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

El problema de coloreo de aristas por etiquetado total bajo un enfoque de programación lineal entera

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Fabrizio Borghini

Directoras: Isabel Méndez-Díaz, Paula Zabala

Buenos Aires, 2015

EL PROBLEMA DE COLOREO DE ARISTAS POR ETIQUETADO TOTAL BAJO UN ENFOQUE DE PROGRAMACIÓN LINEAL ENTERA

En este trabajo se aborda bajo un enfoque de programación lineal entera el *problema de coloreo de aristas por etiquetado total*. Este concepto fue introducido por S. Brandt, D. Rautenbach y M. Stiebitz en el año 2007.

A lo largo del trabajo se realizan formulaciones de programación lineal entera para modelar el problema y se compara el rendimiento de los modelos mediante distintos criterios. Además, para cada modelo desarrollado, se realiza un estudio poliedral con el propósito de encontrar familias de desigualdades válidas a partir de las cuáles desarrollar un algoritmo de *Branch & Cut*.

Con el objetivo de mejorar la eficiencia en la resolución de instancias, se desarrollan distintas heurísticas iniciales que permiten contar con una solución factible de la cuál partir para optimizar el problema. También se implementa una heurística primal para cada modelo con la intención de encontrar soluciones enteras cercanas al óptimo.

Por último, se presenta los resultados obtenidos sobre diversas instancias de grafos para evaluar la performance de los modelos implementados.

Palabras claves: Etiquetado total, Coloreo de aristas, Programación Lineal Entera, Branch-and-Cut.

ÍNDICE GENERAL

1. INTRODUCCIÓN

El *problema de coloreo de aristas por etiquetado total* ($PCAET$) es un problema de grafos introducido en el año 2007 [1], que consiste en etiquetar los vértices y las aristas de un grafo G con etiquetas $1, 2, \ldots, k$ de manera tal que los colores de las aristas formen un coloreo de aristas de G. En esta definición, el color de una arista de G es determinado como la suma de su etiqueta y de las etiquetas de sus vértices extremos. Se define $\chi'_t(G)$ como el menor entero k tal que G tiene un coloreo de aristas por un k-etiquetado total.

Formalmente, sea un grafo $G = (V, E)$ con V el conjunto de vértices y E el conjunto de aristas. Se denomina *coloreo de aristas por etiquetado total* a la función $f : V \cup E \to \{1, 2, ..., k\}$ tal que los colores de las aristas definidos como:

$$w(uv) = f(u) + f(uv) + f(v)$$

producen un coloreo de aristas, donde $(u, v) \in E$. Esta función f es la que define el etiquetado.

Para entender mejor la definición, en la Fig. 1.1 se muestra a modo de ejemplo un grafo etiquetado en donde no se cumplen las condiciones.

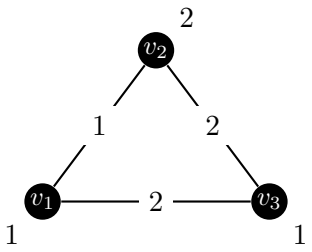

Fig. 1.1: Grafo cuyo etiquetado no define un coloreo de aristas.

Notar que $w(v_1v_2) = w(v_1v_3) = 4$, por lo tanto los colores conformados por el etiquetado no definen un coloreo de aristas.

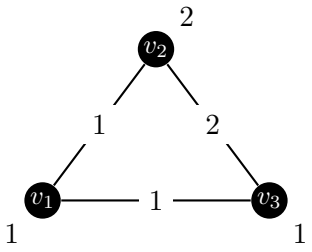

Fig. 1.2: Grafo cuyo etiquetado define un coloreo de aristas.

En el caso de la Fig. 1.2 el etiquetado define un coloreo de aristas. Además se puede verificar que, para este grafo, no existe solución que utilice un valor menor para la etiqueta máxima. Por lo tanto, el etiquetado es óptimo para este grafo.

El *PCAET* se puede considerar una relajación del problema *edge-irregular total labellings* [2] donde la definición del color de una arista es la misma, pero se requiere que todos los colores de las aristas sean diferentes.

En el mismo trabajo donde se presenta el problema [1], S. Brandt, D. Rautenbach y M. Stiebitz muestran cotas inferior y superior de este cromático en términos del grado máximo Δ de G,

$$\left\lceil \frac{\Delta + 1}{2} \right\rceil \leq \chi'_t(G) \leq \Delta$$

No se encontró en la literatura estudios del *PCAET* basados en programación lineal entera. Sin embargo, este problema se analizó bajo un enfoque de teoría de grafos para diferentes clases especiales de grafos. En [1], se llegó a las siguientes resultados:

- Para un grafo cúbico G, $\chi'_t(G) = 2$ si y sólo si el conjunto de vértices de G puede ser particionado en dos partes A y B tal que induzcan *matchings* perfectos.
- Para bosques F con grado máximo Δ, se cumple que $\chi'_t(F) = \lceil \frac{\Delta+1}{2} \rceil$.
- Sea K_n grafo completo de n vértices. Si $n \not\equiv 2 \pmod 4$, entonces $\chi'_t(K_n) = \lceil \frac{n}{2} \rceil$ y si $n \equiv 2 \pmod 4$, entonces $\chi'_t(K_n) \leq \frac{n}{2} + 1$.

Además, se obtuvieron resultados exactos del problema para grafos generalizados de Petersen [3] y se estudió para grafos circulantes 4-regulares [4].

El objetivo de esta tesis es desarrollar un algoritmo exacto para el *PCAET* mediante formulaciones de programación lineal entera. Para lograr esto se proponen dos modelos que abordan el problema de manera diferente y luego se trabaja con cada uno para mejorar los algoritmos.

Se comienza haciendo una introducción a la programación lineal entera, en dónde se describen los conceptos generales y los principales algoritmos utilizados. Luego, se presentan los dos modelos desarrollados para la resolución del problema y sus principales características.

En el siguiente capítulo se hace una comparación entre los modelos propuestos. No sólo sirve para comparar el desempeño de los modelos, sino también establece una base de trabajo para mejorar los resultados obtenidos.

Una vez hecha la comparación, se destina un capítulo al desarrollo de distintas heurísticas para conseguir soluciones factibles para el problema. En dicho capítulo se proponen tres heurísticas iniciales. La primera heurística presentada logra obtener un coloreo por un Δ-etiquetado total. Las siguientes dos heurísticas buscan obtener una solución por un k-etiquetado total con k menor a Δ. También, se presenta una heurística primal para cada modelo.

Luego, se busca mejorar los algoritmos para cada modelo mediante el estudio poliedral de las formulaciones. Se identifican y analizan distintas desigualdades válidas.

En el siguiente capítulo se presentan los resultados obtenidos sobre varias instancias para poder comparar el rendimiento del algoritmo trabajado contra el algoritmo original.

El capítulo siguiente se destina a exponer las conclusiones que se pudieron extraer a lo largo del trabajo y algunos puntos en los que se puede seguir trabajando.

Por último, se dedica un apéndice para mostrar que las desigualdades válidas presentadas para los modelos son efectivamente planos de corte y que consituyen familias independientes entre ellas, es decir, que una familia no es implicada por las otras.

2. PROGRAMACIÓN LINEAL ENTERA

Un problema de programación lineal consiste en minimizar o maximizar una función lineal restringida a determinandas desigualdades o igualdades lineales. La función lineal que se quiere optimizar se denomina función objetivo. En este tipo de problemas, las variables que se usan en la función objetivo y restricciones para modelar el problema toman valores reales.

En este trabajo nos vamos a enfocar en la programación lineal entera (PLE), en donde las variables utilizadas en la formulación se restringen a que sólo puedan tomar valores enteros. También existe la programación lineal entera mixta (PLEM) que es una generalización de las anteriores. En una formulación PLEM pueden existir variables que tomen valores reales y variables restringidas a tomar valores enteros.

Con PLE se pueden modelar diversos problemas de optimización, como por ejemplo, problemas de planificación de producción, de redes de telecomunicaciones, asignación de tareas, etc. La PLE es `NP-hard`, es decir, no se conocen algoritmos polinomiales que lo resuelvan.

En las siguientes secciones se describen los principales conceptos de la PLE y las técnicas generales que se utilizan para resolver este tipo de problemas. El objetivo de estas secciones es entender los pasos básicos que se siguieron en el trabajo para estudiar el problema presentado bajo un enfoque de programación lineal entera. Para más detalle en los temas que se introducen en este capítulo, ver [5].

2.1. Formulación

Como se mencionó, el objetivo de un problema de programación lineal entera es optimizar una función objetivo sujeta a determinado número finito de restricciones lineales. Entonces, un problema con n variables y m restricciones se puede formular de la siguiente forma (en este caso, la intención es minimizar la función objetivo):

$$
\begin{aligned}
\text{Minimizar} \quad & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{sujeto a} \quad & \\
& a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\
& \quad \vdots \qquad \vdots \qquad\qquad \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\
& x_1, x_2, \cdots, x_n \in \mathbb{Z}_{\geq 0}
\end{aligned}
$$

Entonces:

- $c_1x_1 + c_2x_2 + \cdots + c_nx_n$ es la función objetivo, en donde $c_1, c_2, \ldots, c_n$ son coeficientes de la función y $x_1, x_2, \ldots, x_n$ son las variables del modelo.

- $a_{i1}x_1 + a_{i2}x_2 \cdots + a_{in}x_n \leq b_i$, con $i = 1, \ldots, m$ denota la i-ésima restricción del modelo.

- $x_1, x_2, \ldots, x_n \in \mathbb{Z}_{\geq 0}$ representa la restricción de que las variables sean enteras positivas.

Otra manera usual de mostrar una formulación de PLE es usando notación matricial. Consideremos el vector $c = (c_1, c_2, \ldots, c_n) \in \mathbb{R}^{1\times n}$, los vectores $b \in \mathbb{R}^m$, $x \in \mathbb{Z}^n_{\geq 0}$ y la matriz $A \in \mathbb{R}^{m\times n}$.

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Luego, la formulación del problema queda de la siguiente manera:

$$\begin{aligned} \text{Minimizar} \quad & cx \\ \text{sujeto a} \quad & \\ & Ax \leq b \\ & x \in \mathbb{Z}^n_{\geq 0} \end{aligned}$$

Un vector $x = (x_1, x_2, \ldots, x_n) \in Z^n_{\geq 0}$ que cumple con todas las restricciones de la formulación se denomina *punto factible*. El conjunto de todos los puntos factibles constituye la *región factible*. Informalmente se puede decir que la programación lineal (y la PLE) tiene como objetivo encontrar de entre todos los puntos factibles, aquel que optimice la función objetivo (la minimice o maximice, según lo que necesite el problema).

Toda formulación de PLE tiene asociado un subconjunto de $\mathbb{R}^n$ definido por las restricciones lineales, el *poliedro* $P = \{x \in \mathbb{R}^n_{\geq 0} : Ax \leq b\}$. Este poliedro se denomina relajación lineal del PLE. El conjunto de soluciones factibles del problema es $S = P \cap \mathbb{Z}^n$, ya que buscamos que las variables sean enteras.

En la Fig. 2.1 se puede ver el poliedro P asociado a una formulación que consta de dos variables x_1, x_2.

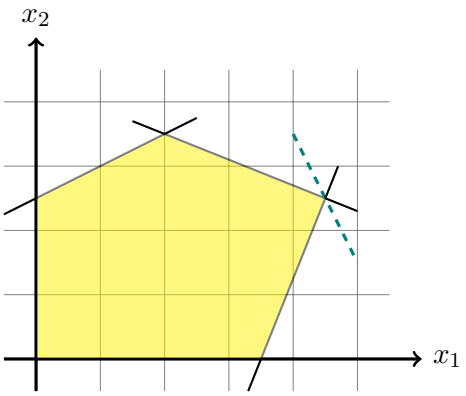

Fig. 2.1: Región factible del poliedro asociado P (relajación lineal).

