esta solución. Es decir, queremos hallar una clique $K \in V$, un vértice $k \in K$ y un conjunto $Q = \{c_1, c_2, c_3\} \subseteq C$ de tres colores consecutivos, tales que

$$\sum_{v \in K \setminus \{k\}} \hat{z}_{vk} < (\hat{x}_{kc_1} + \hat{x}_{kc_2} + \hat{x}_{kc_3}) \quad + \sum_{v \in K \setminus \{k\}} \hat{x}_{vc_2} \quad - \quad 1. \tag{4.2}$$

Ahora bien, la cantidad de posibles vértices $k \in V$ es lineal y por lo tanto no sería un gran problema probar una vez con cada uno de los vértices del grafo. Lo mismo ocurre con la cantidad de conjuntos de tres colores consecutivos de C. Sin embargo, dados $k \in V$, y $Q \subseteq C$, el verdadero problema reside entonces en hallar una clique $K \subseteq \mathcal{N}(k)$ que viole la desigualdad 3CIK asociada a ellos (es decir, que ocurra (4.2)), donde $\mathcal{N}(k)$ representa el conjunto de vértices adyacentes a k, ya que la cantidad de cliques podría ser de un orden exponencial.

Si consideramos un vértice $k \in V$ y un conjunto de tres colores consecutivos $Q = \{c_1, c_2, c_3\} \subseteq C$, podemos definir el *peso* en $\hat{y}$ de un vértice $v \in \mathcal{N}(k)$ con respecto a k como

$$\omega_k(v) = \hat{x}_{vc_2} - \hat{z}_{vk}.$$

Podemos ver que (4.2) puede reescribirse de la siguiente manera

$$\sum_{v \in K \setminus \{k\}} \omega_k(v) > 1 - (\hat{x}_{kc_1} + \hat{x}_{kc_2} + \hat{x}_{kc_3}),$$

y por lo tanto, si hallamos una clique tal que el peso de sus vértices supere el lado derecho de la anterior desigualdad, estaríamos hallando el plano de corte buscado.

De esta manera, podemos escribir el procedimiento de separación para las desigualdades 3CIK como se muestra en el Algoritmo 2. En este algoritmo, γ_k

Algoritmo 2 Hallar plano de corte en la familia *3CIK*

Entrada: $G = (V, E)$, C, $\hat{y} = (\hat{x}, \hat{z})$
1: **para todo** $k \in V$ **hacer**
2: **para todo** $Q = \{c_1, c_2, c_3\} \subseteq C$, colores consecutivos **hacer**
3: Hallar $K \subseteq \mathcal{N}(k)$ tal que

$$\sum_{v \in K} \omega_k(v) > \gamma_k \tag{4.3}$$

4: **fin para**
5: **fin para**

representa el valor a superar por el peso de los vértices de la clique para que la desigualdad sea violada por $\hat{y}$, es decir $\gamma_k = 1 - (\hat{x}_{kc_1} + \hat{x}_{kc_2} + \hat{x}_{kc_3})$.

Como se dijo anteriormente, el punto fundamental de este algoritmo es la búsqueda de la clique que viole la desigualdad, y para ello se implementaron los dos métodos mencionados anteriormente y detallados más adelante, es decir el algoritmo de *backtracking* y la heurística golosa. Lo importante es que estos algoritmos se generalizaron para poder ser utilizados no sólo en esta familia de desigualdades sino también en las demás. Así, parametrizando los valores de la función de pesos ω_k y el valor a superar γ_k, es posible reutilizar este procedimiento para hallar cliques para las demás familias de desigualdades.
Ya describimos estos valores para las *3CIK inequalities*, a continuación se muestra cómo se utiliza esta generalización para las demás familias de desigualdades válidas:

3-colors outer clique inequalities: En este caso, dados $k \in V$ y $Q = \{c_1, c_2, c_3\}$ es sencillo ver que tomando

$$\omega_k(v) = (\hat{x}_{vc_1} + \hat{x}_{vc_3}) - \hat{z}_{kv}$$
$$\gamma_k = 2 - (\hat{x}_{kc_1} + 2\hat{x}_{kc_2} + \hat{x}_{kc_3})$$

debemos hallar, al igual que antes, una clique $K \subseteq \mathcal{N}(k)$ que cumpla (4.3).

4-colors vertex clique inequalities: Para estas desigualdades, dados $k \in V$

y $Q = \{c_1, c_2, c_3, c_4\}$ se debe tomar

$$\omega_k(v) = (\hat{x}_{vc_2} + \hat{x}_{vc_3}) - \hat{z}_{kv}$$
$$\gamma_k = 2 - (\hat{x}_{kc_1} + 2\hat{x}_{kc_2} + 2\hat{x}_{kc_3} + \hat{x}_{kc_4})$$

y así, como antes, se deberá hallar una clique $K \subseteq \mathcal{N}(k)$ que cumpla (4.3).

clique inequalities: Este caso es levemente distinto, ya que esta familia de desigualdades no requiere un vértice $k \in V$ distinguido. Sin embargo, dado que la clique es obviamente no vacía, siempre se puede seleccionar un vértice inicial para la clique, y como k se hará variar para todo vértice del grafo, no se perderá generalidad. Entonces, dado $k \in V$ y $c \in C$ podemos tomar

$$\omega_k(v) = \hat{x}_{vc}$$
$$\gamma_k = 1 - \hat{x}_{kc}$$

y de esta manera el procedimiento general servirá también para esta familia de desigualdades, es decir, se deberá hallar una clique $K \subseteq \mathcal{N}(k)$ que cumpla (4.3).

Si bien cualquier clique que viole la desigualdad servirá como plano de corte para la solución $\hat{y}$, es deseable hallar más de una clique para ello. Así, se puede aprovechar esta etapa para agregar más de un corte a la relajación lineal y de esta manera recortar aun más el poliedro actual.
Con respecto a la calidad de los cortes agregados, no hay un fundamento teórico que los caracterice cualitativamente, sin embargo, mientras mayor sea la diferencia entre el lado izquierdo y el lado derecho de (4.3), más lejos estará de la solución hallada el hiperplano definido por el corte, y por lo tanto se podría esperar que este tipo de cortes resulten ser más efectivos. Con lo cual, es interesante no sólo encontrar cliques con las que se viole la desigualdad, sino encontrar las cliques que lo hagan por un mayor margen. De esta forma, el problema de separación, para estas familias de desigualdades, puede reducirse a un problema de *clique de peso máximo*.

A continuación se comentan los dos algoritmos desarrollados para la búsqueda de cliques. Ambos reciben los siguientes parámetros:

- el conjunto de vértices $\mathcal{N}(k) \subseteq V$,
- el conjunto de aristas $E' \subseteq E$ inducido por $\mathcal{N}(k)$,
- la función de pesos de los vértices $\omega_k : W \to \mathbb{R}$,
- el valor a superar γ_k, y
- la cantidad de cliques a devolver

y el objetivo será encontrar cliques $K \subseteq \mathcal{N}(k)$ que cumplan (4.3).

4.2.1.1. Algoritmo de *backtracking*

El primer algoritmo desarrollado en este trabajo para la búsqueda de cliques utiliza la técnica de *backtracking* para inspeccionar "todas"[1] las cliques posibles dentro de un conjunto de vértices, el cual como se mostró previamente corresponde a la vecindad de algún vértice k en particular.
Para poder explicar mejor este algoritmo visualizaremos el conjunto de posibles cliques utilizando un árbol binario. Cada nivel del mismo corresponde a un vértice del conjunto, y desde cada nodo de ese nivel, el subárbol izquierdo representa los conjuntos de vértices que contienen al vértice en cuestión y el subárbol derecho los conjuntos que no lo contienen. Así, para cualquier subconjunto de vértices existe un camino desde la raíz del árbol hasta el último nivel que representa a dicho subconjunto. En la Figura 4.1 puede visualizarse un árbol de backtracking para un conjunto de vértices $W = \{v_1, v_2, v_3, v_4, v_5\}$. El camino

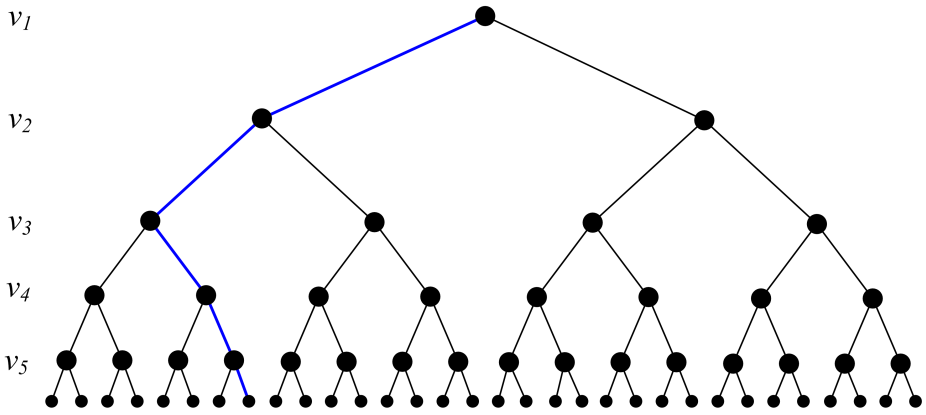


Figura 4.1: Arbol de *backtracking*

marcado en el mismo representa al subconjunto de vértices $W' = \{v_1, v_2\}$. Si este subconjunto induce una clique de G, entonces éste será un camino válido a recorrer por el algoritmo, en caso contrario el algoritmo no lo inspeccionará nunca.

El procedimiento para hallar una clique de peso máximo mediante esta técnica es el siguiente: se comienza armando una clique inicial de un sólo vértice con el primero del conjunto (es decir, v_1) y luego en cada paso se irán incorporando, en orden, los vértices que le siguen, siempre y cuando el vértice a incorporar sea adyacente a todos los vértices ya incorporados a la clique. Al llegar al final del camino, es decir al último nivel del árbol, la clique obtenida es registrada. El proceso de *bactracking*, a continuación, consiste en subir por el árbol hasta

[1] Veremos más adelante que el algoritmo no inspecciona el conjunto de cliques posibles en su totalidad sino que cuenta con un criterio de parada que impide que eventualmente se demore demasiado tiempo, ya que obviamente esta cantidad de posibilidades podría ser de orden exponencial.

el nodo correspondiente al último vértice agregado (v_4 en este ejemplo) y continuar por la rama del árbol derecha de ese nodo, evitando así incorporar dicho vértice a la clique. Finalmente, cuando se recorra la última rama del árbol (la de más a la derecha) se terminará la ejecución del algoritmo y se elegirá la clique de mayor peso que se haya encontrado. En el algoritmo 3 se representa este procedimiento, utilizando ω_k como la función de pesos de los vértices[2]. Para mayor

Algoritmo 3 Clique de peso máximo (*backtracking*)

Entrada: W, E', ω_k
1: $K \leftarrow \{v_1\}$
2: $i \leftarrow 2$
3: $mejorK \leftarrow K$
4: **loop**
5: **si** $\forall w \in K, wv_i \in E'$ **entonces**
6: $K \leftarrow K \cup \{v_i\}$
7: **fin si**
8: **si** $i < |W|$ **entonces**
9: $i \leftarrow i + 1$
10: **si no** /* final de esta rama */
11: **si** $\omega_k(K) > \omega_k(mejorK)$ **entonces**
12: $mejorK \leftarrow K$ /* actualiza mejor solución */
13: **fin si**
14: **si** $K = \emptyset$ **entonces** /* final del arbol */
15: **devolver** $mejorK$
16: **si no** /* realiza el backtracking */
17: $j \leftarrow$ `ultimoSubIndice`(K)
18: $K \leftarrow K \setminus \{v_j\}$
19: $i \leftarrow j + 1$
20: **fin si**
21: **fin si**
22: **fin loop**

claridad, el algoritmo mostrado devuelve una sola clique, y en este caso la de mayor peso. Sin embargo, en la implementación real de este algoritmo, se tienen en cuenta otros factores. En primera instancia, como se comentó en la sección anterior, el algoritmo recibe un parámetro que indica la cantidad de cliques a devolver y por lo tanto, en lugar de guardar sólo una, es necesario mantener una lista de cliques de al menos ese tamaño. Además de esto, todas las cliques devueltas tienen que cumplir (4.3), con lo cual es necesario también realizar esa

[2]Se utiliza $\omega_k(W) = \sum_{v \in W} \omega_k(v)$ como abuso de notación.

verificación antes de registrar la clique en cuestión.