La región sombreada con amarillo es el poliedro asociado a la formulación. El poliedro queda definido por la intersección de las restricciones de la formulación y la orientación de las propias restricciones. El conjunto de soluciones son los puntos que pertenecen a P (en general, infinitos). La recta celeste punteada del gráfico corresponde a la función objetivo. Como es una función lineal, queda representada por un hiperplano en el espacio. En general, la función objetivo es un hiperplano con pendiente definida que toma diferentes valores según dónde esté ubicada en el espacio. Si la función objetivo es cx, la meta es encontrar el valor z tal que el hiperplano $cx = z$ pase por el punto de la región factible que mejor solución le haga tomar.

En programación lineal cada punto del poliedro es una solución factible del problema ya que no se restringe a que las variables tomen valores enteros. Además, se demostró que en los poliedros acotados se puede encontrar una solución óptima en uno de los extremos o vértices del poliedro. De esta manera, un algoritmo posible para resolver problemas usando programación lineal es enumerar todos los vértices del poliedro y quedarse con aquel que optimice la función objetivo. Sin embargo, la cantidad de vértices del poliedro puede llegar a ser exponencial, de manera que no es un método eficiente para obtener la solución óptima. Basado en esta idea, Dantzing [6] desarrolló el algoritmo simplex, que si bien en el peor caso tiene un comportamiento no polinomial, en la práctica se comporta como un algoritmo muy eficiente.

En el caso de PLE, el conjunto de soluciones se corresponde con los puntos que están dentro del poliedro, pero cuyas coordenadas toman valores enteros. En la Fig. 2.2 se muestra el mismo poliedro que la formulación anterior, pero restringiendo las variables a valores enteros.

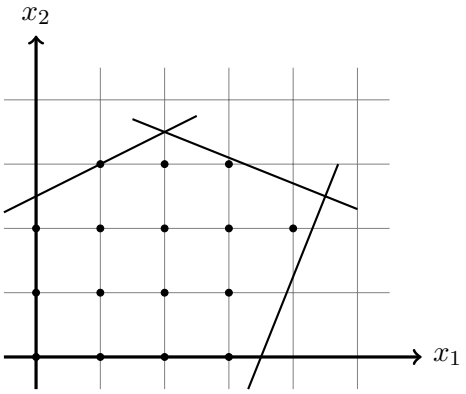

Fig. 2.2: Conjunto S de soluciones.

En el gráfico, las soluciones factibles son las puntos marcados en negro dentro del poliedro. Es decir, son todos los puntos pertenecientes a S.

Un problema de PLE puede ser definido en función de la cápsula convexa de S, $conv(S)$, el menor poliedro que contiene a S (Fig. 2.3). Como todos los puntos extremos de $conv(S)$ pertenecen a S, luego PLE equivale a resolver $opt = \{cx : x \in conv(S)\}$ (donde $opt = min$ en el caso de una minimización, y $opt = max$ en el caso de una maximización), es decir el problema puede ser resuelto como un problema de programacion lineal.

Si se pudiera caracterizar $conv(S)$ con un número polinomial, en la cantidad de variables y de desigualdades lineales, se podría resolver el problema como si fuera un problema de programación lineal con el método simplex. Sin embargo, para la mayoría de los problemas no se ha encontrado una caracterización completa de cápsula convexa. En muchos casos, se necesita una cantidad exponencial de desigualdades lineales para describir la cápsula convexa lo que hace poco probable su resolución mediante los métodos conocidos de programación lineal.

Traducir la descripción de un problema en una formulación de PLE (y de programación lineal en general) no es un tema menor. Se deben identificar las variables que parezcan necesarias para el problema y, usando estas variables, definir un conjunto de restricciones de tal manera que los puntos factibles correspondan a soluciones factibles del problema y la función objetivo. Entonces, existen diferentes formulaciones para un mismo problema. En principio, se puede describir el mismo problema usando diferentes conjuntos de variables (en tanto se definan apropiadamente las restricciones y la función objetivo). Además, se puede dar el caso que, con las mismas variables, existan muchas maneras de modelar el problema utilizando diferentes conjuntos de restricciones.

En el caso que las formulaciones tengan diferentes conjuntos de variables, probablemente los poliedros asociados pertenezcan a espacios de dimensiones diferentes. Pero si se cuenta con formulaciones que utilizan las mismas variables, aunque los poliedros asociados sean diferentes, pertenecen a un mismo espacio.

Supongamos dos formulaciones F_1 y F_2, cuyas relajaciones lineales asociadas son los poliedros P_1 y P_2 que pertenecen a la misma dimensión. Se dice que F_1 es mejor formulación que F_2, si $P_1 \subset P_2$ y $P_2 \setminus P_1 \neq \emptyset$.

Siguiendo este concepto, como se mencionó anteriormente, la mejor formulación que se puede conseguir de un problema es justamente cuando la relajación lineal coincide con

la cápsula convexa. En la Fig. 2.3 se muestra la primera formulación presentada, junto a la cápsula convexa de S.

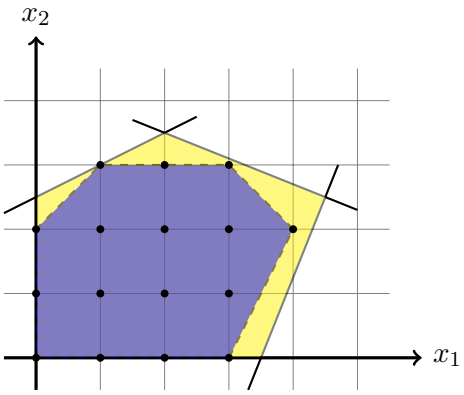

Fig. 2.3: Cápsula convexa de S.

La región azul encerrada por las líneas punteadas corresponde a la cápsula convexa. La relajación lineal de la primera formulación incluye la cápsula convexa y las regiones de color amarillo. Los hiperplanos que definen la cápsula convexa de un problema se denominan *facetas*. En general, resulta beneficioso en problemas de PLE encontrar formulaciones lo más ajustadas posibles. Es decir, que las restricciones se 'aproximen' lo más posible a las *facetas*. Esta tarea suele ser difícil en la mayoría de los problemas que surgen en la práctica.

2.2. Algoritmos para PLE

En esta sección se describen los algoritmos más usuales para resolver problemas de PLE. La forma más simple para resolver un problema de PLE es enumerar todos las soluciones posibles y quedarse con la mejor para el problema. Este procedimiento puede servir solamente para instancias pequeñas del problema. Para instancias mayores, la gran cantidad de soluciones posibles hacen que este procedimiento no sea eficiente.

Los algoritmos más utilizados para resolver los problemas de PLE (y en particular, los utilizados en este trabajo) se enmarcan dentro de alguno de los siguientes esquemas:

- algoritmos *Branch & Bound*.
- algoritmos de *Planos de corte*.
- algoritmos *Branch & Cut*.

A continuación se explican las características principales de cada algoritmo. Para las siguientes secciones, se considera la formulación PLE con poliedro correspondiente a la relajación lineal asociada $P = \{x \in \mathbb{R}^n_{\geq 0} : Ax \leq b\}$ y función objetivo cx. Recordar que la meta es encontrar la mejor solución del conjunto $S = P \cap \mathbb{Z}^n$ en términos de la función objetivo. Entonces, el problema consiste en encontrar $z = min\{cx : x \in S\}$.

2.2.1. Algoritmos *Branch & Bound*

Una idea que surge naturalmente es partir el problema en subproblemas que sean más fáciles de resolver para luego reunir la información con el objetivo de resolver el problema original. Considerando esto, se cuenta con la siguiente propiedad: si $S = S_1 \cup \cdots \cup S_R$ es una descomposición de S en conjuntos más pequeños y se consideran los problemas $z^r = min\{cx : x \in S_r\}$ con $r = 1, \ldots, R$, entonces $z = min_r\ z^r$. Es decir, se puede separar el espacio de soluciones en conjuntos más pequeños para hacer la búsqueda más fácil. Esta idea está asociada a un enfoque *Divide & Conquer*.

Sin embargo, no es posible enumerar todas las soluciones ya que, como vimos, el espacio de soluciones de un problema puede ser enorme. Entonces, el propósito de un algoritmo *Branch & Bound* es realizar una enumeración inteligente de las soluciones factibles, utilizando información relacionada al problema para reducir el espacio de búsqueda. *Branching* hace alusión a la división en subproblemas y *bounding* se refiere al proceso de poda de posibles soluciones.

Este esquema se puede representar mediante un árbol cuya raíz corresponde al problema original y las ramas son el resultado de partir el espacio de búsqueda. Cada nodo del árbol se corresponde con un subproblema que consiste en buscar el óptimo dentro de una parte del espacio de soluciones. Además, argumentos de dominancia y factibilidad permitirán podar ramas del árbol en el proceso de búsqueda.

Una posibilidad para descartar ramas del árbol, denominada *bounding*, consiste en calcular en cada nodo del árbol cotas inferiores del óptimo restringido al espacio de soluciones que considera dicho nodo. De esta manera, si el valor de la cota inferior que se obtuvo es mayor que la cota superior que se conoce hasta el momento para el problema, entonces no es necesario seguir explorando esa rama. El procedimiento para encontrar estas cotas debe lograr un equilibrio entre calidad y esfuerzo para obtenerlas. Una cota débil puede devenir en exploración innecesaria de ramas, pero un procedimiento que calcule cotas ajustadas a un alto costo puede perjudicar el rendimiento del algoritmo en su totalidad.

Una forma usual para obtener cotas inferiores consiste en resolver una relajación del problema. La intención de calcular la relajación es reemplazar el problema de PLE por un problema que sea más fácil de resolver y que provea una solución que sea menor o igual que el óptimo del problema original. Es deseable obtener buenas relajaciones. Generalmente, la diferencia entre el valor óptimo de la relajación y el valor óptimo del problema (*gap*) es una medida usada para evaluar la calidad de la relajación. Las opciones más utilizadas en la práctica para obtener cotas son la relajación lineal y la relajación lagrangeana. Como ya lo mencionamos, el objetivo de la relajación lineal es agrandar el espacio de soluciones factibles sobre el cual se optimiza permitiendo que las variables tomen valores continuos. La relajación lagrangeana busca una función objetivo que tenga igual o menor óptimo.

La relajación lineal puede ser útil también para probar optimalidad del problema. Si la relajación lineal resulta ser infactible, entonces el problema original también lo es. Si el óptimo de la relajación lineal cumple las restricciones de integralidad, entonces también es la solución óptima del problema original.

El proceso de *branching* consiste en dividir la región factible del nodo que está siendo resuelto en dos o más regiones factibles más pequeñas. Cada nueva región producto de la división, genera un nodo hijo (es decir, un nuevo subproblema) en el árbol en donde se agregan nuevas restricciones al problema asociado al nodo padre. Un requerimiento fundamental es que cada solución factible del nodo padre pertenezca por lo menos a

alguno de sus hijos. Estos nuevos nodos creados son agregados a la lista de nodos no explorados. La regla más sencilla de *branching* es considerar una variable entera x_i que tenga un valor fraccionario x_i^* en la actual solución de la relajación. A partir de ella, se desprenden dos subproblemas que dejan fuera la posibilidad de asignar x_i^* nuevamente a x_i. El primer subproblema es idéntico al problema asociado al nodo padre, con el agregado de la restricción $x_i \leq \lfloor x_i^* \rfloor$. En el otro subproblema se agrega la restricción $x_i \geq \lceil x_i^* \rceil$. Este procedimiento se aplica recursivamente a cada nodo del árbol.

En la Fig. 2.4 se muestra una división en dos partes del poliedro de la formulación original. La división consiste en tomar para un conjunto de soluciones que la variable $x_1 \leq 1$ y para el otro conjunto, $x_1 \geq 2$. Notar que la partición cumple con que todos las soluciones enteras factibles pertenecen a alguno de los dos subconjuntos.

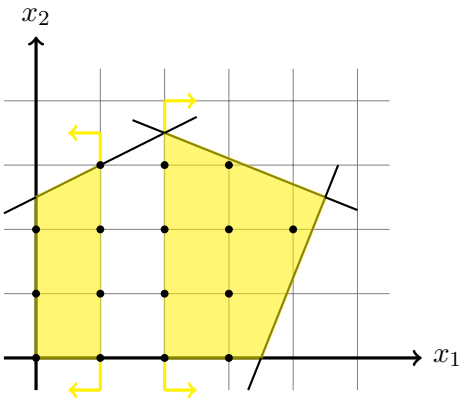

Fig. 2.4: Proceso de *branching*.

Otro factor importante en el algoritmo de *Branch & Bound* es el criterio para elegir el siguiente nodo a explorar. Entre las principales estrategias se encuentran *búsqueda en profundidad*, *búsqueda en amplitud* y *mejor cota primero*. Con *búsqueda en profundidad* se trata de encontrar soluciones factibles rápidamente. *Búsqueda en amplitud* resuelve los nodos recién creados. Por último, el criterio de *mejor cota primero* propone elegir el nodo activo con cota más chica de la relajación.

No existe una combinación de estos factores que produzcan siempre mejores resultados para todos los problemas. Es necesario utilizar criterios basados en una combinación de teoría, sentido común y experimentación.

El siguiente pseudocódigo muestra el esquema general de un algoritmo *Branch & Bound*. El algoritmo recibe la formulación PLE que se quiere optimizar. El conjunto $\mathcal{N}$ contiene los nodos activos del árbol (al principio, se lo inicializa con el problema original, que es la raíz del árbol). Para cada nodo k del conjunto, x^k es la solución óptima de la relajación lineal y z^k el valor de la función objetivo. La mejor solución obtenida se actualiza en x^*, con valor de función objetivo z^*.

Algoritmo 1 Esquema general de algoritmo *Branch & Bound*
Entrada: una formulación de PLE.

1: **Inicialización**
$\mathcal{N} = \{\text{PLE}\}$, $z^* = \infty$

2: **Elección de próximo nodo**
Si $\mathcal{N} = \emptyset$:
- Si $z^* = \infty$, PLE es infactible.
- Si $z^* < \infty, z^*$ es óptimo, con solución x^*.

Si $\mathcal{N} \neq \emptyset$:
- Elegir y borrar un nodo k de $\mathcal{N}$.

3: **Evaluación**
Resolver la relajación lineal del nodo k
(a) Si no es factible, ir a **Elección de próximo nodo**.
(b) Si es factible y $z^k \geq z^*$, ir a **Elección de próximo nodo**.
(c) Si es factible y x^k es entero, actualizar $z^* = min\{z^*, z^k\}$ y $x^* = x^k$ e ir a **Elección de próximo nodo**.

4: **División**
Dividir la región factible del nodo k en dos o más regiones y agregar a $\mathcal{N}$ un nuevo nodo por cada región. Ir a **Elección de próximo nodo**.

En el algoritmo se puede ver dónde se realizan los procesos que se explicaron. En el punto **Evaluación** se hace *bounding*: si $z^k \geq z^*$, entonces en esa rama no puede existir una solución factible mejor de la obtenida hasta el momento. El *branching* se hace en el punto **División** al partir la región factible en regiones más pequeñas.

2.2.2. Algoritmos de planos de corte

El propósito de un algoritmo de planos de corte es ir mejorando la relajación lineal mediante la adición de desigualdades lineales válidas que eliminen (corten) soluciones de la relajación. Una desigualdad lineal $\pi x \leq \pi_0$ es *desigualdad válida* para S si $\pi\bar{x} \leq \pi_0, \forall \bar{x} \in S$. Es decir, se busca identificar desigualdades que satisfagan todos los puntos enteros factibles y además corten soluciones de la relajación que no pertenezcan a $conv(S)$. El objetivo es separar el óptimo fraccionario del conjunto de soluciones enteras factibles S (con un algoritmo de separación). Identificar y agregar desigualdades válidas a la formulación permite contar con relajaciones más ajustadas para el problema.

El éxito de la métodología depende en gran medida de la posibilidad y eficiencia de encontrar *planos de corte* (desigualdades lineales válidas violadas por el óptimo de la relajación) que puedan ser agregadas a la formulación para *separar* soluciones fraccionarias.

A continuación se presenta el esquema general de la metodología. Aquí, PLE es la formulación del problema que se quiere resolver, x^* es el óptimo de la relajación lineal y z^* el valor de la función objetivo evaluada en x^*.

Algoritmo 2 Esquema general de algoritmo de *planos de corte*

Entrada: una formulación de PLE.

1: **Inicialización**
P = PLE

2: **Evaluación**
Resolver la relajación lineal
(a) Si no es factible, entonces PLE no es factible y el algoritmo termina.
(b) Si x^* es entero, entonces x^* es el óptimo del problema y el algoritmo termina.
(c) **Separación**: encontrar una desigualdades válidas violadas por x^*.
- Si se encuentran cortes, agregarlos a P e ir a **Evaluación**.
- Si no se encuentran, z^* es cota inferior de z. El algoritmo termina.

Si resulta que el problema es factible, se repite el procedimiento hasta que se encuentra el óptimo entero del problema o no se identifican más desigualdades válidas que separen el óptimo de la relajación actual.

Para entender mejor la metodología, la Fig. 2.5 muestra un *plano de corte* para la formulación original y el óptimo de la relajación lineal. Se ve cómo el *plano de corte* 'deja afuera' la solución fraccionaria de la relajación.

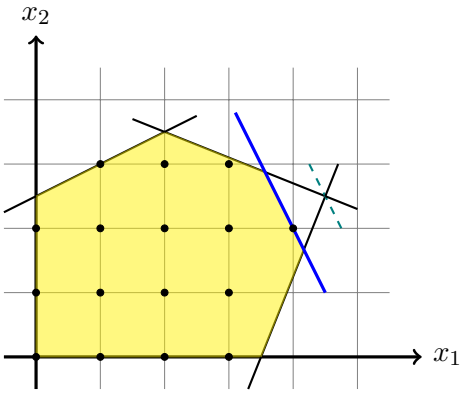

Fig. 2.5: Plano de corte.

En la Fig. 2.5, el *plano de corte* es la línea azul y la recta punteada celeste corresponde a la función objetivo evaluada en el óptimo de la relajación lineal. Se puede observar cómo la desigualdad válida reduce el poliedro. Separa la solución óptima fraccionaria de la relajación, pero no 'corta' ninguna solución entera factible de S. De esta manera, al volver a resolver la relajación lineal, se encontrará un óptimo diferente.

Los planos de corte pueden ser generados bajo dos enfoques:

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

  En los años 60, Gomory desarrolló un algoritmo general para generar desigualdades válidas que separan la solución óptima de la relajación lineal. Las desigualdades se

obtienen en función de la expresión de las variables básicas en funcion de las no básicas, usando únicamente argumentos de integralidad. Además, se demostró que bajo ciertas condiciones, converge a la solución óptima del problema original. Sin embargo, tiene la desventaja de que la implementación computacional es númericamente inestable y no parece ofrecer un buen rendimiento en la práctica, si bien hay algunas implementaciones relativamente eficientes como la que tiene el CPLEX [7].

Existen otra variedad de algoritmos que generan planos de cortes para cualquier problema de programación lineal entera, como por ejemplo los cortes *disyuntivos, clique, cover,* etc. Si bien estos algoritmos poseen propiedades interesantes en el plano teórico, su éxito en la práctica es discutible. Pueden ser efectivos para determinado tipo de problemas, pero para otros no lo son. Este tipo de algoritmos tiene la gran ventaja que pueden ser utilizados para cualquier problema de programación lineal entera. Sin embargo, en muchos casos, resulta conveniente analizar la estructura específica del problema para llegar a procedimientos con mejor rendimiento. Este es el sentido del próximo enfoque.

- *Explotando la estructura particular del problema*

  Como se mencionó, hay datos y propiedades particulares a cada problema que pueden servir para identificar planos de corte. El estudio poliedral consiste justamente en esto, analizar la estructura propia del problema para encontrar planos de corte. Con esta técnica se pretende describir lo mejor posible la cápsula convexa de S. Esta herramienta resultó ser de las más efectivas en la práctica. Recordemos que si se obtiene una caracterización completa de la cápsula convexa de S, se puede resolver eficientemente el problema mediante un algoritmo de programación lineal.

  Una propiedad deseada para un plano de corte es que separe la mayor cantidad de soluciones fraccionarias. Por lo tanto, los mejores planos de corte que se pueden identificar son las *facetas* (desigualdades lineales que describen $conv(S)$). Sin embargo, encontrar *facetas* de $conv(S)$ puede resultar una tarea difícil y la mayoría de los problemas no suelen tener una cantidad polinomial de *facetas.*

  Con fines algorítmicos, el estudio poliedral debe estar acompañado de algoritmos de separación eficientes. En este sentido, hay un resultado muy importante gracias a Grötschel, Lovász y Schrijver [8], que relaciona la complejidad del problema de separación con la complejidad del problema de optimización. El problema de optimización $max\{cx : x \in conv(S)\}$ puede resolverse polinomialmente si y sólo si el problema de separación ($x \in conv(S)$ *ó encontrar una desigualdad válida violada*) es polinomial. Esto indica que si el problema que estamos analizando es `NP-hard`, debe existir por lo menos alguna familia de facetas cuya separación es `NP-hard`.

En el desarrollo de un algoritmo de planos de corte, el primer objetivo es identificar desigualdades lineales que corten la solución de la relajación actual y sea satisfecha por todos los puntos enteros factibles. Esto es lo que se denomima problema de separacion. En algunos casos, los algoritmos de separación pueden tener una complejidad alta y será necesario desarrollar procedimientos heurísticos. Existe la posibilidad de que estos procedimientos no sean capaces de encontrar una desigualdad violada aunque exista.

Como conclusión, es posible que un algoritmo de planos de corte no encuentre la solución óptima para el problema, debido a que no logra encontrar más desigualdades

violadas o se excede el límite establecido para la ejecución. Sin embargo, puede resultar muy útil para obtener buenas cotas inferiores del valor óptimo del problema.

2.2.3. Algoritmos *Branch & Cut*

La resolución de problemas difíciles de programación lineal entera puede requerir la combinación de diferentes técnicas desarrolladas. En este sentido, los algoritmos *Branch & Cut* proponen combinar los dos técnicas presentadas anteriormente. La idea principal del algoritmo consiste en generar planos de corte a lo largo del árbol de enumeración del *Branch & Bound*. En lugar de obtener rápidamente una cota en cada nodo del árbol, se busca trabajar más el nodo mediante la adición de planos de corte para lograr cotas más ajustadas. En general, al agregar desigualdades válidas a la formulación del subproblema asociado al nodo se obtienen mejores cotas que las que produciría un *Branch & Bound*. Esta metodología empezó a aplicarse a comienzos de los años 80 y permitió resolver exitosamente instancias de tamaño considerable de varios problemas de optimización.