Por otro lado, se agregó en la implementación la posibilidad de caracterizar las cliques devueltas por el algoritmo. Los valores posibles para este parámetro son:

`first N:` Al utilizar este valor, el algoritmo devolverá las primeras `N` cliques halladas.

`best N:` Al utilizar este valor, el algoritmo devolverá las `N` cliques de mayor peso halladas.

`all:` Al utilizar este valor, el algoritmo devolverá todas las cliques halladas.

En todos los casos las cliques devueltas deben obviamente cumplir (4.3).[3]

Puede verse que este algoritmo, tal cual está implementado, recorre en el peor de los casos una cantidad exponencial de nodos del árbol. Obviamente, esto no se puede permitir ya que la etapa de separación es ejecutada una gran cantidad de veces a lo largo de la resolución del *Branch & Cut*. Por lo tanto es necesario reducir el tiempo de ejecución de la misma, aunque esto signifique transformar el procedimiento en un método heurístico. A continuación se presentan diversas optimizaciones a este método, que ayudaron significativamente a reducir los tiempos de ejecución de este procedimiento:

Sólo pesos positivos

Para todas las familias puede verse que el valor de γ_k es siempre no negativo, sin embargo no ocurre lo mismo para los pesos de los vértices, y dado que lo que se está buscando es superar el valor de γ_k con estos pesos, el hecho de agregar vértices de peso negativo no ayuda. Por lo tanto, antes de comenzar el algoritmo se descartan todos los vértices cuyo peso sea negativo. Esto ayuda a reducir el tamaño del árbol.

Límite en los nodos inspeccionados

Pese a cualquier optimización que se implemente, el algoritmo seguirá siendo, al menos en el peor caso, de un orden exponencial. Es por ello que se requiere un criterio de parada duro. Este criterio es sencillamente un límite a la cantidad de nodos del árbol a inspeccionar. De esta forma es posible controlar los tiempos máximos de ejecución de esta etapa. Este límite se agrega como parámetro del algoritmo.

[3] Veremos más adelante que el algoritmo respeta también un límite impuesto a la cantidad de nodos del árbol a inspeccionar, con lo cual las opciones `best N` y `all` quedan sujetas a este valor.

Ordenamiento de vértices

El hecho de imponer un límite en la cantidad de nodos a inspeccionar, implica que sólo los primeros caminos del árbol serán las potenciales cliques a devolver. Con esto, es deseable que estos primeros candidatos sean los más prometedores, y esto se logra ordenando los vértices del conjunto según su peso de manera descendente. De esta forma, las primeras cliques se construirán de una manera *golosa* y por lo tanto tendrán más posibilidades de generar buenos cortes.

Acotación de los nodos

Esta optimización consiste en conseguir cotas superiores para las potenciales cliques de cada subárbol. Usaremos como ejemplo nuevamente la Figura 4.1. Supongamos que la clique conseguida por el camino marcado en la figura (es decir $W' = \{v_1, v_2\}$) resultó ser la clique de mayor peso hasta ese momento, y llamemos ω_{max} a este peso. El paso siguiente a haber llegado al final de una rama es realizar el procedimiento de *backtracking*, el cual en este caso nos llevaría al nodo marcado como A en la Figura 4.2.
Los siguientes pasos del algoritmo consistirían en inspeccionar todos los caminos posibles del subárbol correspondiente a este nodo del árbol, y en cada nodo del mismo, debería ver si el vértice correspondiente es adyacente a todos los vértices ya incorporados a la clique para decidir si es posible agregarlo o no. Sin embar-

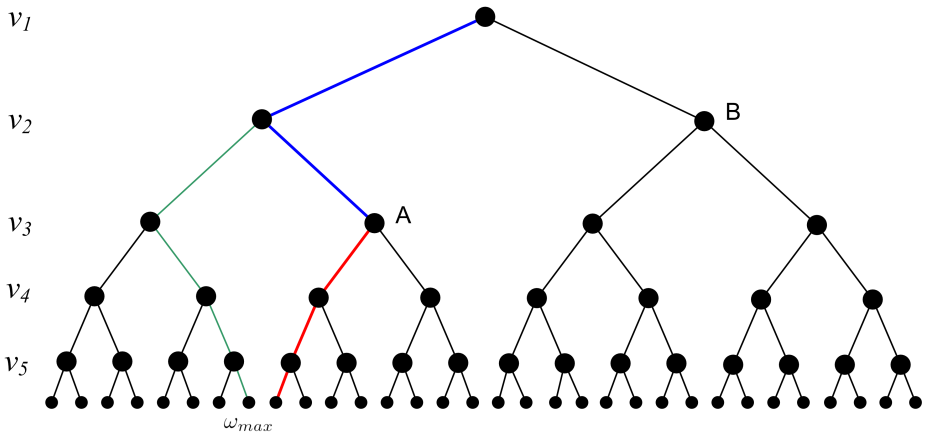


Figura 4.2: Obtención de cotas para un subárbol del *backtracking*

go, es posible obtener una cota superior para los pesos de las cliques formadas con este subárbol. Esta cota puede obtenerse asumiendo que se incluirán en la clique todos los vértices del subárbol (es decir, se tomará el camino marcado en rojo en la Figura 4.2). El cálculo del peso de este camino puede hacerse de manera inmediata sin siquiera revisar que el camino final sea precisamente una clique del grafo[4]. Supongamos entonces que el peso de esta potencial clique es ω

[4]De hecho, para optimizar este cálculo se pueden precalcular los pesos de los últimos k vértices para todo $k = 1, \ldots, |W|$ y de esta manera el peso de este camino se calculará en

y además $\omega < \omega_{max}$. Entonces, es posible "podar" este subárbol por completo y descartarlo continuando el proceso desde el nodo B del árbol.
Dado que los vértices están ordenados según su peso, es de esperar que las primeras cliques obtenidas sean las de mayor peso, con lo cual es de esperarse también que esto último suceda con frecuencia.

Esta idea es fácilmente adaptable al hecho de que el algoritmo devuelva más de una clique. En este caso se debe comparar ω con el peso de la clique de menor peso dentro de la lista actual de cliques de mayor peso.

4.2.1.2. Heurística golosa

El segundo método utilizado para la búsqueda de cliques de peso máximo es una heurística golosa. La misma comienza armando un conjunto unitario con el vértice de mayor peso. Luego, irá recorriendo el conjunto de vértices en orden de pesos y, ante cada uno de ellos, lo agregará a la clique si esto es posible, es decir si el mismo es adyacente a todos los vértices ya incorporados a la clique. El Algoritmo 4 muestra el pseudocódigo de la heurística mencionada. Este algoritmo,

Algoritmo 4 Clique de peso máximo (heurística)

Entrada: W, E', ω_k
1: $v \leftarrow$ `buscarMasPesado`(W, ω_k)
2: $W \leftarrow W \setminus \{v\}$
3: $K \leftarrow \{v\}$
4: **mientras** $W \neq \emptyset$ **hacer**
5: $v \leftarrow$ `buscarMasPesado`(W, ω_k)
6: $W \leftarrow W \setminus \{v\}$
7: **si** $\forall w \in K, vw \in E'$ **entonces**
8: $K \leftarrow K \cup \{v\}$
9: **fin si**
10: **fin mientras**
11: **devolver** K

sin embargo, devuelve una sola clique y, como se mencionó anteriormente, se requería la posibilidad de devolver más de una (esto en función de un parámetro de la heurística). La solución es ejecutar varias veces este procedimiento, variando el vértice inicial (línea 1 del algoritmo) entre todos los vértices de W.

4.2.2. Busqueda de cliques para las MCCK

El procedimiento de separación para las *MCCK inequalities* consta de dos partes. La idea es conseguir primero una desigualdad de tipo *CCK* que sea

tiempo O(1).

violada por la solución en cuestión y agregarla a la formulación actual. Luego, se intentará *extender* esta desigualdad para conseguir una o más desigualdades de tipo *MCCK*, para agregarlas también como cortes del poliedro.

4.2.2.1. Obtención de una desigualdad de tipo *CCK*

Como se comentó previamente, la búsqueda de planos de corte de esta familia de desigualdades difiere levemente de la de las demás. La primer diferencia notable se refiere al conjunto de colores consecutivos utilizado, pues en este caso la cantidad de conjuntos posibles no es lineal sino cuadrática[5]. Aunque por otro lado, para esta desigualdad no se requiere un vértice distinguido $k \in V$ y por lo tanto no es necesario recorrer además el conjunto de vértices. Se decidió entonces, en esta implementación, recorrer efectivamente todos los posibles conjuntos $Q \subseteq C$ de colores consecutivos e intentar encontrar una desigualdad utilizando cada uno de ellos. Así, dado $Q = \{c_1, \ldots, c_q\}$, sólo resta entonces conseguir una clique $K \subseteq V$ tal que

$$\sum_{v \in K} \left[x_{vc_1} + x_{vc_q} + \sum_{c \in Q \setminus \{c_1, c_q\}} 2x_{vc} \right] > (q-1) + \sum_{v,w \in K} z_{vw}. \qquad (4.4)$$

El procedimiento de separación para las demás familias, presentado en la Sección 4.2.1, utiliza fuertemente la definición de *peso* de un vértice, y este peso se define independientemente de la clique a la cual se agregue el vértice. Es sencillo ver que en este caso, el *peso* de un vértice, es decir el valor aportado por el mismo a la desigualdad, depende de la clique a la cual pertenezca.
En estas condiciones es posible definir el *peso* de un vértice, con respecto a la clique K, de la siguiente manera:

$$\omega_K(v) = \left[x_{vc_1} + x_{vc_q} + \sum_{c \in Q \setminus \{c_1, c_q\}} 2x_{vc} \right] - \sum_{w \in K} \frac{1}{2} z_{vw},$$

y así, (4.4) puede reescribirse como

$$\sum_{v \in K} \omega_K(v) > \gamma_Q, \qquad (4.5)$$

donde $\gamma_Q = (|Q| - 1) = (q - 1)$.
Tenemos ahora un problema similar al de *clique de peso máximo* pero con la diferencia de que los pesos de los vértices en este caso son "dinámicos", pues dependen de la clique a la cual pertenecen. Veremos cómo repercute esta característica en los algoritmos implementados para esta separación.

[5] La cantidad de subconjuntos de colores consecutivos de un conjunto $C = \{c_1, \ldots, c_t\}$ con dos o más elementos es $\binom{t}{2}$.