El esquema básico de un algoritmo de *Branch & Cut* es aplicar el método *Branch & Bound*, generando planos de corte en los nodos del árbol. No sólo se busca reducir la cantidad de nodos del árbol, sino también fortalecer la formulación agregando planos de corte. Para la implementación de un algoritmo *Branch & Cut* se debe tener en cuenta las estrategias propias de un algoritmo de *Branch & Bound*, en conjunto con las de un algoritmo de planos de corte.

A continuación se presenta el esquema del algoritmo. Recordar que $\mathcal{N}$ almacena los nodos activos del árbol y para cada nodo k, x^k es la solución óptima de la relajación lineal y z^k el valor de la función objetivo. En x^* se mantiene la mejor solución obtenida al momento y en z^*, el valor de la función objetivo.

Algoritmo 3 Esquema general de algoritmo *Branch & Cut*

Entrada: una formulación de PLE.

1: **Inicialización**
$\mathcal{N} = \{\text{PLE}\}$, $z^* = \infty$

2: **Elección de próximo nodo**
Si $\mathcal{N} = \emptyset$:
- Si $z^* = \infty$, PLE es infactible.
- Si $z^* < \infty, z^*$ es óptimo, con solución x^*.

Si $\mathcal{N} \neq \emptyset$:
- Elegir y borrar un nodo k de $\mathcal{N}$.

3: **Evaluación**
Resolver la relajación lineal del nodo k
(a) Si no es factible, ir a **Elección de próximo nodo**.
(b) Si es factible y $z^k \geq z^*$, ir a **Elección de próximo nodo**.
(c) Si es factible y x^k es entero, actualizar $z^* = min\{z^*, z^k\}$ y $x^* = x^k$ e ir a **Elección de próximo nodo**.

4: **División vs. Separación**
- Si se decide buscar planos de corte, ir a **Separación**.
- Si se decide no buscar planos de corte, ir a **División**.

5: **Separación**
Resolver el problema de separación para la solución de la relajación lineal de k.
- Si se encuentran desigualdades violadas, agregarlos e ir a **Evaluación**.
- Si no se encuentran, ir a División.

6: **División**
Dividir la región factible del nodo k en dos o más regiones y agregar a $\mathcal{N}$ un nuevo nodo por cada región. Ir a **Elección de próximo nodo**.

En el algoritmo se puede notar que se suman nuevas decisiones, por ejemplo: cuántas iteraciones se realizan para buscar planos de corte, cuántos planos de corte se agregan en cada iteración, etc. El rendimiento del algoritmo depende de estos factores y de muchos otros. En la práctica, no es tarea sencilla lograr un equilibrio de ellos y depende mucho del problema que se quiere resolver.

Siguiendo con la formulación original, se muestra una ejemplo en dónde se divide el problema en dos subproblemas, y en uno de ellos se aplica un plano de corte.

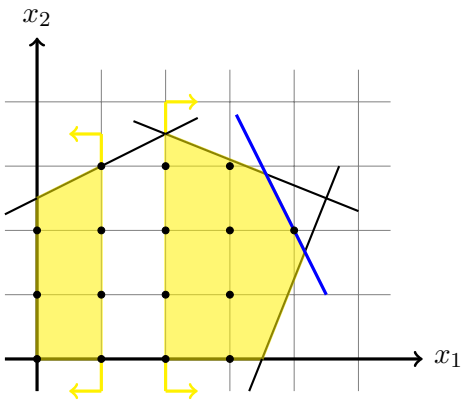

Fig. 2.6: Plano de corte en un subproblema.

En la Fig. 2.6 se observa un plano de corte (la recta de color azul) agregado a la formulación del subproblema correspondiente a la rama $x_1 \geq 2$ del *Branch & Bound*.

3. MODELOS PROPUESTOS

En este capítulo se presentan los modelos de programación lineal entera para resolver el PCAET. Para cada modelo se seguirá el siguiente esquema: se hará una descripción del modelo, se definirán las variables, luego la formulación y por último se dará una explicación de la formulación.

Para este apartado, se considera un grafo $G = (V, E)$ de grado máximo $\Delta = \Delta(G)$.

3.1. Modelo 1

En el primer modelo se considera una variable entera general por cada vértice y arista del grafo que indica el valor de la etiqueta que le es asignada. De esta manera, se obtiene un etiquetado total del grafo.

Sin embargo, queda por resolver el modelado del color que queda definido para las aristas, para lograr un coloreo propio. Para lograr esto, se usan variables adicionales δ que sirven para modelar el hecho de que dos aristas que inciden en un mismo vértice deben recibir un color diferente.

Por último, se define una variable utilizada para minimizar el valor de las etiquetas.

3.1.1. Variables

Las variables definidas para el modelo son las siguientes:

- $z \in \{1, 2, \ldots, \Delta\}$

  Esta variable es la que se va minimizar en la función objetivo. En otras palabras, será la que represente el valor del cromático $\chi'_t(G)$.

- $x_v \in \{1, 2, \ldots, \Delta\}$

  Esta variable representa la etiqueta que le fue asignada al vértice $v \in V$. Es decir, si $x_v = k$, entonces $f(v) = k$.

- $x_{uv} \in \{1, 2, \ldots, \Delta\}$

  Esta variable representa la etiqueta que le fue asignada a la arista $(u, v) \in E$. Es decir, si $x_{uv} = k$, entonces $f(uv) = k$.

- $\delta_{ujv} = \begin{cases} 1 & \text{si } w(uj) < w(jv) \\ 0 & \text{si } w(uj) > w(jv) \end{cases}$

  Esta variable servirá para expresar que $w(uj) \neq w(jv)$, de tal manera que los colores que reciben las aristas definan un coloreo. Cabe destacar que dada una terna u, j, v de vértices tal que las aristas $(u, j), (j, v)$ pertenecen al grafo, se cuenta con una única variable δ definida según el orden de lectura de las aristas. En otras palabras, si se define la variable δ_{ujv} en el modelo, entonces no se define δ_{vju}.

Las variables sólo pueden tomar valores enteros. Recordar que está demostrado que $\chi'_t(G) \leq \Delta$, por lo tanto, es posible acotar el valor de las etiquetas, tanto de vértices como de aristas, por Δ.

3.1.2. Formulación

A continuación, se presenta la formulación del modelo, en donde se definen las restricciones lineales y la función objetivo.

minimizar	z		
sujeto a			
(1)	$x_w \leq z$		$\forall w \in V$
(2)	$x_{uv} \leq z$		$\forall (u,v) \in E$
(3)	$x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1)$		$\forall (u,j),(j,v) \in E$ y $u \neq v$
(4)	$x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1)$		$\forall (u,j),(j,v) \in E$ y $u \neq v$
(5)	$1 \leq x_w \leq \Delta,$ con $x_w \in \mathbb{Z}$		$\forall w \in V$
(6)	$1 \leq x_{uv} \leq \Delta,$ con $x_{uv} \in \mathbb{Z}$		$\forall (u,v) \in E$
(7)	$1 \leq z \leq \Delta,$ con $z \in \mathbb{Z}$		
(8)	$\delta_{ujv} \in \{0,1\}$		$\forall (u,j),(j,v) \in E$ y $u \neq v$

Fig. 3.1: Formulación del modelo 1.

Como se mecionó, la variable z representará el valor del cromático $\chi'_t(G)$. Por esta razón, la función objetivo es simplemente la minimización de esta variable. Además, las restricciones (1) y (2) indican que z tiene que ser mayor o igual que cada etiqueta que fue asignada a los vértices y aristas. De esta forma, ya que la intención es minimizar z, al final de la optimización, el valor de la variable será igual al valor de la máxima etiqueta que fue asignada.

Las restricciones (3) y (4) son las que definen el coloreo de aristas. Es una manera de representar que dos aristas que inciden en el mismo vértice reciben colores diferentes, es decir, $w(uj) \neq w(jv)$ si $(u,j),(j,v) \in E(G)$. El objetivo de estas restricciones es representar esta ecuación como una disyunción: tiene que suceder que $w(uj) < w(jv)$ o bien $w(uj) > w(jv)$. Recordemos que el color de una arista (u,v) es $w(uv) = f(u) + f(uv) + f(v)$. En el modelo, la función f la define las variables x, por lo tanto debe ser $x_u + x_{uj} + x_j \neq x_j + x_{jv} + x_v$, o lo que es lo mismo $x_u + x_{uj} - x_{jv} - x_v \neq 0$. Entonces, la variable δ_{ujv} ayuda a expresar esta ecuación por el siguiente motivo.

- veamos que si $\delta_{ujv} = 1$, efectivamente se cumple que $w(uj) < w(jv)$ como aparece en la definición de la variable. En este caso, la desigualdad (3) no impone restricción alguna. El lado derecho de la restricción se puede escribir como $x_u + x_{uj} - x_{jv} - x_v = x_u + x_{uj} - (x_{jv} + x_v)$. Para obtener una cota inferior de esta expresión, se tendría que asignar la etiqueta 1 a las variables con coeficiente positivo y la etiqueta Δ a las

variables con coeficiente negativo. Entonces, $x_u + x_{uj} - (x_{jv} + x_v) \geq 1 + 1 - (\Delta + \Delta) = 2 - 2\Delta$. Justamente es esta la expresión que queda si $\delta_{ujv} = 1$, la cuál se cumple para cualquier asignación válida de etiquetas.

En cambio, la restricción (4) dice que $x_u + x_{uj} - x_{jv} - x_v \leq -1$. Como las variables son enteras, se deduce que $x_u + x_{uj} - x_{jv} - x_v < 0$, es decir, $x_u + x_{uj} < x_{jv} + x_v$. Esto es lo mismo que decir que $w(uj) < w(jv)$.

- si $\delta_{ujv} = 0$, se tiene que cumplir que $w(uj) > w(jv)$ como aparece en la definición de la variable. La restricción (3) impone $x_u + x_{uj} - x_{jv} - x_v \geq 1$. Como las variables son enteras, se deduce que $x_u + x_{uj} - x_{jv} - x_v > 0$, es decir, $x_u + x_{uj} > x_{jv} + x_v$. Esto es equivalente a que $w(uj) > w(jv)$.

  Por otra parte, la desigualdad (4) no impone restricción alguna. Se cuenta con el lado derecho de la restricción $x_u + x_{uj} - x_{jv} - x_v = x_u + x_{uj} - (x_{jv} + x_v)$. Para conseguir una cota superior de la expresión, se tendría que asignar la etiqueta Δ a las variables con coeficiente positivo y la etiqueta 1 a las variables con coeficiente negativo. Entonces, $x_u + x_{uj} - (x_{jv} + x_v) \leq \Delta + \Delta - (1 + 1) = 2\Delta - 2$. En efecto, es la expresión que queda si $\delta_{ujv} = 0$, la cuál se cumple para cualquier asignación válida de etiquetas.

La restricciones (5), (6) y (7) acotan los valores de las variables x_w, x_{uv}, z respectivamente e indican que sólo pueden tomar valores enteros. Por definición del problema las variables deben ser mayores o iguales que 1, y se pueden acotar superiormente por Δ. Por último, la restricción (8) define a las variables δ_{ujv} como binarias.

Si $n = |V|$ el número de nodos y $m = |E|$ el número de aristas del grafo, se tiene lo siguiente: n variables para los nodos, m variables para las aristas y un orden cúbico en la cantidad de nodos de variables δ (además de la variable de minimización z). Como m es a lo sumo de orden cuadrático en n, el modelo tiene $\mathcal{O}(n^3)$ variables.

En cuanto a las restricciones, se cuenta con n restricciones de tipo (1) y m restricciones de tipo (2). Para cada variable δ, existen dos restricciones, por lo tanto el orden de estas restricciones es cúbico en n. Entonces, también se cuenta con $\mathcal{O}(n^3)$ restricciones.

3.1.3. Relajación lineal

Al analizar los puntos de la relajación para distintos grafos, se verifica siempre la siguiente situación:

- $x^*_v = 1$ para todo vértice v y $x^*_{uv} = 1$ para toda arista (u, v).
- Las variables δ^* toman valores $\frac{1}{2\Delta-1} \leq \delta^*_{ujv} \leq \frac{2\Delta-2}{2\Delta-1}$ para cumplir con las restricciones (3) y (4). De hecho, en la práctica, todas toman el mismo valor, que es un extremo de estas cotas.

Es decir, cualquiera sea el grafo de entrada, $z^* = 1$ (ya que se trata de minimizar esta variable) y por lo tanto el valor de la función objetivo es 1 también. Como $z \geq 1$, se puede concluir que este es un punto óptimo y que el valor óptimo de la relajación lineal es 1.

En el capítulo 6, se tratará de cortar este punto mediante desigualdades válidas para lograr relajaciones más ajustadas.

3.2. Modelo 2

El segundo modelo propuesto está basado en una formulación multimatching [9] para el coloreo de aristas. Al generar el etiquetado total del grafo, se debe tener cuidado que los colores de las aristas que queden determinados definan un coloreo. Por este motivo, agregando algunas variables y restricciones, se puede adaptar la formulación de coloreo de aristas.

Siguiendo la misma filosofía que la formulación multimatching, las variables representan la decisión de asignar determinado color a una arista. caso, se cuenta con variables binarias para cada vértice y arista que indican si reciben o no el valor de una etiqueta. Además, se introducen otras variables para las aristas con el objetivo de calcular sus colores, haciendo uso de las anteriores variables.

Al igual que en el modelo anterior, hay una variable usada para minimizar el valor de las etiquetas, siendo ésta la única variable que aparece en la función objetivo.

3.2.1. Variables

Las variables utilizadas en el modelo son las siguientes:

- $y \in \{1, 2, \ldots, \Delta\}$

  La variable y va a minimizar la función objetivo. Es decir, será la que represente el valor $\chi'_t(G)$.

- $x_{vj} = \begin{cases} 1 & \text{si } v \text{ recibe la etiqueta } j \\ 0 & \text{caso contrario} \end{cases}$

  Esta variable representa la decisión de etiquetar el vértice v con la etiqueta j. Si $x_{vj} = 1$ indica que $f(v) = j$.

- $w_{uvj} = \begin{cases} 1 & \text{si } (u, v) \text{ recibe la etiqueta } j \\ 0 & \text{caso contrario} \end{cases}$

  Esta variable representa la decisión de etiquetar a la arista (u, v) con la etiqueta j. Si $w_{uvj} = 1$ indica que $f(uv) = j$.

- $z_{uvk} = \begin{cases} 1 & \text{si } (u, v) \text{ recibe el color } k \\ 0 & \text{caso contrario} \end{cases}$

  La variable indica si quedó definido el color k para la arista (u, v). Si $z_{uvk} = 1$ indica que $w(uv) = f(u) + f(uv) + f(v) = k$.

Para las variables x_{vj} y w_{uvj}, el valor de la etiqueta j va a pertenecer al rango $\{1, \ldots, \Delta\}$ ya que siempre existe un coloreo de aristas por un Δ-etiquetado total para cualquier grafo. Con respecto a las variables z_{uvk}, como el color k depende del valor de las etiquetas, naturalmente estará en el rango $\{3, \ldots, 3\Delta\}$.

3.2.2. Formulación

A continuación se presenta la formulación del modelo:

$$
\begin{aligned}
&\text{minimizar} && y \\
&\text{sujeto a} \\
&(9) && \sum_{j=1}^{\Delta} x_{vj} = 1 && \forall v \in V \\
&(10) && \sum_{j=1}^{\Delta} w_{uvj} = 1 && \forall (u,v) \in E \\
&(11) && \sum_{k=3}^{3\Delta} z_{uvk} = 1 && \forall (u,v) \in E \\
&(12) && z_{uvk} \geq x_{uj} + w_{uvr} + x_{vp} - 2 && \forall (u,v) \in E \text{ y } j,r,p \in \{1,\ldots,\Delta\} \\
& && && \text{tal que } k = j + r + p \\
&(13) && \sum_{v \in N(u)} z_{uvk} \leq 1 && \forall u \in V, d(u) > 1 \text{ y } k \in \{3,\ldots,3\Delta\} \\
&(14) && y \geq \sum_{j=1}^{\Delta} j\, x_{vj} && \forall v \in V \\
&(15) && y \geq \sum_{j=1}^{\Delta} j\, w_{uvj} && \forall (u,v) \in E \\
&(16) && x_{vj} \in \{0,1\} && \forall v \in V \text{ y } j \in \{1,\ldots,\Delta\} \\
&(17) && w_{uvj}, z_{uvk} \in \{0,1\} && \forall (u,v) \in E \text{ y } k \in \{3,\ldots,3\Delta\}, \\
& && && j \in \{1,\ldots,\Delta\} \\
&(18) && 1 \leq y \leq \Delta, \text{ con } y \in \mathbb{Z}
\end{aligned}
$$

Fig. 3.2: Formulación del modelo 2.

donde $d(v)$ es el grado del vértice v y $N(v)$ es la vecindad de v (es decir, los vértices adyacentes a v). La función objetivo es únicamente la minimización de la variable y, variable que tomará el valor máximo de las etiquetas asignadas en el grafo.

Las restricciones (9) y (10) indican que cada vértice y cada arista deben recibir exactamente una etiqueta, respectivamente, y la restricción (11) dice que el color de la arista es único. Como son variables binarias y la sumatoria es igual a 1, quiere decir que exactamente una variable tiene que tomar el valor 1. De esta manera, nos aseguramos el etiquetado total del grafo y la unicidad del color definido en las aristas.

La restricción (12) representa la definición del color de una arista. Recordemos que las variables z se usan justamente para indicar el color que recibe una arista. Supongamos

que al vértice u se le asigna la etiqueta j, al vértice v la etiqueta p y la arista (u, v) que los une recibe etiqueta r. Entonces, por cómo se definieron las variables se debe cumplir que $z_{uvk} \geq x_{uj} + w_{uvr} + x_{vp} - 2 = 1 + 1 + 1 - 2 = 1$, para $k = j + r + p$. Entonces si $z_{uvk} \geq 1$, debe ser necesariamente $z_{uvk} = 1$ porque estas variables son binarias. Luego, es correcto ya que la arista (u, v) recibe el color k.

La restricción (13) verifica, por cada vértice, que todas las aristas que incidan sobre él tengan un color diferente, para formar un coloreo propio. Esto se hace en el modelo pidiendo que, dado un color k, solamente una de las aristas tenga la posibilidad de recibir dicho color.

Las restricciones (14) y (15) definen el valor de la variable y. Supongamos que el vértice v recibió la etiqueta j. Como la primera restricción asegura que los vértices reciben exactamente una etiqueta, la variable x_{vj} vale 1 y todas las demás x_{vi} son 0 para $i \in \{1, \ldots, \Delta\}$, $i \neq j$. Por lo tanto se cumple que $y \geq j$ y esto significa que el valor de y es al menos j. Entonces y va a tomar el valor máximo de las etiquetas usadas (la explicación es análoga para las etiquetas de las aristas).

Por último, las restricciones (16), (17) y (18), indican el dominio de las variables. Las variables x_{vj}, w_{uvj} y z_{uvk} son binarias, mientras que la variable y es entera y toma valores en el rango $\{1, \ldots, \Delta\}$.

Si $n = |V|$ el número de nodos y $m = |E|$ el número de aristas del grafo, se tiene lo siguiente: existe una variable por cada combinación vértice - etiqueta válida, o sea, una cantidad de $n\Delta$ variables. También se cuenta con una variable por cada combinación arista - etiqueta válida, otras $m\Delta$ variables. Lo mismo sucede con las variables que combina arista - color definido, que llega a un orden de $3m\Delta$ variables. Además se cuenta con la variables de minimización. Δ es a lo sumo $n - 1$ y m tiene orden cuadrático como máximo. Por lo tanto, el modelo tiene $\mathcal{O}(n^3)$ variables.

Se cuenta con n restricciones de tipo (1) y m restricciones de tipo (2) y tipo (3). Las restricciones (4) son del orden de $m\Delta^3$ ya que por cada arista se necesitan todos los colores que se pueden formar con la suma de las etiquetas correspondientes. Las restricciones (5) tienen un orden de $3n\Delta$ y existen n restricciones de tipo (6) y m de tipo (7). Entonces, sabiendo que Δ tiene orden lineal en n y m tiene orden cuadrático en m, el modelo posee $\mathcal{O}(n^5)$ restricciones.

3.2.3. Relajación inicial

En este caso, un punto óptimo de la relajación lineal satisface la siguiente configuración de valores para las variables:

- las variables $x^*_{u1} = \frac{2\Delta+1}{3\Delta}$, $x^*_{u2} = \frac{\Delta-1}{3\Delta}$ y $x^*_{ul} = 0$ para cualquier vértice u y etiqueta $l \neq 1, 2$.
- las variables $w^*_{uv1} = \frac{2\Delta+1}{3\Delta}$, $w^*_{uv2} = \frac{\Delta-1}{3\Delta}$ y $w^*_{uvl} = 0$ para cualquier arista (u, v) y etiqueta $l \neq 1, 2$.
- las variables z^*_{uvk} se acomodan para cumplir las restricciones (3) y (4).

De esta manera, el valor de $y^* = \frac{4\Delta-1}{3\Delta}$ para satisfacer las restricciones (6) y (7). Veamos la cuenta para un vértice u cualquiera (el resultado es similar para las aristas).

$$\begin{aligned} y^* &\geq x^*_{u1} + 2x^*_{u2} + 3x^*_{u3} + \cdots + \Delta x^*_{u\Delta} \\ &= \frac{2\Delta+1}{3\Delta} + 2\frac{\Delta-1}{3\Delta} + 3{,}0 + \cdots + \Delta{,}0 \\ &= \frac{2\Delta+1}{3\Delta} + \frac{2\Delta-2}{3\Delta} \\ &= \frac{2\Delta+1+2\Delta-2}{3\Delta} \\ &= \frac{4\Delta-1}{3\Delta} \end{aligned}$$

con lo cuál $y^* = \frac{4\Delta-1}{3\Delta}$ y entonces ese es el valor de la función objetivo ya que se trata de un problema de minimización.

A continuación, se va a demostrar que este es el valor óptimo de la relajación lineal haciendo uso del problema dual. El problema dual tiene una variable por cada restricción del problema primal y una restricción por cada variable del problema primal. Además, como el primal es un problema de minimización, el dual será un problema de maximización.

Está demostrado que para cualquier punto factible z del dual, el valor de la función objetivo del dual en ese punto es una cota inferior para el problema primal. Como corolario se tiene que si x^* es un punto factible del primal y z^* es un punto factible del dual en dónde los respectivos valores de las funciones objetivo evaluadas en ese punto coinciden, entonces x^* es un óptimo del problema primal y z^* es un óptimo del problema dual.