Algoritmo de *backtracking*

El hecho de no conocer *a priori* los pesos de los vértices conlleva algunas diferencias significativas con respecto al algoritmo de *backtracking* previamente implementado. En primera instancia, no es posible un ordenamiento de los vértices según su peso, así como tampoco es posible tener en cuenta el hecho de que el peso de un vértice sea negativo para descartarlo antes de comenzar el algoritmo. Por otro lado, tanto al agregar un vértice a la clique actual como al eliminarlo, el cálculo del nuevo peso de la clique es más costoso que antes. Y lo mismo ocurre con el cálculo de las cotas para los subárboles.
Con respecto al ordenamiento de los vértices, y con el objetivo de no descartar esta idea, se definió el *peso estimado* de un vértice, independientemente de la clique a la cual pertenezca, de la siguiente manera:

$$\tilde{\omega}(v) = \quad x_{vc_1} + x_{vc_q} + \sum_{c \in Q \setminus \{c_1, c_q\}} 2x_{vc}. \tag{4.6}$$

La idea es utilizar este valor como un estimador del peso real de los vértices para realizar el ordenamiento previo de los mismos.
Por otro lado, es facil ver que $\tilde{\omega}(v)$ es siempre mayor o igual que $\omega_K(v)$ para todo $v \in V$, y por lo tanto esta estimación puede utilizarse también para calcular cotas para los subárboles de una manera muy eficiente[6].

De esta manera, el algoritmo de backtracking implementado para obtener una desigualdad de tipo *CCK* resulta muy similar al primero, salvo por las diferencias recién mencionadas.

Heurística golosa

Para la heurística golosa, el hecho de conocer los pesos de los vértices *a priori* resulta fundamental, ya que es casi el único elemento que se utiliza para armar la clique. La solución en este caso fue también utilizar el peso estimado definido en (4.6) como reemplazo del peso real.

4.2.2.2. Extensión de intervalos de color para la *MCCK*

Luego de obtener una desigualdad de tipo CCK y haber agregado el corte generado por la misma, se procede a extenderla, si es posible, para construir una desigualdad *MCCK* a partir de ella.

Sean entonces $Q^1 = \{c_1^1, \ldots, c_{q_1}^1\} \subseteq C$ y $K \subseteq V$ el conjunto de colores consecutivos y la clique hallada, respectivamente. Lo que se busca es poder encontrar nuevos conjuntos de colores consecutivos $Q^2, \ldots, Q^p \subseteq C$ para construir una desigualdad de tipo *MCCK*.
El algoritmo comienza entonces buscando un conjunto de colores consecutivos

[6] Obviamente, estas cotas no serán tan buenas como las que se pueden calcular con los pesos reales, pero se gana en eficiencia temporal.

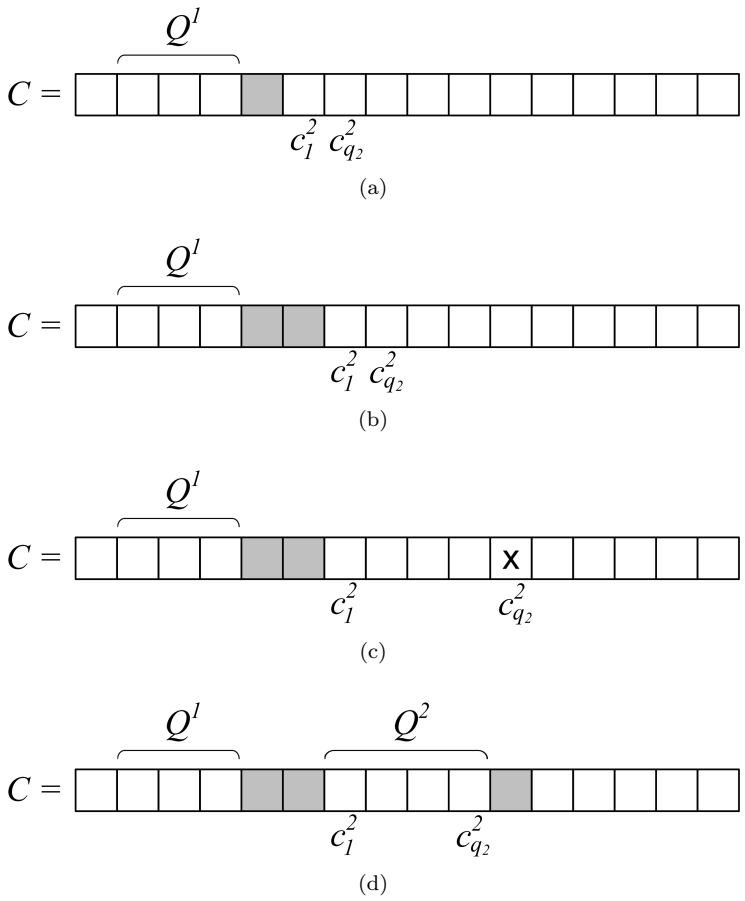


Figura 4.3: Extensión de intervalos de color para la *MCCK*

$Q^2 = \{c_1^2, \ldots, c_{q_2}^2\}$ (no adyacente a Q^1). Para ello ubica primero los colores extremos del nuevo conjunto tal como se ve en la Figura 4.3(a). En el caso en que la desigualdad *MCCK* resultante de esta disposición de colores no sirva para cortar la solución fraccionaria en cuestión, se avanzarán ambos colores como se puede ver en la Figura 4.3(b), y así sucesivamente hasta que esto no ocurra. Luego, al haber hallado un posible intervalo para Q^2, se procede a intentar ampliar el intervalo haciendo avanzar $c_{q_2}^2$, siempre y cuando la desigualdad resultante siga sirviendo como corte para $\hat{y}$. En el momento en que esto deje de cumplirse (representado en la Figura 4.3(c)), se interrumpe este proceso de avance y se fija el color $c_{q_2}^2$ en el color inmediatamente anterior al que falló. De esta manera, tal como se ve en la Figura 4.3(d), queda definido el nuevo conjunto de colores consecutivos Q^2. Agregando este conjunto al que ya se tenía, se arma una nueva desigualdad válida de tipo *MCCK* y se la agrega a la definición del poliedro como corte para $\hat{y}$.

Este procedimiento se repite hasta que ya no se pueda hallar un nuevo conjunto de colores consecutivos.

4.3. Redondeo de la solución

Un punto fundamental en un algoritmo *Branch & Cut* es conseguir soluciones enteras a medida que avanza la ejecución del mismo. Estas soluciones representan cotas que pueden ser usadas para reducir drásticamente el tamaño del árbol de *branching*. Teniendo una solución entera, si en la relajación lineal de un subproblema se obtiene como óptimo un valor peor que esta solución, entonces este subproblema puede darse por cerrrado (evitando también todos sus potenciales subproblemas).
La técnica de *redondeo de soluciones* utiliza información de las soluciones fraccionarias conseguidas en las relajaciones para intentar deducir de ellas soluciones enteras válidas para el problema. Esta técnica, si bien es de uso general, se ve muy beneficiada si se usan características particulares del problema que se está resolviendo.

En este caso, dada una solución fraccionaria $\hat{y} = (\hat{x}, \hat{z}) \in \mathbb{R}^{nt+m}$, se intenta reconstruir un coloreo válido a partir de ella. Para ello, se comienza por identificar la variable fraccionaria más cercana a 1, dentro del conjunto de variables de asignación x_{vc}. Llamemos v y c al vértice y al color, respectivamente, asociados a dicha variable. Se puede ver fácilmente que que $\hat{x}_{wc} < 1$ para todo $w \in \mathcal{N}(v)$, pues $\hat{y}$ es una solución factible. Se considera entonces que el vértice v será coloreado con el color c, y por lo tanto se fija esta variable en 1. Luego, para mantener la validez de esta solución, se fijan en cero todas las variables $\hat{x}_{vc'}$ para todo color $c' \neq c$ y con el mismo objetivo se fijan en cero las variables

Algoritmo 5 Redondeo de solución

Entrada: $\hat{y} = (\hat{x}, \hat{z})$
1: **mientras** haya $\hat{x}_{vc}$ fraccionaria **hacer**
2: elegir (v, c) tal que $\hat{x}_{vc} = \text{máx}\{x_{vc} : 0 < x_{vc} < 1\}$
3: fijar $\hat{x}_{vc} = 1$
4: fijar $\hat{x}_{vc'} = 0$ para todo $c' \in C \setminus \{c\}$
5: fijar $\hat{x}_{wc} = 0$ para todo w tal que $vw \in E$
6: **fin mientras**
7: **si** $\sum_{c \in C} \hat{x}_{vc} = 1$ para todo $v \in V$ **entonces**
8: **devolver** $\hat{y}$
9: **si no**
10: no se encontró solución
11: **fin si**

$\hat{x}_{wc}$ para todo $w \in \mathcal{N}(v)$.

Finalmente, cuando todas las variables de asignación poseen un valor entero, se verifica si cada vértice del grafo recibe exactamente un color [7]. De ser así, se ha obtenido un coloreo válido y por lo tanto, fijando las variables de adyacencia en sus valores correspondientes, se obtiene una solución factible entera para el problema. Si al finalizar el algoritmo algún vértice ha quedado sin color asignado, entonces la solución hallada, si bien entera, no es válida. El Algoritmo 5 ilustra este procedimiento.

4.4. Selección de la variable de branching

El método de *branching* consiste en elegir una variable fraccionaria en la solución obtenida y generar a partir de ella dos subproblemas. En uno de ellos se agregará una restricción que obliga a esta variable a tomar el valor 0, y en el otro a tomar el valor 1.
La variable elegida para realizar sobre ella el *branching* puede ser cualquier variable cuyo valor sea fraccionario. Sin embargo, a veces la fijación de algunas variables influyen más que la de otras.
En nuestro caso, fijar una variable de asignación x_{vc} puede influenciar más que fijar una variable de adyacencia z_{vw}. Por este motivo, en la implementación del *Branch & Cut* se decidió priorizar el *branching* de este tipo de varables.

4.5. Fijación por implicaciones lógicas

Como se mencionó en la sección anterior, el método de *branching* impone, en ambos subproblemas, condiciones sobre la variable elegida. Si bien estas condiciones son sólo para dicha variable, muchas veces implican condiciones para otras variables relacionadas con ella.
Si bien estas implicaciones se dan como consecuencia de determinadas restricciones del modelo, es cierto que si se las identifica previamente, es posible reducir el tamaño de los subproblemas eliminando determinadas variables mediante fijación de valores sobre ellas.

El algoritmo de fijación de variables tiene dos etapas. En primer lugar, se identifica las variables de asignación que se hayan fijado en el valor 1, es decir, que indiquen que un determinado vértice v recibe un determinado color c. Con esta información es posible fijar en cero todas las variables $x_{vc'}$ con $c' \neq c$, y todas las variables x_{wc} con $w \in \mathcal{N}(v)$. La segunda parte del algoritmo, recorre cada arista $vw \in E$ y verifica si ambos vértices tienen ya un color asignado, es decir, si existen $c_1, c_2 \in C$ tal que las variables x_{vc_1} y x_{wc_2} han sido fijadas en

[7] Vale aclarar que por la manera de funcionar del algoritmo, las posibilidades al final del mismo para cada vértice son sólo dos: que el vértice reciba exactamente un color, o que no reciba ninguno.

1. De ser así, ya es posible fijar el valor de la variable de adyacencia z_{vw} en 1, si los colores son adyacentes, y en 0 si no. Estos procedimientos se formalizan en el Algoritmo 6.