Entonces, para demostrar que el punto descripto es el óptimo de la relajación del modelo 2, se va a contruir el problema dual y mostrar un punto factible con igual función objetivo. Para la contrucción del modelo dual, se van a definir las siguiente variables:

- V_v, $\forall v \in V(G)$ las variables duales correspondientes a las restricción (1)
- A_{uv}, $\forall (u,v) \in E(G)$ las variables duales correspondientes a las restricción (2)
- P_{uv}, $\forall (u,v) \in E(G)$ las variables duales correspondientes a las restricción (3)
- Z_{uvjrp}, $\forall v \in V(G)$, $j,r,p \in \{1,\dots\Delta\}$ las variables duales correspondientes a las restricción (4)
- C_{uk}, $\forall u \in V(G)$, $k \in \{3,\dots,3\Delta\}$ las variables duales correspondientes a las restricción (5)
- Y_v, $\forall v \in V(G)$ las variables duales correspondientes a las restricción (6)
- Y_{uv}, $\forall (u,v) \in E(G)$ las variables duales correspondientes a las restricción (7)

Por simplicidad, al formular el problema dual no se consideran las restricciones (8), (9) y (10) ya que las restricciones (1), (2) y (3) del modelo original garantizan que x_{vj}, w_{uvj}, $z_{uvk} \leq 1$ y, como se está minimizando la variable y en la función objetivo, no es necesaria una cota superior para la variable. Luego, el modelo del problema dual queda definido de la siguiente forma:

$$\text{maximizar} \sum_{v\in V(G)} V_v + \sum_{(u,v)\in E(G)} A_{uv} + \sum_{(u,v)\in E(G)} P_{uv} - 2 \sum_{\substack{(u,v)\in E(G)\\ j,r,p\in\{1,\dots,\Delta\}}} Z_{uvjrp} - \sum_{\substack{u\in V(G), d(u)>1\\ k\in\{3,\dots,3\Delta\}}} C_{uk}$$

sujeto a

(1) $$\sum_{v\in V(G)} Y_v + \sum_{(u,v)\in E(G)} Y_{uv} \leq 1$$

(2) $$V_v - \sum_{\substack{\forall u:(v,u)\in E(G)\\ r,p\in\{1,\dots,\Delta\}}} Z_{vujrp} - \sum_{\substack{\forall u:(u,v)\in E(G)\\ r,p\in\{1,\dots,\Delta\}}} Z_{uvprj} - j\, Y_v \leq 0 \qquad \forall v \in V(G) \text{ y } j \in \{1,\dots,\Delta\}$$

(3) $$A_{uv} - \sum_{r,p\in\{1,\dots,\Delta\}} Z_{uvrjp} - j\, Y_{uv} \leq 0 \qquad \forall (u,v) \in E(G) \text{ y } j \in \{1,\dots,\Delta\}$$

(4) $$P_{uv} + \sum_{\substack{s,r,p\in\{1,\dots,\Delta\}\\ j=s+r+p}} Z_{uvsrp} - C_{uj} - C_{vj} \leq 0 \qquad \forall (u,v) \in E(G), d(u) > 1, d(v) > 1 \text{ y } j \in \{3,\dots,3\Delta\}$$

(5) V_v, A_{uv}, P_{uv} libres

(6) $Y_v, Y_{uv}, Z_{uvprj}, C_{uk} \geq 0$

Fig. 3.3: Formulación del problema dual para el modelo 2.

Sea v un vértice de grado máximo, es decir $d(v) = \Delta$. Se puede ver que el siguiente es un punto factible del problema dual:

- $V_v = \frac{2}{3}$ y $V_u = \frac{2}{3\Delta}$, $\forall u \in N(v)$
- $A_{uv} = \frac{2}{3\Delta}$, $\forall u \in N(v)$
- $Y_v = \frac{1}{3}$ y $Y_u = \frac{1}{3\Delta}$, $\forall u \in N(v)$
- $Y_{uv} = \frac{1}{3\Delta}$, $\forall u \in N(v)$
- $C_{v3} = \frac{1}{3\Delta}$
- $Z_{uv111} = \frac{1}{3\Delta}$, $\forall u \in N(v)$
- el resto de las variables en 0

Veamos cuánto vale la función objetivo en este punto:

$$
\begin{aligned}
fo &= \sum_{v \in V(G)} V_v + \sum_{(u,v) \in E(G)} A_{uv} + \sum_{(u,v) \in E(G)} P_{uv} - 2 \sum_{\substack{(u,v) \in E(G) \\ j,r,p \in \{1,\dots,\Delta\}}} Z_{uvjrp} - \sum_{\substack{u \in V(G), d(u) > 1 \\ k \in \{3,\dots,3\Delta\}}} C_{uk} \\
&= \left(\frac{2}{3} + \Delta \frac{2}{3\Delta} \right) + \Delta \frac{2}{3\Delta} + 0 - 2\Delta \frac{1}{3\Delta} - \frac{1}{3\Delta} \\
&= \frac{2}{3} + \frac{2}{3} + \frac{2}{3} - \frac{2}{3} - \frac{1}{3\Delta} \\
&= \frac{4}{3} - \frac{1}{3\Delta} \\
&= \frac{4\Delta - 1}{3\Delta}
\end{aligned}
$$

Entonces, encontramos un punto factible del dual cuyo valor en la función objetivo coincide con el punto descripto del problema primal en su función objetivo. Luego, el punto del problema primal es un óptimo de la relajación lineal (además, el punto obtenido para el dual es óptimo).

En el capítulo 7 se presentan algunas desigualdades válidas con el objetivo de cortar este punto para lograr mejores relajaciones.

4. COMPARACIÓN ENTRE LOS MODELOS

Para analizar los modelos presentados se realizaron pruebas con distintos grafos. Se utilizaron grafos de 15 nodos con distintas densidades de aristas. Las aristas se eligieron de forma aleatoria y se usaron densidades que van de 10 % a 90 %. Entonces, por cada densidad se generaron 10 grafos al azar. Vamos a tomar como grafos de densidad baja los que tienen 10 %, 20 % y 30 % de densidad de aristas. Los grafos que tienen una densidad de 40 %, 50 % y 60 % los clasificamos como grafos de densidad media y por último los grafos de densidad alta serán los que tengan 70 %, 80 % y 90 % de densidad de aristas. Para cada una de estas clasificaciones se tomó valores promedio.

Se implementaron los modelos en *C++* utilizando el software CPLEX 12.4 y las pruebas se ejecutan en una computadora con un procesador i7 de 3.4GHz y 16Gb de memoria ram.

En primera instancia se tomó en cuenta para la comparación de los modelos los tiempos de ejecución y los valores de las relajaciones lineales. En el capítulo anterior se caracterizaron los óptimos de las relajaciones de ambos modelos. El valor óptimo de la relajación del primer modelo es 1 y el del segundo modelo es $\frac{4\Delta-1}{3\Delta}$ (valor que está entre 1 y $\frac{4}{3}$). Además, los tiempos medidos en esta prueba para ambos modelos son casi instantáneos, en milésimas de segundos. A pesar de que los tiempos registrados para el modelo 1 fueron ligeramente menores a los registrados por el modelo 2, los valores cercanos obtenidos para las relajaciones y tiempos medidos no nos pareció medida suficiente para distinguir entre alguno de los dos modelos.

Como segundo criterio de comparación se decidió utilizar el algoritmo *Branch & Cut* que viene por defecto en CPLEX y establecer como tiempo límite para la resolución de cada instancia 3600 segundos (1 hora). La Tabla 4.1 muestra una comparación de los resultados conseguidos para cada modelo. Los tiempos están en segundos y los *gaps* son porcentajes. Se debe considerar que la cuenta para hallar el *gap* es la siguiente:

$$gap = 100 * \frac{cota_{superior} - cota_{inferior}}{cota_{superior}}$$

donde $cota_{superior}$ se refiere al valor óptimo si se encontró o a la mejor solución entera que se pudo obtener y $cota_{inferior}$ es la mejor relajación obtenida en el árbol del algoritmo.

	Modelo 1		*Modelo 2*	
Densidad	tiempo (seg)	gap (%)	tiempo (seg)	gap (%)
Baja	129.96	1.167	262.5	2.56
Media	2569.19	26.90	3361.72	43.05
Alta	3600	67.58	3600	67.27

Tab. 4.1: Resultados obtenidos para cada modelo.

En primera instancia se puede notar que los valores para cada modelo no difieren en gran medida. Para grafos de baja y media densidad se puede observar que el modelo 1 logra *gaps* más chicos en menos tiempo que el modelo 2. Ambos modelos llegan al valor

óptimo en casi todas las instancias de grafos de baja densidad. Sin embargo, el modelo 1 resuelve más instancias de grafos de media densidad y para los que no llega a calcular el óptimo, en la mayoría de los casos, obtiene un mejor *gap*.

En el caso de grafos de densidad alta, en la tabla se observa que ninguno de los dos modelos logra obtener valores óptimos para las instancias ya que agotan el tiempo límite en cada una. A diferencia de lo sucedido con los grafos de baja y media densidad, el *gap* promedio de los modelos son similares (incluso, el del modelo 2 es levemente más bajo). Además, en el tiempo establecido, los modelos no obtienen un buen *gap*, es decir, la mejor solución encontrada está lejos de la mejor relajación.

El análisis realizado no brinda evidencia que permita concluir la superioridad de alguno de los modelos como base de un algoritmo de *Branch & Cut*.

Su comportamiento fue similar, tanto en tiempo de resolución de las relajaciones, cota de éstas y performance dada por el *Branch & Cut* general implementado por CPLEX para cada modelo. En base a esto se decidió no descartar a ninguno de ellos y continuar con el estudio de ambos modelos.

Los próximos capítulos tienen como objetivo desarrollar un algoritmo de *Branch & Cut* para cada modelo que logren mejorar estos resultados. Se implementarán heurísticas para mejorar las soluciones enteras y diversas desigualdades válidas que ajusten las relajaciones. La intención es estudiar en más profundidad las características del conjunto de soluciones enteras factibles para obtener información general que ayude a resolver cualquier instancia del problema.

Como se mencionó en capítulos anteriores, un problema de programación lineal entera tiene asociado un poliedro $P = \{x \in \mathbb{R}^n_{\geq 0} : Ax \leq b\}$, que reúne las soluciones de la relajación lineal. El conjunto de soluciones en donde se busca el óptimo es $S = P \cap \mathbb{Z}^n$ y sería deseable poder caracterizar la cápsula convexa $conv(S)$, el menor poliedro que contiene a S. No es tarea fácil describir $conv(S)$, pero se analizarán características de este poliedro con el objetivo de fortalecer el modelo identificando desigualdades válidas que puedan ser usadas en un algoritmo *Branch & Cut*.

El estudio del poliedro asociado al problema permite identificar desigualdades lineales útiles como planos de corte, es decir que separen las soluciones óptimas de las relajaciones lineales en cada nodo del árbol de *Branch & Bound*. De esta manera, se busca disminuir el tamaño del árbol de enumeración y obtener cotas más ajustadas en cada nodo. Se debe lograr un equilibrio entre la cantidad de desigualdades agregadas al modelo y el tiempo que tarda en calcularse la relajación lineal. Es decir, se puede agregar una cantidad de restricciones al modelo que ocasione un aumento considerable en el tiempo de resolución de la relajación. Aquí se debe considerar la mejora en el valor de la relajación. No son de gran utilidad restricciones que aumentan mucho el tiempo computacional y no logran obtener buenas cotas.

5. HEURÍSTICAS

5.1. Heurísticas iniciales

Incorporar una heurística inicial previo a empezar con el *Branch & Bound* puede ser un factor importante para mejorar el rendimiento del algoritmo. Se busca proveer una solución factible de buena calidad por más que no sea la óptima. En este caso, dado un grafo G, el objetivo es producir de manera rápida un etiquetado total que forme un coloreo de aristas. Esta solución inicial permite contar con una cota superior para el *Branch & Bound*. De esta manera, no es necesario explorar los nodos en donde el valor de la función objetivo de la relajación inicial correspondiente es mayor que el valor de la solución hallada. Además, en este caso, sirve para mejorar la cota superior y reducir la cantidad de variables y restricciones.