Algoritmo 6 Fijación por implicaciones lógicas

Entrada: $\hat{y} = (\hat{x}, \hat{z})$
para todo $v \in V$ y $c \in C$ **hacer**
  si $\hat{x}_{vc} = 1$ **entonces**
    fijar $\hat{x}_{vc'} = 0$ para todo $c' \in C \setminus \{c\}$
    fijar $\hat{x}_{wc} = 0$ para todo w tal que $vw \in E$
  fin si
fin para
para todo $vw \in E$ y $c_1, c_2 \in C$ **hacer**
  si $\hat{x}_{vc_1} = 1$ y $\hat{x}_{wc_2} = 1$ **entonces**
    fijar $\hat{z}_{vw} = 1$ **si** $|c_1 - c_2| = 1$
    fijar $\hat{z}_{vw} = 0$ **si no**
  fin si
fin para

CAPÍTULO 5

Resultados computacionales

En este capítulo se presentan los resultados computacionales del *Branch & Cut* implementado. En primera instancia se describe la preparación del conjunto de instancias de prueba que fue utilizado a lo largo de la experimentación. Luego se presentan los resultados de la misma, la cual está dividida en cuatro etapas progresivas. La primer etapa consiste en la separación individual de cada una de las familias de desigualdades, es decir, para cada familia se realizó un conjunto de pruebas independiente en el cual sólo se utilizaba dicha familia para el procedimiento de separación. En la segunda etapa, se combinan las familias de desigualdades para buscar de esta manera la combinación que obtenga mejores resultados. A continuación, en la tercera etapa, se utiliza la combinación de familias obtenida y se experimenta con los dos algoritmos de separación implementados y con los distintos parámetros de los mismos. Finalmente, utilizando la mejor combinación de parámetros obtenidos, la cuarta etapa consiste en la variación de parámetros propios del *Branch & Cut* con el objetivo de conseguir la mejor configuración posible, al menos para este juego de instancias de prueba. Estos parámetros, a saber el *skip factor* y la cantidad de rondas de cortes aplicadas en cada iteración, pueden variar significativamente los resultados obtenidos.

Este capítulo se cierra con algunas conclusiones surgidas de los resultados obtenidos y finalmente, con una comparación de los mismos contra los resultados de la ejecución de CPLEX utilizando el modelo original. Es decir, se ejecuta CPLEX sobre las instancias sin el agregado de los cortes presentados en este trabajo.

5.1. Preparación de las instancias y contexto de pruebas

Con el objetivo de conseguir instancias de prueba que resulten desafiantes para el *Branch & Cut*, se recurrió a las instancias de la experimentación hecha

en la Sección 2.4 y se eligieron de allí las instancias que resultaron más difíciles de resolver. De esta manera, se construyó un conjunto de 30 instancias, compuesto por las 10 instancias más difíciles de los grupos de 16, 18 y 20 vértices.

Los experimentos fueron ejecutados en una computadora con 2 Gb de memoria RAM y un microprocesador AMD Athlon© 64 corriendo a 1.5 GHz. Todas las ejecuciones realizadas cuentan con un límite de tiempo máximo de ejecución de media hora. Concurrido este tiempo, el proceso se interrumpe y se registran las mejores cotas obtenidas hasta ese momento.

5.2. Etapa I: Separación individual

La primer etapa de la experimentación consistió en realizar pruebas con cada familia de desigualdades por separado. Para cada una de ellas, se ejecutó el algoritmo *Branch & Cut* utilizando cuatro configuraciones distintas para el procedimiento de separación. Estas configuraciones corresponden a:

- Algoritmo de *bactracking* con la opción `first 10`,
- Algoritmo de *bactracking* con la opción `best 10`,
- Algoritmo de *bactracking* con la opción `all` y
- Heurística golosa devolviendo 10 cliques.

Por otro lado, se agregó para cada familia el procedimiento de separación para las *clique inequalities* dado que éstas, según [16], resultaron ser muy efectivas para el problema clásico de coloreo. Además, se realizaron también pruebas individuales para esta familia, es decir sin la incorporación de ninguna de las familias de desigualdades presentadas en este trabajo.

5.2.1. Resultados

Los resultados obtenidos para las familias *3CIK*, *3COK* y *4CVK* fueron realmente desalentadores, pues en ningún caso se logró resolver en forma óptima ninguna de las instancias propuestas en el límite de tiempo dado. En la mayoría de ellas se alcanzaba este límite con un *gap* de optimalidad grande o incluso muchas veces sin haber encontrado una solución factible. En el resto de las instancias simplemente el proceso se quedaba sin memoria y debía ser interrumpido.

Con respecto a las *clique inequalitites*, utilizadas individualmente, el resultado fue el mismo. Sin embargo, en las pruebas que combinaban éstas con las *MCCK* se obtuvieron resultados significativamente mejores. En la mayoría de los casos, las instancias eran resueltas en cuestión de decenas de segundos y en algunos casos en cuestión de minutos. Hubo, igualmente, instancias que no pudieron ser resueltas en forma óptima dentro del límite de tiempo.

Una conjetura surgida de estos últimos resultados fue el hecho de que los cortes que realmente ayudaron a los tiempos son sólo los de tipo *MCCK* y que los de tipo *K* en realidad no beneficiaban al procedimiento. Sin embargo, para descartar esta conjuetura se realizaron pruebas utilizando sólamente las *MCCK* y en este caso los resultados obtenidos fueron tan malos como los obtenidos para las demás familias, no pudiendo resolver ninguna de las instancias propuestas. De aquí, se deduce que es la combinación entre estas dos familias la que realmente ayuda, y no alguna de ellas por separado.

Las *clique inequalities* no utilizan variables de adyacencia, puediendo decirse entonces que su objetivo es ayudar a la definición de un coloreo válido. Por otro lado, las *MCCK inequalities* apuntan a la definición apropiada de la función objetivo imponiendo cotas para las variables de adyacencia. Es interesante el hecho de que la combinación de estos dos elementos (dominio del problema por un lado y función objetivo por otro), típica de un problema de optimización, sea la que obtenga mejores resultados.

Esta combinación (*MCCK+K*) fue entonces la elegida como base para continuar con la experimentación. No se presentan los resultados de esta etapa en forma numérica ya que los mismos se verán en la siguiente sección comparados con otros resultados.

5.3. Etapa II: Combinaciones entre las familias

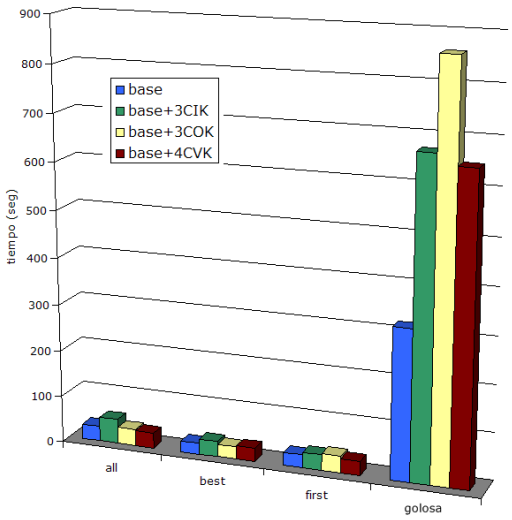
Partiendo entonces de la combinación base de las familias *MCCK+K*, esta segunda etapa de experimentación consiste en intentar agregar las familias que quedaron fuera, es decir las *3CIK*, *3COK* y *4CVK*, para ver si con ellas se pueden mejorar los resultados. Se proponen entonces las siguientes combinaciones de familias para realizar las pruebas con ellas:

- *base*
- *base + 3CIK*
- *base + 3COK*
- *base + 4CVK*

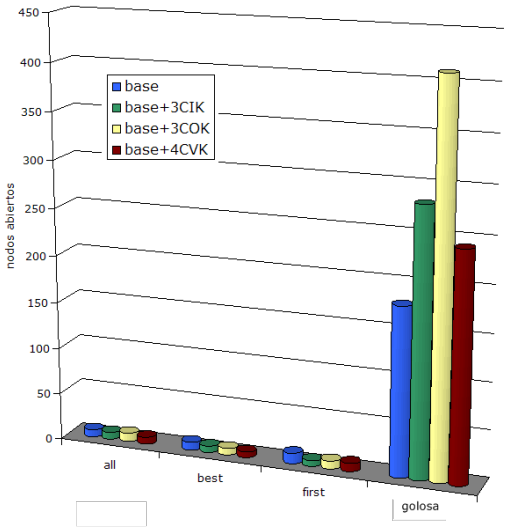
donde *base = MCCK+K*.
Además, para cada una de estas familias se utilizan nuevamente las cuatro configuraciones presentadas en la sección anterior.

5.3.1. Resultados

La Figura 5.1 muestra los resultados obtenidos en esta etapa. En la parte (a) se muestran los tiempos de ejecución promedio para cada familia y cada



(a) Tiempos de ejecución



(b) Nodos recorridos

Figura 5.1: Combinaciones de familias y métodos de separación

combinación, y en la parte (b) se puede ver la cantidad de nodos promedio abiertos por el algoritmo de *Branch & Cut*. Las cuatro series de las figuras correponden a las 4 combinaciones de familias mencionadas previamente, y los resultados están a su vez agrupados según el método de separación utilizado, los cuales se pueden ver en los rótulos del eje x de los gráficos.

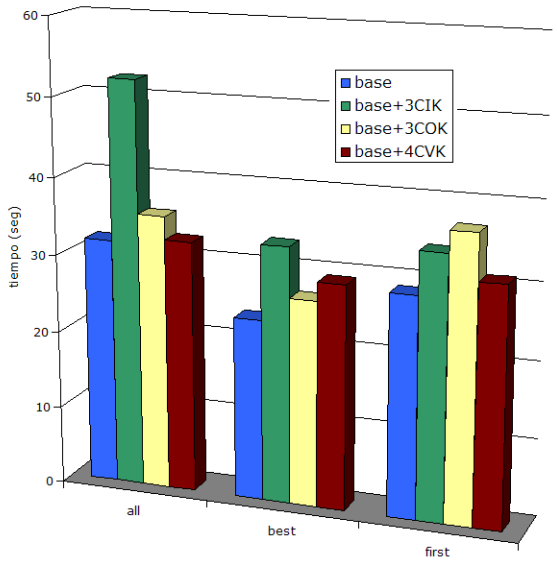
Lo primero que sobresale en estos resultados es el rendimiento de la heurística golosa en comparación con el método de *backtracking*, en cualquiera de sus tres versiones. Se puede ver que tanto los tiempos de ejecución como la cantidad de nodos abiertos al utilizar esta heurística es extremadamente superior a los valores obtenidos por el *backtracking*. Esta diferencia tan marcada llleva a descartar la heurística y continuar el análisis utilizando sólo el algoritmo de *backtracking*.

En la Figura 5.2 se muestra un detalle de los valores obtenidos por el algoritmo de *backtracking*. Al igual que antes, pueden verse los tiempos de ejecución en el primer gráfico y los nodos abiertos en el segundo.

Con respecto a los tiempos de ejecución, puede notarse que el mejor desempeño se obtiene utilizando la combinación de *base* y el método de separación de *backtracking* que devuelve las mejores cliques halladas.

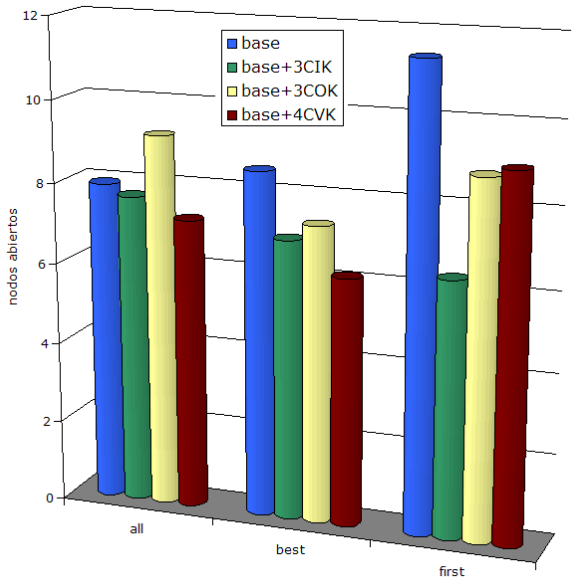
En cuanto a la cantidad de nodos abiertos, el método con mejor desempeño es el mismo pero en este caso la mejor combinación resulta ser *base+4CVK*. Sin embargo, los valores obtenidos para la cantidad de nodos abiertos son extremadamente bajos, con lo cual estas diferencias resultan ser poco significativas.