En esta sección se explican tres heurísticas iniciales para afrontar el problema. En primer lugar se describe una heurística que usa Δ, el grado máximo de G, como valor máximo en el etiquetado. Este etiquetado se sabe que existe y el objetivo es encontrarlo. Luego, se desarrollan dos heurísticas más con el objetivo bajar de Δ la etiqueta máxima de la solución.

Con las últimas dos heurísticas, la intención es generar un coloreo de aristas por un k-etiquetado total, donde $k < \Delta$. Si no se pudiera lograr esto, se cuenta con el primer algoritmo que asegura un Δ-etiquetado.

5.1.1. Heurística Constructiva (HC)

En [1] se realiza una demostración constructiva del siguiente teorema:

$$\textit{si } G \textit{ es un grafo con grado máximo } \Delta\textit{, entonces } \chi'_t(G) \leq \Delta$$

Siguiendo esta demostración implementamos un algoritmo que etiqueta cualquier grafo usando Δ como valor máximo del etiquetado. Este etiquetado se va construyendo en una serie de pasos que se explican a continuación.

- Sea $c : E(G) \to \{1, 2, \ldots, \Delta + 1\}$ un coloreo propio de aristas de G, el cuál existe por el teorema de Vizing.

  Para generar este coloreo de aristas se hizo uso de una demostración constructiva del teorema de Vizing ideada por Misra y Gries [10].

- El subgrafo que contiene las aristas que recibieron color Δ y Δ+1 consiste en caminos y ciclos pares, por lo tanto es bipartito. Lo siguiente es fijar una bipartición $A \cup B$ de $V(G)$ tal que todos las aristas con colores Δ y $\Delta + 1$ tengan un extremo en A y el otro extremo en B (los vértices que no tengan ninguna arista incidente con color Δ ni $\Delta + 1$ pueden fijarse en cualquiera de los dos conjuntos).

- En este paso se debe asignar con etiqueta 1 a todos los vértices pertenecientes a A y con etiqueta Δ a todos los vértices de B.

- Asignar etiqueta $c(e)$ a toda arista e entre vértices de A y etiqueta $c(e) + 1$ a toda arista e entre vértices de B.
- Por último, se determinan las etiquetas de las aristas en el grafo bipartito conformado por las aristas que unen A y B mediante un Δ coloreo de aristas. El número óptimo de colores para un grafo bipartito es exactamente Δ y se puede realizar en tiempo polinomial.

Al terminar de aplicar estos pasos, el grafo queda etiquetado con valores de etiquetas menores o iguales a Δ. Además, se puede ver que las aristas que unen vértices en A reciben colores entre 3 y $\Delta + 1$, las aristas que unen los conjuntos A y B reciben colores entre $\Delta + 2$ y $2\Delta + 1$, y las aristas que unen vértices en B reciben colores entre $\Delta + 2$ y 3Δ. Ya que estos colores forman coloreos de aristas entre los conjuntos A y B, y dentro de ellos, se generó un coloreo de aristas del grafo. Entonces, con este algoritmo se produce un coloreo de aristas por un Δ-etiquetado total.

5.1.2. Heurística por Coloreo de Vértices (HCV)

Para tratar de obtener un etiquetado cuya máxima etiqueta sea menor que Δ, se propone primero realizar un coloreo de los vértices. Es decir, etiquetar los vértices de tal manera que las etiquetas de vértices adyacentes sean diferentes. Luego, usando este etiquetado de vértices como punto de partida, se procede a etiquetar las aristas teniendo cuidado que los colores (recordar que el color de una arista se define como la suma de la etiqueta propia de la arista y las etiquetas de sus vértices extremo) que reciban aristas que inciden en un mismo vértice sean diferentes para generar un coloreo de aristas. La idea detrás de la heurística es tratar de generar una buena distribución de las etiquetas de los vértices para después no tener que usar valores altos para las etiquetas de las aristas.

El coloreo de vértices se realiza mediante un algoritmo goloso. Este algoritmo utiliza como etiqueta máxima a lo sumo el valor $\Delta + 1$. Notar que si se utiliza la etiqueta con valor $\Delta + 1$ en este paso, no sirve para el propósito de esta heurística y no es necesario continuar. De la misma manera, para etiquetar las aristas se eligió un enfoque goloso. A continuación se describen los dos algoritmos.

El procedimiento para etiquetar los vértices del grafo es el algoritmo secuencial clásico que recorre cada vértice consecutivamente hasta haber etiquetado todos.

Algoritmo 4 Etiquetado secuencial de vértices

Entrada: grafo G, con orden de vértices $\{v_1, v_2, \ldots, v_n\}$

1: `vertexLabel`$(v_1) = 1$
2: **desde** $j = 2$ **hasta** n **hacer**
3: `vertexLabel`$(v_j) = min\{l : l \geq 1$ y `vertexLabel`$(v_i) \neq l, \forall (v_j, v_i) \in E\}$
4: **fin desde**
5: **devolver** etiquetado de vértices formado por `vertexLabel`

Comienza etiquetando el vértice v_1 con 1. Luego recorre los siguientes vértices v_j, asignándole la mínima etiqueta que no es usada por sus vértices adyacentes.

El algoritmo para etiquetar las aristas tiene una lógica similar al algoritmo anterior. Se va etiquetando todas la aristas secuencialmente.

Algoritmo 5 Etiquetado secuencial de aristas

Entrada: grafo G, con orden de aristas $\{e_1, e_2, \ldots, e_m\}$ y vértices etiquetados

1: $\texttt{edgeLabel}(e_1) = 1$
2: **desde** $j = 2$ **hasta** m **hacer**
3: $e_j = (u_j, v_j)$
4: $p = \texttt{vertexLabel}(u_j) + \texttt{vertexLabel}(v_j)$
5: $W = \{w : w = \texttt{vertexLabel}(s_i) + \texttt{edgeLabel}(s_i, u_j) + \texttt{vertexLabel}(u_j),$
$\forall (s_i, u_j) \in E,\ 1 \leq i \leq j-1\} \bigcup$
$\{w : w = \texttt{vertexLabel}(v_j) + \texttt{edgeLabel}(v_j, r_i) + \texttt{vertexLabel}(r_i),$
$\forall (v_j, r_i) \in E,\ 1 \leq i \leq j-1\}$
6: $\texttt{edgeLabel}(e_j) = min\{l : l \neq w - p \text{ y } l \geq 1,\ \forall w \in W\}$
7: **fin desde**
8: **devolver** etiquetado de aristas formado por `edgeLabel`

Comienza etiquetando la arista e_1 con 1. Luego, prosigue con las siguientes aristas e_j, haciendo lo siguiente: en primer lugar se calcula la suma p de las etiquetas de los vértices extremo de e_j. A continuación se calculan los colores de las aristas que inciden en e_j y ya fueron etiquetadas. La diferencia entre cada uno de estos colores y p determinan las etiquetas que no se puede asignar a e_j. Entonces, se elige la menor etiqueta que puede ser usada por la arista. Para resumir, la heurística consiste en etiquetar los vértices del grafo de manera que defina un coloreo de vértices y después completar el etiquetado total con las aristas.

5.1.3. Heurística por Grafo Auxiliar (HGA)

Al analizar el procedimiento y los resultados de la HCV se puede extraer dos conclusiones en lo referido a etiquetar los vértices del grafo como paso inicial. La primera es que no es necesario que vértices adyacentes reciban etiquetas distintas (como sucede al colorear los vértices del grafo). La única condición del problema es que aristas adyacentes reciban colores distintos. Esta situación es una restricción fuerte impuesta por la heurística anterior. La segunda conclusión se puede ver con un ejemplo concreto (Fig. 5.1).

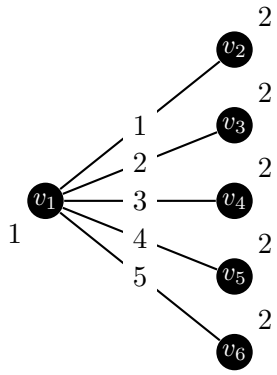

Fig. 5.1: Grafo de ejemplo.

En este caso, al colorear primero los vértices, habría que asignar etiquetas diferentes a cada arista, generando un coloreo de aristas por un 5-etiquetado total. Sin embargo, este grafo particular es un árbol A y en [1] se demuestra que $\chi'_t(A) = \lceil \frac{\Delta+1}{2} \rceil$ con Δ el grado máximo de A. Por lo tanto, en este ejemplo, el óptimo es $\chi'_t(A) = \lceil (5+1)/2 \rceil = 3$ y se podría etiquetar como muestra la Fig. 5.2.

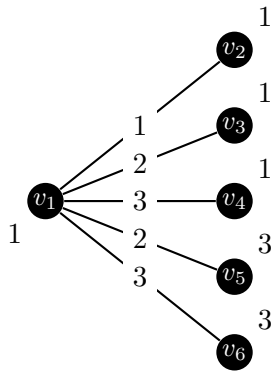

Fig. 5.2: Grafo con solución óptima.

Teniendo en cuenta este ejemplo, para un vértice v, no parece una buena decisión asignar la misma etiqueta a los vértices adyacentes de v (o a la mayoría de ellos). Por esta razón, en esta heurística, el primer paso será construirse un grafo auxiliar X a partir del grafo de entrada G. El grafo X tendrá exactamente los mismos vértices de G, pero existirá una arista entre dos vértices v y w si y sólo si existe un vértice u en G que sea adyacente a v y a w. Es decir,

$$\exists (v,u),(u,w) \in E(G) \iff (v,w) \in E(X)$$

Supongamos que se cuenta con el grafo de entrada de la Fig. 5.3.

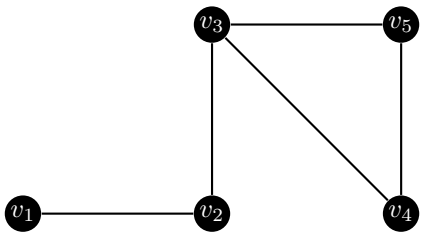

Fig. 5.3: Grafo G de entrada.

El grafo auxiliar que queda construido a partir del grafo de entrada es el que muestra la Fig. 5.4.

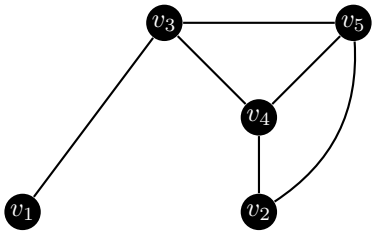

Fig. 5.4: Grafo auxiliar X construido a partir del grafo de entrada.

Una vez contruido el grafo auxiliar X se procede a etiquetarlo de la misma manera que en la HCV, pero usando valores de etiquetas menores o iguales a $\lceil \frac{\Delta+1}{2} \rceil + K$, donde K es un parámetro a variar. Por lo tanto, se tendrá la intención de etiquetar los vértices del grafo X para que vértices en G que poseen un vértice adyacente en común reciban distinto valor de etiqueta. Notar que al restringir el valor máximo de las etiquetas a asignar puede no generarse un coloreo de vértices de X. Esta decisión se tomó porque nuevamente no hay necesidad de imponer condiciones en las etiquetas de vértices adyacentes, en principio pueden recibir la misma etiqueta. Además, se etiqueta usando valores menores o iguales a $\lceil \frac{\Delta+1}{2} \rceil + K$ porque se sabe que $\lceil \frac{\Delta+1}{2} \rceil$ es una cota inferior del problema (es decir, no se puede generar un coloreo de aristas por un k-etiquetado total con $k < \lceil \frac{\Delta+1}{2} \rceil$). En los experimentos, se varía el K de tal manera que $\lceil \frac{\Delta+1}{2} \rceil + K < \Delta$ ya que el propósito de la heurística es generar un etiquetado total con valor de etiqueta máxima menor que Δ.