Finalmente, priorizando el mejor desempeño en el tiempo de ejecución por



(a) Tiempos de ejecución

Figura 5.2: Combinaciones de familias y métodos de separación (detalle)



(b) Nodos recorridos

Figura 5.2: Combinaciones de familias y métodos de separación (detalle)

sobre la cantidad de nodos abiertos, esta etapa concluye con la elección de la combinación de familias de *base* y la utilización del algoritmo de *backtracking* devolviendo las *mejores* cliques halladas.

5.4. Etapa III: Parámetros de la separación

Una vez obtenida la combinación de familias de desigualdades válidas y el método de separación a utilizar, se procede a experimentar con los distintos parámetros del método, el cual es en este caso el algoritmo de *backtracking* devolviendo las *mejores* cliques halladas.

Un primer parámetro importante a probar es la cantidad de cliques devueltas por este algoritmo, ya que esto determina en gran medida dos aspectos del *Branch & Cut*; por un lado, mientras más cortes se introduzcan al modelo, más se espera que se recorte el poliedro y por lo tanto mayor es la "porción" de soluciones fraccionarias que se elimina, sin embargo, a su vez estos cortes aumentan el tamaño de los subproblemas haciendo que las sucesivas relajaciones lineales demoren más tiempo en resolver. Esto representa un *trade-off* que vale la pena analizar. Se hace variar este parámetro `N` con valores del intervalo $[1, \ldots, 34]$[1].

[1]Se toman sólo los valores pares para disminuir la cantidad de pruebas a ejecutar.

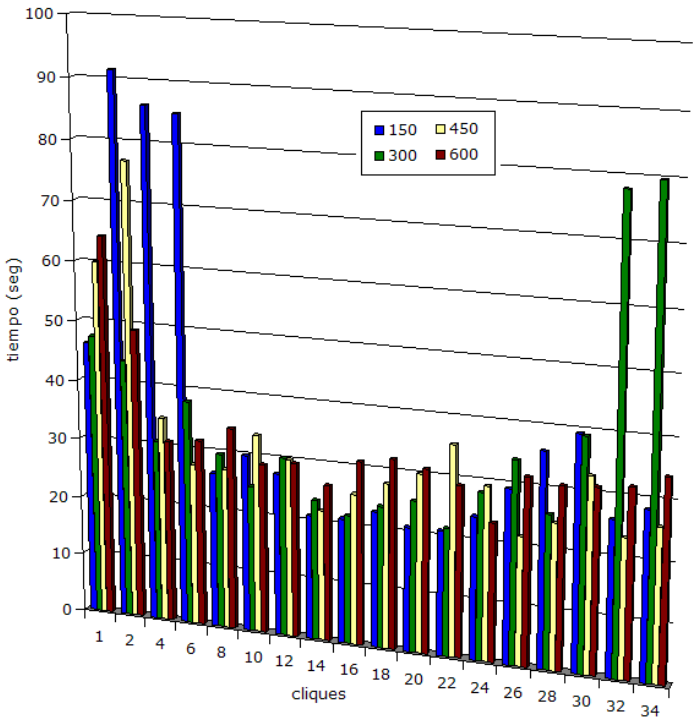
El segundo parámetro tenido en cuenta en esta etapa es el límite impuesto a la cantidad de nodos a recorrer en el árbol de *backtracking*. Este valor se fijó en cuatro posibles alternativas, siendo estas 150, 300, 450 y 600 nodos.

Se ejecutó el *Branch & Cut* sobre el conjunto de instancias de prueba para cada una de las posibles combinaciones entre estos dos parámetros.

5.4.1. Resultados

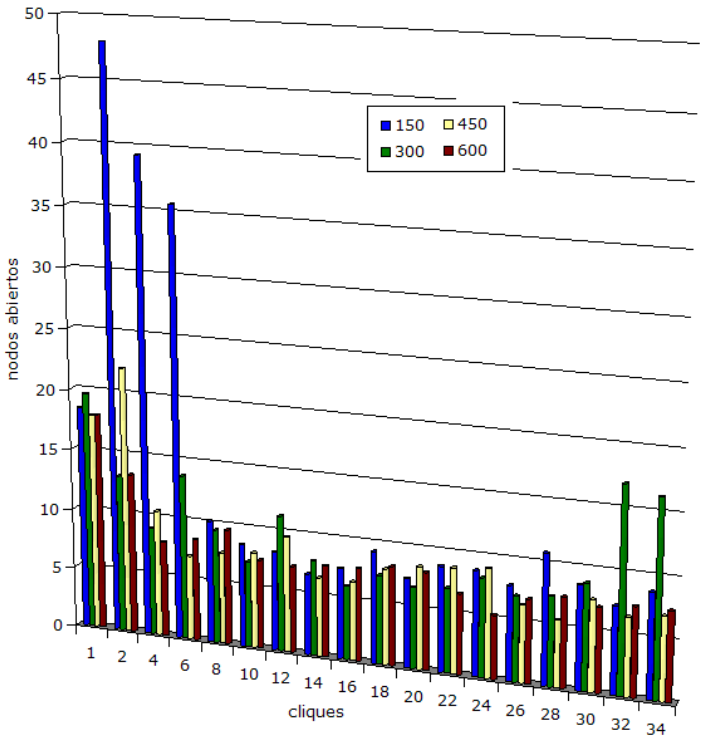
La Figura 5.3 muestra los resultados obtenidos en esta etapa. En la parte (a) se muestran los tiempos de ejecución promedio para cada valor de `N` y cada límite de nodos para el árbol del backtracking, y en la parte (b) se puede ver la cantidad de nodos promedio abiertos por el algoritmo de *Branch & Cut*. Las series de las figuras correponden a los límites de tiempo mencionados, y en el eje x de los gráficos se muestran los distintos valores para el parámetro `N`.

Se puede observar claramente en ambas figuras que los mejores resultados



(a) Tiempos de ejecución

Figura 5.3: Cantidad de cliques devueltas y límite de nodos

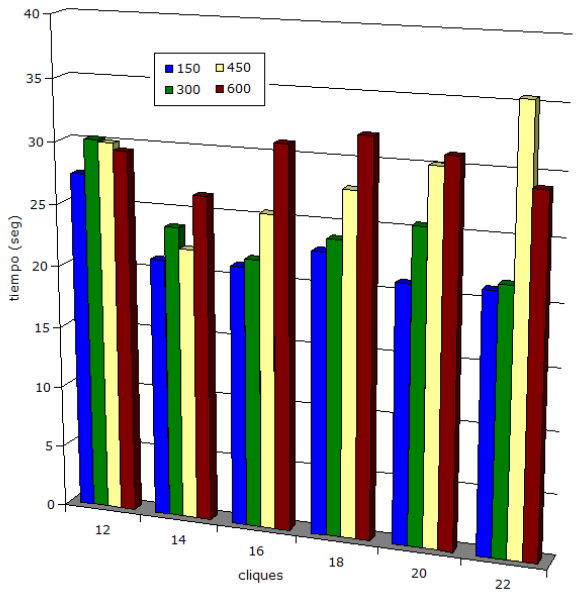


(b) Nodos recorridos

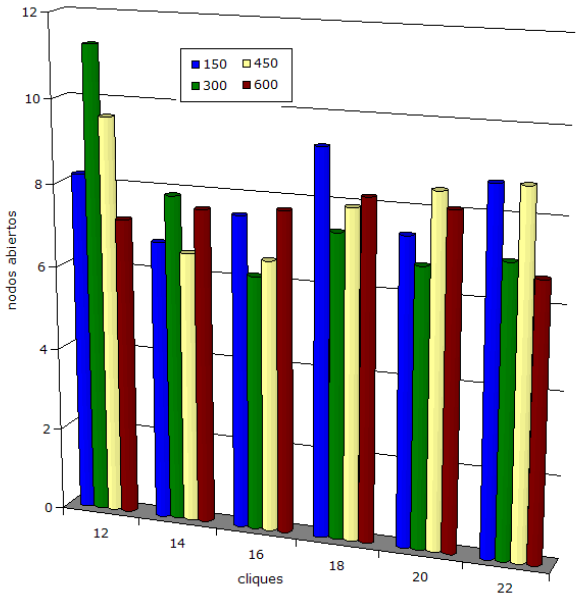
Figura 5.3: Cantidad de cliques devueltas y límite de nodos

se obtienen cuando el valor de `N`, es decir la cantidad de cliques devueltas, se encuentra entre 12 y 22. Por este motivo se presenta en la Figura 5.4 un detalle de estos resultados, en el cual se pueden ver claramente los valores a analizar. De nuevo, la parte (a) muestra los tiempos promedio de ejecución y la parte (b) muestra los nodos recorridos por el *Branch & Cut*.

Con respecto a los tiempos de ejecución, puede verse que en general se obtienen mejores resultados cuando se utiliza el límite de 150 nodos para el *backtracking*. Este es un hecho interesante ya que parece indicar que las mejores cliques se encuentran siempre dentro de las primeras ramas del árbol y no es necesario perder más tiempo buscando en las demás. Esto probablemente se deba a las optimizaciones mencionadas en la Sección 4.2.1.1, ya que el objetivo de las mismas era exactamente este.
Con respecto al valor para la cantidad de cliques devueltas, al parecer utilizando valores entre 14 y 20 resulta ser lo más apropiado, sin haber diferencias notables entre estos valores.



(a) Tiempos de ejecución



(b) Nodos recorridos

Figura 5.4: Cantidad de cliques devueltas y límite de nodos (detalle)

En la parte (b) de esta figura se presenta la cantidad promedio de nodos recorridos. Nuevamente, las diferencias con respecto a estos valores no resultan significativas. De todas maneras, se verifica que los mejores resultados se obtienen utilizando un valor de `N` entre 14 y 20. No hay tampoco una diferencia significativa con respecto al límite de nodos impuesto.

La etapa III concluye entonces indicando que los mejores resultados, al menos sobre este conjunto de instancias, se obtienen utilizando un valor de `N` entre 14 y 20 y un límite de 150 nodos para el árbol de backtracking.

5.5. Etapa IV: Parámetros del branch & cut

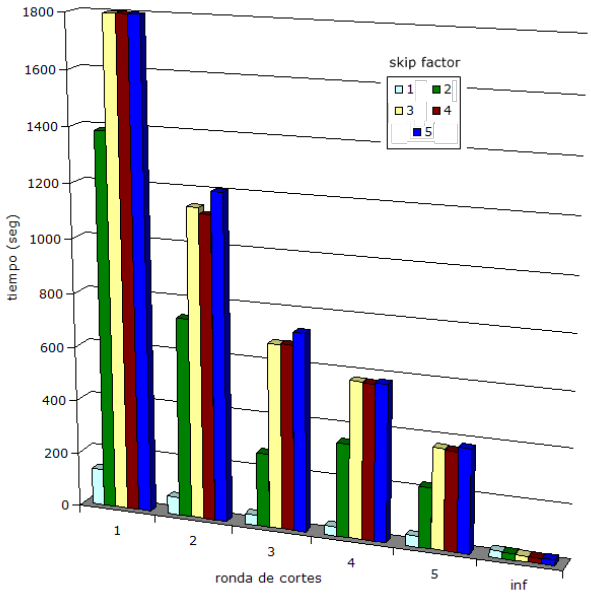
Finalmente, en esta sección se presentan los resultados obtenidos al probar diferentes valores para algunos de los parámetros propios del *Branch & Cut*.
El primero de ellos es el *skip factor*; este parámetro indica la frecuencia con la cual se aplica una fase de búsqueda de planos de corte en los subproblemas del árbol de *Branch & Cut*. Por ejemplo, un *skip factor* de 3 indica que esta fase se aplica solamente en uno de cada 3 subproblemas analizados. En los subproblemas en los que no se aplica esta fase, simplemente se realiza el procedimiento de *branching*.

Además del *skip factor*, el segundo parámetro con el cual se experimenta en este trabajo es la cantidad de rondas de cortes en cada fase de búsqueda de planos de corte. Luego de cada relajación lineal se buscan cortes utilizando el procedimiento de separación y al finalizar se vuelve a resolver la relajación lineal en busca de una nueva solución. Este parámetro indica cuántas relajaciones lineales se resuelven en cada subproblema antes de realizar finalmente el *branching* sobre el mismo.

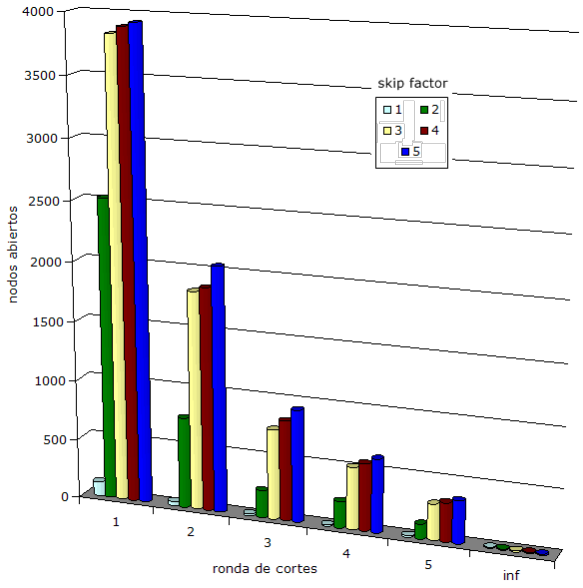
Cada uno de estos parámetros representa un *trade-off* interesante para analizar pues si bien el hecho de agregar cortes reduce el poliedro en gran medida, cada fase de búsqueda de planos de corte significa ejecutar varios procedimientos de separación que pueden ser muy costosos.
Tanto para el *skip factor* como para la cantidad de rondas de cortes se experimenta con valores entre 1 y 5. Para el segundo parámetro, además se agrega el valor "infinito", el cual representa el hecho de aplicar rondas de corte hasta que no se hallen más cortes para la solución encontrada.

5.5.1. Resultados

En la Figura 5.5 pueden verse los resultados de esta etapa. Como en las secciones anteriores, la figura consta de dos partes en las que se muestran los tiempos promedio de ejecución y la cantidad promedio de nodos abiertos.



(a) Tiempos de ejecución



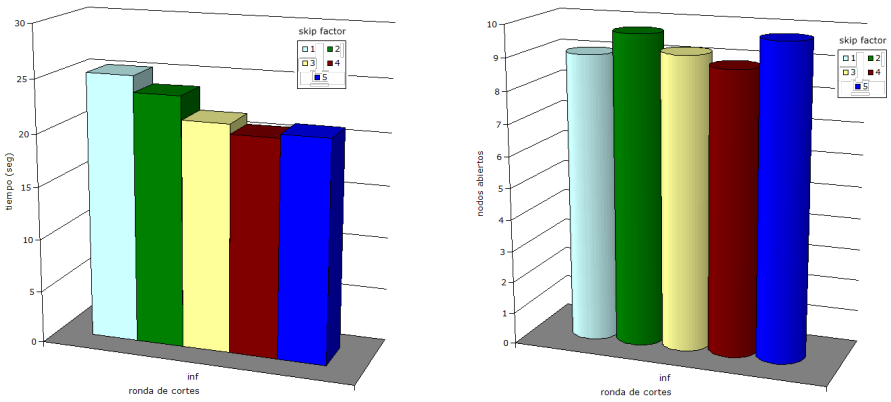
(b) Nodos recorridos

Figura 5.5: *Skip factor* y cantidad de rondas de cortes

Con respecto a los tiempos de ejecución, puede verse claramente el hecho de que mientras menos rondas de cortes se apliquen, peores resultados se consiguen. Al utilizar sólo una ronda de cortes, por ejemplo, se puede ver que con un *skip factor* de 3, 4 o 5 se alcanza el límite de tiempo en todas las instancias no pudiendo resolverse ninguna[2].
Un aspecto remarcable de este gráfico es la influencia del *skip factor* en los tiempos de ejecución. Puede verse que aunque se utilicen pocas rondas de cortes, si éstas se utilizan en todos los subproblemas (es decir, el *skip factor* es 1), los resultados son abruptamente mejores que en el resto de los casos. A su vez, puede verse que los tiempos conseguidos con un *skip factor* de 1, van también mejorando mientras más rondas de cortes se realicen.

Con respecto a la cantidad de nodos abiertos el resultado es exactamente el mismo. En este caso, de hecho, la diferencia entre un *skip factor* de 1 y uno mayor es mucho más marcada.

Se puede ver claramente que los mejores resultados se obtienen al utilizar el valor de "infinito" para la cantidad de rondas de cortes, sin embargo, dada la escala de los gráficos no es posible distinguir en detalle esa parte del gráfico. La Figura 5.6 muestra un detalle de esta parte del gráfico en el cual se puede ver que las diferencias entre los distintos valores para el *skip factor* en estas partes del gráfico no resultan ser significativas.



(a) Tiempos de ejecución (b) Nodos recorridos

Figura 5.6: *Skip factor* y cantidad de rondas de cortes (detalle)

Finalmente, podemos decir que la configuración más eficiente en este aspecto corresponde a utilizar tantas rondas de cortes como se pueda y efectuar la fase

[2]Como se mencionó previamente, el límite de tiempo es de media hora, es decir 1800 segundos.

de *cutting* en todos los subproblemas analizados. Esta conclusión resulta ser extremadamente interesante ya que indica que los cortes dados por las familias de desigualdades utilizadas son realmente eficaces a la hora de reducir el poliedro con la intención de llevarlo hacia la cápsula convexa de las soluciones enteras del mismo.

5.6. Conclusiones

La mejor combinación de familias, al menos para este conjunto de instancias de prueba, resulta ser *MCCK+K*. El resto de las familias de desigualdades válidas presentadas en este trabajo parecen no favorecer a los tiempos de ejecución del *Branch & Cut*.

En cuanto al procedimiento de separación, se puede ver que los mejores resultados se obtienen utilizando un algoritmo de *backtracking*, y en particular en su versión que encuentra las mejores 20 cliques, dentro de un límite de 150 nodos recorridos en el árbol de *backtracking*. El hecho de que al aumentar este límite no mejoren los tiempos de ejecución resulta interesante, como se comentó previamente, pues indica que las cliques buscadas se encuentran dentro de las primeras ramas del árbol, verificando así las hipótesis planteadas al implementar las diversas optimizaciones para el algoritmo.

Con respecto a los parámetros correspondientes al *skip factor* y la cantidad de rondas de cortes a utilizar en cada fase de *cutting*, las pruebas indican que los mejores resultados se obtienen al generar la mayor cantidad de cortes posibles en todos los subproblemas analizados. Este resultado indica que los cortes propuestos, junto con el procedimiento de separación, son altamente eficientes y parecen ayudar en forma decisiva a la resolución.

5.7. Comparación contra CPLEX

Presentamos en esta sección una comparación de los resultados obtenidos contra los resultados obtenidos por CPLEX, sobre el modelo original. En esta ocasión se utilizó la versión 9.0 de CPLEX y las ejecuciones se realizaron en la misma máquina descripta en la Sección 5.1, en este mismo capítulo.

La Tabla 5.1 muestra los tiempos de ejecución individuales de cada una de las instancias ejecutadas tanto por el *Branch & Cut* como por CPLEX. Se muestra también, para el *Branch & Cut*, la cantidad de nodos abiertos por el mismo, la cantidad de cortes utilizados de cada familia (K y MCCK) y el tiempo utilizado en el proceso de separación. Para el caso del CPLEX, en los casos en que se haya alcanzado el límite de tiempo[3], se muestra el *gap* de optimalidad obtenido.

[3]Representado en la tabla con los símbolos ***.

$\|V\|$	Densidad	*Branch & Cut*					CPLEX	
		Nodos	K	MCCK	Tiempo sep(s)	Tiempo(s)	Tiempo(s)	GAP
14	48,35 %	3	169	1196	1,02	**1,91**	**240**	
	49,45 %	5	116	553	0,93	**1,72**	**16**	
	49,45 %	5	137	676	1,47	**2,42**	**380**	
	56,04 %	5	242	1139	1,2	**2,61**	***	33 %
	57,14 %	9	229	1024	2,38	**4,26**	**108**	
	58,24 %	11	273	1329	3,39	**6,39**	**188**	
	59,34 %	1	279	1531	0,73	**2,1**	**149**	
	60,44 %	13	224	1317	3,64	**7,03**	**63**	
	64,84 %	2331	640	6728	679,89	**1714,81**	**1650**	
	65,93 %	2593	497	7145	662,9	***/9 %	***	9 %
16	55,00 %	5	227	1490	2,53	**5,18**	***	58 %
	56,67 %	3	156	1477	2,12	**4,09**	***	22 %
	56,67 %	15	311	1900	5,39	**12,04**	***	69 %
	60,00 %	1	288	1814	1,7	**4,66**	***	45 %
	60,00 %	1	288	1814	1,7	**4,63**	***	45 %
	60,00 %	1	288	1814	1,75	**4,73**	***	45 %
	60,00 %	1	288	1814	1,75	**4,68**	***	45 %
	60,83 %	17	453	2641	12,52	**26,81**	***	56 %
	62,50 %	1	333	2580	2,2	**10,55**	***	48 %
	70,00 %	3	468	2510	5,04	**19,27**	***	61 %
18	46,41 %	3	337	2053	4,95	**8,88**	***	72 %
	47,06 %	3	180	1669	4,47	**7,22**	***	67 %
	54,25 %	3	312	2133	3,14	**6,53**	***	75 %
	56,21 %	1	296	2090	1,76	**6,13**	***	76 %
	57,52 %	1	557	2395	2,94	**8,96**	***	75 %
	60,13 %	5	378	2803	7,01	**20,15**	***	75 %
	61,44 %	1	180	1620	1,44	**5,14**	***	75 %
	64,05 %	5	568	4344	11,25	**45,8**	***	75 %
	66,67 %	1	452	3653	4,37	**31,51**	***	81 %
	73,86 %	9	888	3501	18,13	**53,57**	***	79 %
20	38,95 %	11	193	2806	7,12	**17,73**	***	75 %
	41,05 %	13	265	2952	9,7	**23,02**	***	80 %
	46,84 %	1	309	2626	2,09	**7,98**	***	75 %
	47,37 %	3	226	1747	3,32	**7,42**	***	50 %
	48,42 %	13	224	2594	10,54	**22,65**	***	72 %
	50,00 %	17	198	2938	15,13	**34,34**	***	75 %
	53,16 %	9	284	2883	12,04	**26,36**	***	75 %
	53,16 %	7	312	2649	8,87	**19,31**	***	75 %
	54,21 %	11	295	2723	9,29	**26,56**	***	75 %
	68,42 %	1	972	5354	10,32	**103,74**	***	77 %

Tabla 5.1: Comparación de resultados contra CPLEX

Se agregó también al conjunto de instancias de prueba, un grupo de 10 instancias de 14 vértices, con el objetivo de poder ver con más claridad la progresión en los tiempos de ejecución de CPLEX.

Analizando el primer grupo de instancias (para $|V| = 14$), podemos ver que los tiempos de ejecución de ambos métodos son completamente distintos, ya que con el *Branch & Cut* las instancias se resuelven casi inmediatamente mientras que CPLEX precisa en general algunos minutos para ello. Incluso ya en este grupo se puede ver que CPLEX no resuelve algunas de las instancias propuestas.
Una particularidad que aparece en este grupo son las últimas dos instancias, las cuales al parecer representan instancias muy difíciles de resolver en las que ambos métodos obtienen los mismos resultados.

Se puede ver en la tabla que en los siguientes tres grupos de instancias, CPLEX no logra resolver ninguna de las mismas en el límite de tiempo dado, mientras que el *Branch & Cut* sigue resolviendo en forma óptima y en la mayoría de los casos con tiempos de ejecución menores a un minuto. Solamente para la última instancia del último grupo, el tiempo de ejecución supera el minuto, e incluso sigue siendo menor a los dos minutos de tiempo.

CAPÍTULO 6

Conclusiones y trabajo futuro

Se presentan en este capítulo las conclusiones sobre el trabajo realizado. Algunas de éstas fueron ya enunciadas a lo largo del mismo, sin embargo se reunen a continuación. Finalmente, se enumeran los aspectos del trabajo que quedan abiertos para un posible trabajo futuro.

Modelos de programación lineal entera

Se presentaron en este trabajo tres modelos para el *minimum adjacency vertex coloring problem*. Dos de ellos son adaptaciones de formulaciones preexistentes para el problema clásico de coloreo, mientras que el *distance model* es, a nuestro entender, un nuevo enfoque para el modelado de problemas de coloreo de grafos con técnicas de programación lineal entera, y el mismo puede ser fácilmente adaptado al problema de coloreo clásico.
Se realizó con ellos una experimentación preliminar, con el objetivo de evaluar la *performance* un algoritmo *Branch & Bound* puro, para ser utilizado como punto de partida en un estudio teórico del poliedro asociado al mismo. El modelo elegido fue finalmente el *stable model*.
Los resultados preliminares demostraron además, que el problema a analizar es un problema dificil, pues aun en instancias muy pequeñas los tiempos de ejecución eran considerablemente altos e incluso en muchas de las instancias propuestas ningún modelo lograba obtener una solución óptima.

Resultados poliedrales

Se realizó un estudio teórico del poliedro asociado al *stable model* en el cual se hallaron cuatro desigualdades válidas para este poliedro.

Un resultado interesante de este estudio es el hecho de haber podido demostrar que bajo ciertas condiciones, tres de estas desigualdades definen facetas

del poliedro y la restante, la *MCCK inequality*, lo hace en una versión particular de la misma, aunque se conjetura que en su versión general también define facetas del poliedro en cuestión.

Una característica remarcable es el hecho de que todas las desigualdades válidas halladas en este trabajo provienen de ideas y argumentos combinatorios. Esto resulta muy interesante y beneficia en gran medida el análisis de este problema mediante el uso de técnicas de combinatoria poliedral.

Procedimiento *Branch & Cut*

Utilizando las desigualdades halladas en el estudio poliedral se implementó un algoritmo *Branch & Cut*. Para ello se escribieron procedimientos de separación para las cuatro familias de desigualdades halladas y para la adaptación a este problema de una quinta familia preexistente en la literatura.

Los resultados sobre un conjunto de instancias mostraron que la mejor combinación de familias está dada por las *MCCK inequalities* en conjunto con esta quinta familia, a saber las *clique inequalities*.
Las *clique inequalities* no predican sobre las adyacencias de color generadas sino que sólo se limitan a cortar soluciones que representen coloreos inválidos. Por otro lado, las *MCCK inequalities* apuntan a la definición apropiada de la función objetivo imponiendo cotas para estas variables. Resulta un hecho muy interesante que estas dos familias formen una buena combinación, pues apuntan a dos aspectos típicos de un problema de optimización: dominio del problema y función objetivo.

El mejor procedimiento de separación, dentro de los implementados, resultó ser un algoritmo de *backtracking* en lugar de una heurística golosa y las pruebas sobre parámetros tales como el *skip factor* y la cantidad de rondas de cortes a utilizar sugieren que los cortes implementados resultan ser muy efectivos, y en algunos casos decisivos para la resolución del problema.

Finalmente, los tiempos de resolución logrados por el *Branch & Cut* mejoraron drásticamente los tiempos de ejecución en comparación con el *Branch & Bound* realizado en un principio. Además, al comparar los tiempos obtenidos por el *Branch & Cut* con los de las ejecuciones en CPLEX sobre el modelo original se obtuvieron resultados muy interesantes, pudiendo resolverse en pocos segundos instancias que en CPLEX no se podían resolver en un límite de tiempo de 30 minutos.
Por otro lado, la cantidad de subproblemas analizados por el *Branch & Cut* es, en la mayoría de los casos, mínima indicando que muchas veces el problema se resuelve casi puramente con la aplicación de los planos de corte. Este es otro indicador que sugiere que los cortes proporcionados resultan ser muy eficaces.

Una tarea pendiente de esta experimentación, que surge de manera casi

inmediata, es la ejecución del *Branch & Cut* en instancias de tamaño real. Sin embargo, al momento de buscar este tipo de instancias para este problema, identificamos un impedimento no menor relacionado con la cantidad de colores disponibles. Notamos que el tiempo de ejecución de una instancia, por más grande que sea, está fuertemente relacionado a la cantidad de colores disponibles que se utilice, y la variación en este dato puede variar significativamente los mismos. El impedimento entonces, a la hora de experimentar con instancias de tamaño real, fue la elección de este valor. Por este motivo, no se incluyeron, en este trabajo, pruebas del *Branch & Cut* sobre este tipo de instancias.

6.1. Trabajo futuro

Como continuación de este trabajo, queda abierta la posibilidad de encontrar más desigualdades válidas fuertes y agregarlas a la implementación del *Branch & Cut*. Las desigualdades presentadas en este trabajo están basadas específicamente en cliques. Tal vez agregando desigualdades que utilicen otro tipo de estructuras del grafo se puedan obtener mejores resultados. En [16, 17] por ejemplo, se utilizan para el problema clásico de coloreo, estructuras tales como *caminos* y *ciclos inducidos* del grafo.

Un punto interesante a analizar es el nuevo enfoque dado por el *distance model*. En este trabajo se pudo ver que el mismo no logró obtener mejores resultados que los otros dos modelos, sin embargo creemos que es posible que esto se deba al tamaño del modelo tal cual está presentado. Un posible trabajo futuro puede ser realizar un análisis exhaustivo sobre esta formulación con el objetivo de reducir el tamaño del mismo.
Puede ser interesante también adaptar esta formulación al problema clásico de coloreo y experimentar con esta formulación adaptada sobre instancias de prueba de este problema.

Queda también como un problema abierto estudiar la complejidad computacional de los problemas de separación de las desigualdades encontradas, aunque se sospecha que los mismos son NP-completos, puesto que están basados en el problema de clique máxima con relación a alguna función de pesos adecuada.

***

Si bien queda mucho trabajo por hacer con respecto a este problema, esperamos que este trabajo represente un aporte a su resolución y pueda ser utilizado como fuente para futuros trabajos sobre este problema u otros de características similares.

Bibliografía

[1] K. I. Aardal, C. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for the frequency assignment problem. ZIB-report 01–40, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany, 2001.

[2] M. Atamanesh and M. Farzaneh. Frequency planning of GSM cellular communication network in urban areas including traffic distribution, a practical implementation. *Electromagnetic Compatibility and 19th International Zurich Symposium on Electromagnetic Compatibility*, pages 891–894, 2008.

[3] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. The orientation model for frequency assignment problems. Technical Report TR 98-01, ZIB Berlin, 1998.

[4] M. Campêlo, R. Corrêa, and Y. Frota. Cliques, holes and the vertex coloring polytope. *Inf. Process. Lett.*, 89(4):159–164, 2004.

[5] M. Duque-Antón, D. Kunz, and B. Rüber. Channel assignment for cellular radio using simulated annealing. *IEEE Transactions on Vehicular Technology*, 42:14–21, 1993.

[6] A. Eisenblätter. *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2001.

[7] A. Eisenblätter. Assigning frequencies in GSM networks. In U. Leopold-Wildburger, F. Rendl, and G. Wäscher, editors, *Operations Research Proceedings 2002*, pages 33–40. Springer Verlag, 2003. Selected Papers of the International Conference on Operations Research (SOR 2002), Klagenfurt, September 2 - 5, 2002.

[8] A. Eisenblätter, M. Grötschel, and A. Koster. Frequency planning and ramifications of coloring. *Discussiones Mathematicae Graph Theory*, 22(1):51–88, 2002.

[9] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[10] M. Grötschel. Frequency assignment in mobile phone systems. In Kapoor and Prasad, editors, *Lecture Notes in Computer Science*, volume 1974, pages 81–86. FST TCS 2000, 2000.

[11] A. Koster. FAP Web - A Frequency Assignment Problems web site - http://fap.zib.de. 2008.

[12] D. Kunz. Channel assignment for cellular radio using neural networks. *IEEE Transactions on Vehicular Technology*, 40:188–193, 1991.

[13] H. Lee, D. Lee, and J. Lee. Multi-stage neural networks for channel assignment in cellular radio networks. *Lecture Notes in Computer Science*, 3174:287–292, 2004.

[14] F. Luna, E. Alba, A. Nebro, and S. Pedraza. Evolutionary algorithms for real-world instances of the automatic frequency planning problem in GSM networks. In *In 7th European Conf. on Evolutionary Computation in Combinatorial Optimisation, EVOCOP 2007*, 2007.

[15] I. Méndez-Díaz and P. Zabala. A polyhedral approach for graph coloring. *Electronic Notes in Discrete Mathematics*, 7:1–4, April 2001.

[16] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Appl. Math.*, 154(5):826–847, 2006.

[17] I. Méndez-Díaz and P. Zabala. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics*, 156(2):159–179, 2008.

[18] G. Nemhauser and L. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[19] H. Simon. Approximation algorithms for channel assignment in cellular radio networks. In *FCT '89: Proceedings of the International Conference on Fundamentals of Computation Theory*, pages 405–415, London, UK, 1989. Springer-Verlag.

APÉNDICE A

Conceptos básicos

En este apartado se presenta un marco teórico con los conceptos básicos utilizados en esta tesis.

Dentro del marco de la disciplina conocida como *investigación operativa* se encuentra la *optimización combinatoria*. En un problema de optimización se busca el máximo o mínimo de una cierta función, definida sobre algún dominio. A diferencia de la optimización clásica, en la cual el dominio es continuo, en la optimización combinatoria éste es un dominio discreto.

Dado un conjunto finito $\mathcal{S}$ (conjunto de soluciones factibles) y una función $f : \mathcal{S} \to \mathbb{R}$ (función objetivo), un problema de optimización combinatoria consiste en hallar un elemento $\hat{s} \in \mathcal{S}$ tal que

$$f(\hat{s}) = \text{máx}\left\{f(s) : s \in \mathcal{S}\right\}.$$

Salvo en casos particulares, los problemas de optimización combinatoria suelen ser NP-completos ([9]).

A.1. Programación lineal y programación lineal entera

Los problemas de optimización combinatoria están relacionados con dos modelos de optimización: la programación lineal y la programación lineal entera.

Problema de programción lineal: Dada una matriz $A \in \mathbb{R}^{m\times n}$ y vectores $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, encontrar un vector $\hat{x} \in \mathbb{R}^n$ tal que $A\hat{x} \leq b$ y

$$c^t\hat{x} = max\{c^t x : Ax \leq b\}.$$

Este tipo de problemas fue estudiado intensivamente y existen varios métodos para su resolución, siendo el más reconocido de ellos el método *simplex*,

propuesto por Gearge Dantzig en 1947. Si bien este método tiene un peor caso exponencial, es el método más utilizado en la práctica ya que en la mayoría de los casos es muy eficiente. Se sabe además que la programación lineal pertenece a la clase de problemas P.

De todas formas, pocos son los problemas de optimización combinatoria que pueden modelarse en dominios continuos, en particular en los casos en que el problema es NP-completo, y muchas veces es necesario que ciertas variables de la solución tengan dominio entero. Surgen así los problemas de programación lineal entera mixta:

Problema de programción lineal entera mixta: Dada una matriz $A \in \mathbb{R}^{m \times n}$ y vectores $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, y un subconjunto de índices $I \subseteq \{1, \ldots, n\}$, encontrar un vector $\hat{x} \in \mathbb{R}^n$ tal que $A\hat{x} \leq b$ y $\hat{x}_i \in \mathbb{Z}$ para $i \in I$, donde

$$c^t \hat{x} = max\{c^t x : Ax \leq b \text{ y } x_i \in \mathbb{Z} \text{ para } i \in I\}.$$

En el caso en que $I = \{1, \ldots, n\}$, este problema se denomina *problema de programación lineal entera*. Muchos problemas de optimización combinatoria pueden expresarse utilizando este tipo de problemas, representando las soluciones del mismo con los vectores característicos de las soluciones factibles.

A.2. Algoritmos de planos de corte

Se presenta a continuación una introducción a los conceptos básicos de algoritmos de planos de corte. Consideremos un problema de programación lineal entera y llamemos $\mathcal{S}$ al conjunto de puntos factibles:

$$\mathcal{S} = \{x \in \mathbb{R}^n : Ax \leq b \text{ y } x_i \in \mathbb{Z} \text{ para } i \in I\}.$$

Podemos asociar a este problema el poliedro $P_{\mathcal{S}} := conv(\mathcal{S})$, es decir la cápsula convexa de los puntos de $\mathcal{S}$, y así las soluciones factibles del problema serán los extremos de este poliedro. De esta manera, resolviendo el problema lineal máx$\{c^t x : x \in P_{\mathcal{S}}\}$ se resuleve también el problema de optimización combinatoria, dado que el óptimo del problema lineal se encuentra sobre alguno de los extremos de $P_{\mathcal{S}}\}$. Sin embargo, no es posible caracterizar en forma eficiente la cápsula convexa de un conjunto de puntos *definido en forma implícita* que representan las soluciones factibles de un problema NP-completo, a menos que NP = co-NP.

El conjunto $\mathcal{S}' = \{x \in \mathbb{R}^n : Ax \leq b\}$ se denomina la *relajación lineal* del problema, y las soluciones factibles del problema inicial son aquellos puntos $x \in \mathcal{S}'$ con $x_i \in \mathbb{Z}$ para toda $i \in I$. Resolviendo el problema lineal máx$\{c^t x : Ax \leq b\}$ se obtiene un óptimo $\hat{x}$, que puede no ser una solución entera (si lo fuera, se prueba que también es óptimo del problema entero). De no serlo, es posible

hallar una desigualdad $\pi x \leq \pi_0$ que sea *cumplida por todos los puntos enteros* pero tal que $\pi \hat{x} > \pi_0$. Esta desigualdad puede entonces agregarse al conjunto de restricciones, obteniendo un poliedro más pequeño pero que incluye todas las soluciones enteras y ha "dejado afuera" al punto no entero $\hat{x}$. Resolviendo nuevamente la relajación lineal y repitiendo este proceso hasta hallar una solución entera se resuelve el problema. El Algoritmo 7 describe este método.

Algoritmo 7 Planos de corte

Entrada: A, b, c
1: $\hat{x} \leftarrow$ óptimo del problema lineal máx$\{c^t x : Ax \leq b\}$
2: **mientras** existe $i \in I$ tal que $\hat{x}_i$ es no entero **hacer**
3: Hallar una desigualdad válida $\pi x \leq \pi_0$ tal que $\pi \hat{x} > \pi_0$
4: Agregar $\pi x \leq \pi_0$ al conjunto de restricciones $Ax \leq b$
5: $\hat{x} \leftarrow$ nuevo óptimo del problema lineal extendido
6: **fin mientras**
7: **devolver** solución óptima $\hat{x}$

Veremos graficamente cómo funciona este algoritmo con el ejemplo de la Figura A.1. Supongamos que hemos resuelto la relajación lineal, obteniendo el óptimo que se indica con la flecha en la parte (a) de la figura. Dado que esta solución no es entera, se genera un *corte* para la misma, es decir una desigualdad válida, que la deja fuera del poliedro pero que mantiene a todos los puntos enteros dentro del mismo (en la parte (b) se indica con la flecha el corte agregado). Luego se vuelve a resolver la relajación lineal y se obtiene una nueva solución óptima, la cual se puede ver en la parte (c) de la figura. Dado que esta nueva solución tampoco es entera, se vuelve a generar un corte para ella. Se resuelve nuevamente la relajación y en esta ocasión se obtiene un óptimo que coincide con uno de los puntos enteros dentro del poliedro, con lo cual la solución hallada es la solución óptima del problema (en la parte (d) se puede ver este último corte agregado y la solución entera obtenida).

La principal complicación que encuentra este algoritmo es la generación de una desigualdad válida (cumplida por todos los puntos enteros) violada por el óptimo $\hat{x}$ actual. Esto es lo que se conoce como *problema de separación*. Estas desigualdades pueden buscarse de varias maneras, sin embargo las técnicas que han producido mejores resultados se basan en el estudio de los poliedros asociados a cada problema con el objetivo de hallar desigualdades válidas específicas del mismo. El procedimiento habitual consiste en hallar familias de desigualdades válidas tan fuertes como sea posible y definir luego procedimientos de separación para cada una de ellas. Estos procedimientos son en general de carácter heurístico, pues muchas veces el problema de separación resulta ser NP-completo.

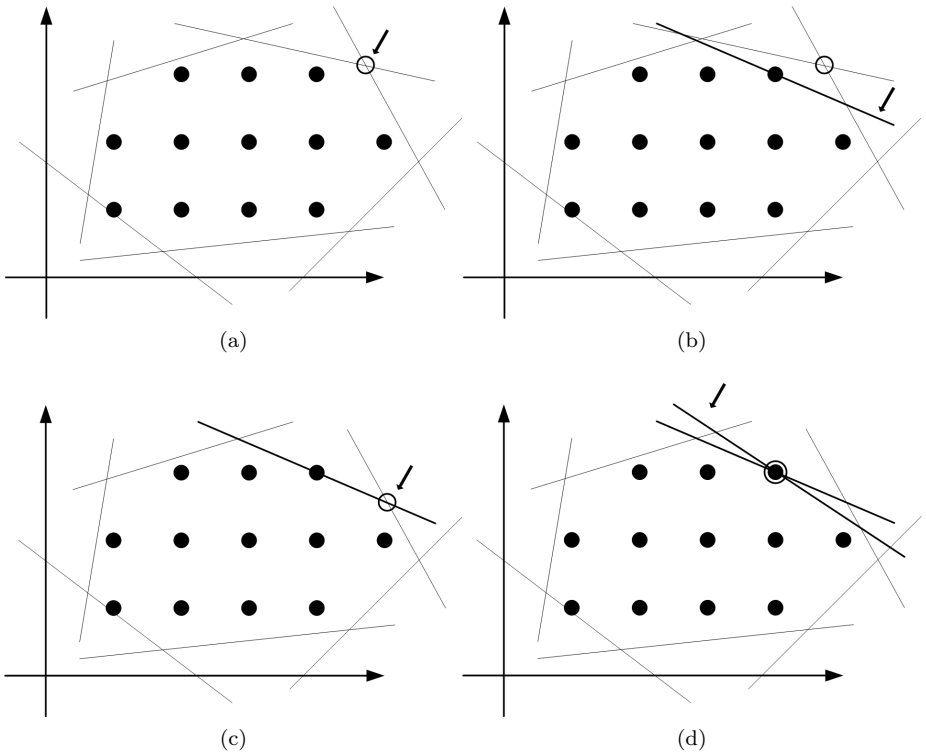


Figura A.1: Ejemplo de planos de corte

Los métodos de resolución basados únicamente en planos de corte no son eficientes en la práctica, sin embargo al combinarse con técnicas de enumeración (*Branch & Bound*) se logran los mejores resultados. Esta combinación recibe el nombre de *Branch & Cut*.

A.3. Algoritmos *Branch & Cut*

Un método *Branch & Bound* constituye un esquema de *divide and conquer*, que intenta resolver el problema original dividiéndolo en subproblemas más pequeños, para los cuales se computan cotas inferiores y superiores. La resolución recursiva de los subproblemas origina una enumeración exhaustiva de todas las soluciones factibles, y es computacionalmente intratable. Manteniendo en forma simultánea los subproblemas aun no resueltos, y calculando cotas inferior y superior para el óptimo, los algoritmos *Branch & Bound* pueden reducir el número de pasos de la enumeración, obteniendo tiempos de ejecución menores.

El método mantiene una lista de subproblemas abiertos y en cada paso

selecciona un elemento de la misma. Se resuelve una relajación de subproblema para obtener una cota superior del óptimo. Si la solución es factible entera, se ha encontrado el óptimo para ese subproblema y por lo tanto se lo elimina de la lista de nodos activos. Si no, se divide el problema en dos o más subproblemas y se incorpora a los mismos a la lista de subproblemas abiertos. En cada paso se actualizan las cotas inferior (mejor solución hallada hasta el momento) y superior. Los componentes básicos del algoritmo son:

Branching: Este proceso de subdivisión puede representarse en forma de un árbol, llamado *árbol de enumeración*, en el cual cada nodo representa un subproblema y sus hijos representan los subproblemas derivados del mismo. Una técnica habitual de *branching* es generar dos nuevos nodos agregando las restricciones $x_i \leq k$ y $x_i \geq k+1$ respectivamente, donde $\hat{x}_i$ es fraccionaria, $i \in I$ y $\lfloor \hat{x}_i \rfloor = k$.

Bounding: Cuando de la relajación de un subproblema se obtiene un valor menor que la cota inferior actual, entonces ninguna de las soluciones factibles del subproblema podrá ser solución óptima y, en este caso, puede eliminarse al subproblema de la lista (se dice que el nodo se "cierra"). Por otro lado, si el óptimo de la relajación es solución factible entera del subproblema, entonces es su óptimo, con lo cual no se lo divide y también se lo cierra.

Utilizando las cotas inferior y superior halladas hasta el momento, se define una medida de calidad de la solución factible actual, conocida como *gap de optimalidad*. Esta medida representa el máximo error relativo que la solución actual puede tener con respecto del óptimo real. El algoritmo *Branch & Bound* termina cuando ambas cotas, inferior y superior, coinciden, en cuyo caso la mejor solución factible hallada hasta ese momento es óptimo del problema original.

El método *Branch & Cut* generaliza al método *Branch & Bound* y agrega la aplicación de planos de cortes a las relajaciones resueltas en cada subproblema. Si luego de resolver la relajación lineal del subproblema no se puede cerrar el nodo correspondiente, se aplica un procedimiento de planos de corte (descripto en la sección anterior). La cantidad de planos de corte aplicados puede ser parametrizada y representa un parámetro muy importante para este tipo de algoritmos.

Se han desarrollado algoritmos basados en este método para un gran número de problemas de optimización combinatoria, logrando resolver instancias de gran tamaño en algunos casos. Hasta la fecha, este enfoque ha resultado el más efectivo para la resolución exacta de esta clase de problemas.