Algoritmo 6 Etiquetado de vértices mediante grafo auxiliar

Entrada: grafo auxiliar X, con orden de vértices $\{v_1, v_2, \ldots, v_n\}$ y un entero K

1: $next = 0$
2: $maxLabel = \left\lceil \frac{\Delta+1}{2} \right\rceil + K$
3: $\texttt{vertexLabel}(v_1) = 1$
4: **desde** $j = 2$ **hasta** n **hacer**
5: $label = min\{l : l \geq 1$ y $\texttt{vertexLabel}(v_i) \neq l, \forall (v_j, v_i) \in E(X)\}$
6: **si** $label \leq maxLabel$ **entonces**
7: $\texttt{vertexLabel}(v_j) = label$
8: **si no**
9: $\texttt{vertexLabel}(v_j) = next + 1$
10: $next = (next + 1)\,\%maxLabel$
11: **fin si**
12: **fin desde**
13: **devolver** etiquetado de vértices formado por `vertexLabel`

Se comienza etiquetando el vértice v_1 con 1 y luego, se etiqueta el resto de los vértices. Para cada vértice v_j se obtiene la mínima etiqueta que no es usada por los vértices adyacentes en X. Si el valor obtenido es menor o igual que la máxima etiqueta permitida, se le asigna ese valor a v_j. De lo contrario, se le asigna una etiqueta dentro del rango permitido de la siguiente forma: se cuenta con una etiqueta *next* que comienza en 1, y cada vez que se supera el valor de la máxima etiqueta permitida, se asigna la etiqueta *next* y se incrementa en uno este valor. Si la etiqueta *next* llega al máximo valor permitido, se vuelve a comenzar en 1.

Una vez etiquetado los vértices de G con el procedimiento descrito anteriormente, se utiliza el mismo algoritmo secuencial explicado en la heurística anterior para etiquetar las aristas.

Finalmente, este algoritmo se ejecuta para cada K en tanto $\left\lceil \frac{\Delta+1}{2} \right\rceil + K < \Delta$. El objetivo es reportar el valor de K que haya obtenido el mínimo valor de la etiqueta más alta en el etiquetado.

5.1.4. Experimentos computacionales

En este apartado se presentan los resultados obtenidos en la ejecución de las heurísticas implementadas, con diversas instancias de grafos. Los grafos de prueba tienen 100 vértices y son generados al azar, con densidades de aristas entre 10 % y 90 %. Para cada densidad se generaron 5 grafos.

Parámetro K	Etiquetado obtenido
0	79-etiquetado total
1	80-etiquetado total
2	85-etiquetado total
3	77-etiquetado total
4	70-etiquetado total
5	76-etiquetado total
6	70-etiquetado total
7	68-etiquetado total
8	63-etiquetado total
9	65-etiquetado total
10	64-etiquetado total
11	66-etiquetado total
12	74-etiquetado total
13	70-etiquetado total
14	50-etiquetado total
15	51-etiquetado total
16	52-etiquetado total
17	53-etiquetado total
18	54-etiquetado total
19	55-etiquetado total
20	56-etiquetado total
21	57-etiquetado total
22	58-etiquetado total
23	59-etiquetado total
24	60-etiquetado total
25	61-etiquetado total
26	62-etiquetado total
27	63-etiquetado total
28	64-etiquetado total
29	65-etiquetado total
30	66-etiquetado total
31	67-etiquetado total
32	68-etiquetado total
33	69-etiquetado total

Tab. 5.1: Ejecución de la HGA para un grafo con $\Delta = 70$.

La Tab. 5.1 muestra, a modo de ejemplo, los resultados obtenidos de la HGA, variando el parámetro K para un grafo. Como se mencionó, el grafo es de 100 vértices generado de manera aleatoria. El grafo usado para esta ejecución tiene una densidad de aristas de 60 % (2851 aristas), con grado máximo $\Delta = 70$. Entonces, la cota inferior para el etiquetado es $\lceil (70+1)/2 \rceil = 36$. Se puede ver que para valores de K bajos, en general la máxima etiqueta supera Δ. Sin embargo, a medida que se va incrementando K pareciera que el valor de la máxima etiqueta en la mayoría de los casos es menor a Δ. En este caso particular, desde $K = 14$ la máxima etiqueta utilizada es igual a Δ o queda por debajo. Justamente, para $K = 14$ se consigue el mejor etiquetado para este grafo utilizando esta heurística, con un coloreo de aristas mediante un 50-etiquetado total.

grafo	grado máximo Δ	tiempo (seg)	Mejor etiquetado obtenido	Mejor parámetro K
G_1	20	0.0197	14-etiquetado total	2
G_2	19	0.0176	15-etiquetado total	2
G_3	19	0.0183	15-etiquetado total	5
G_4	16	0.0137	14-etiquetado total	2
G_5	17	0.0149	14-etiquetado total	4
G_6	28	0.0822	23-etiquetado total	8
G_7	30	0.0891	22-etiquetado total	3
G_8	32	0.104	26-etiquetado total	2
G_9	36	0.0988	27-etiquetado total	5
G_{10}	30	0.0854	25-etiquetado total	0
G_{11}	42	0.245	32-etiquetado total	3
G_{12}	39	0.24	31-etiquetado total	5
G_{13}	38	0.226	33-etiquetado total	7
G_{14}	41	0.255	33-etiquetado total	12
G_{15}	39	0.247	32-etiquetado total	5

Tab. 5.2: Ejecución de la HGA para grafos con densidad baja de aristas.

grafo	grado máximo Δ	tiempo (seg)	Mejor etiquetado obtenido	Mejor parámetro K
G_1	52	0.565	41-etiquetado total	11
G_2	50	0.508	40-etiquetado total	7
G_3	52	0.552	39-etiquetado total	8
G_4	51	0.536	37-etiquetado total	7
G_5	49	0.498	36-etiquetado total	8
G_6	60	0.944	49-etiquetado total	18
G_7	60	0.992	50-etiquetado total	19
G_8	62	0.994	50-etiquetado total	9
G_9	61	1.00	48-etiquetado total	6
G_{10}	57	0.94	47-etiquetado total	8
G_{11}	74	1.73	50-etiquetado total	12
G_{12}	70	1.63	50-etiquetado total	14
G_{13}	71	1.64	50-etiquetado total	14
G_{14}	68	1.62	50-etiquetado total	15
G_{15}	70	1.58	50-etiquetado total	14

Tab. 5.3: Ejecución de la HGA para grafos con densidad media de aristas.

grafo	grado máximo Δ	tiempo (seg)	Mejor etiquetado obtenido	Mejor parámetro K
G_1	80	2.54	56-etiquetado total	9
G_2	77	2.38	58-etiquetado total	11
G_3	78	2.51	62-etiquetado total	10
G_4	82	2.62	59-etiquetado total	10
G_5	81	2.63	56-etiquetado total	9
G_6	86	3.49	65-etiquetado total	21
G_7	89	3.71	67-etiquetado total	22
G_8	89	3.79	67-etiquetado total	22
G_9	87	3.64	66-etiquetado total	6
G_{10}	89	3.75	66-etiquetado total	20
G_{11}	95	5.00	63-etiquetado total	15
G_{12}	95	5.05	61-etiquetado total	13
G_{13}	94	5.06	59-etiquetado total	11
G_{14}	95	5.13	61-etiquetado total	13
G_{15}	95	5.12	60-etiquetado total	12

Tab. 5.4: Ejecución de la HGA para grafos con densidad alta de aristas.

En las Tab. 5.2, 5.3 y 5.4 se muestran los resultados obtenidos de la HGA para los grafos de densidad baja, media y alta, respectivamente. Se presentan los tiempos medidos para cada grafo, el mejor etiquetado generado y el parámetro K que realiza dicho etiquetado. Se puede destacar que se cumple en todos los casos el objetivo de producir un etiquetado cuya etiqueta máxima sea menor que Δ.

Para evaluar el comportamiento del parámetro K, presentamos los gráficos 5.5 a 5.13. En ellos se puede observar, para cada densidad de aristas, el promedio del etiquetado encontrado para cada valor de K, resaltando el valor promedio de Δ mediante la línea horizontal negra y el valor para $K = \lceil \Delta/4 \rceil$ mediante la línea vertical roja. De estos gráficos podemos concluir que no es posible determinar un valor de K que genere siempre el mejor etiquetado, pero muestran que en general para valores de K cercandos a $\lceil \Delta/4 \rceil$ se obtienen los mejores resultados. En base a esto, se decidió variar el valor de K entre $\lceil \Delta/8 \rceil$ y $\lceil 3\Delta/8 \rceil$ para los experimentos computacionales (lo que equivaldría a tomar la porción del medio de los gráficos).

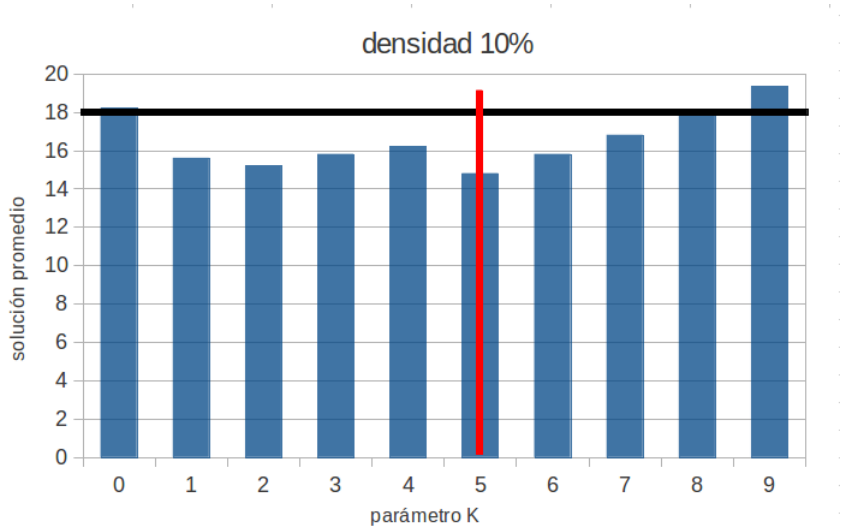

Fig. 5.5: HGA para grafos de densidad 10 %.

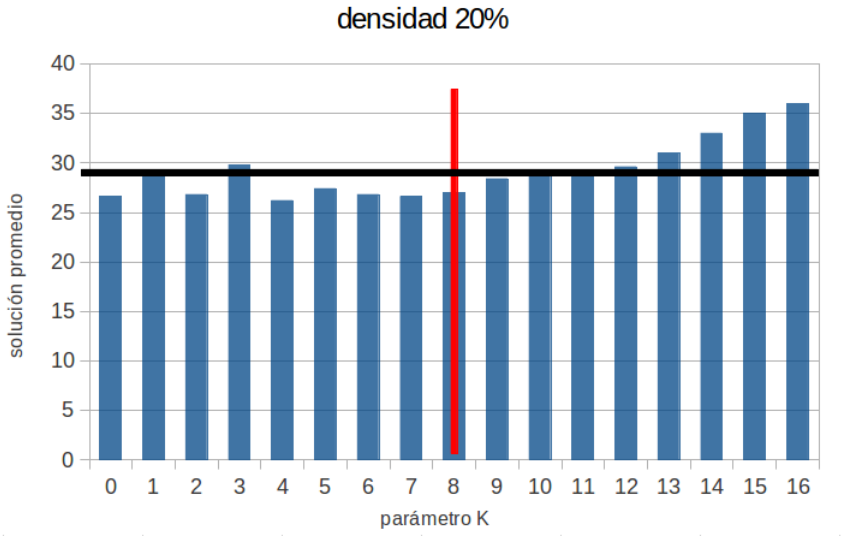

Fig. 5.6: HGA para grafos de densidad 20 %.

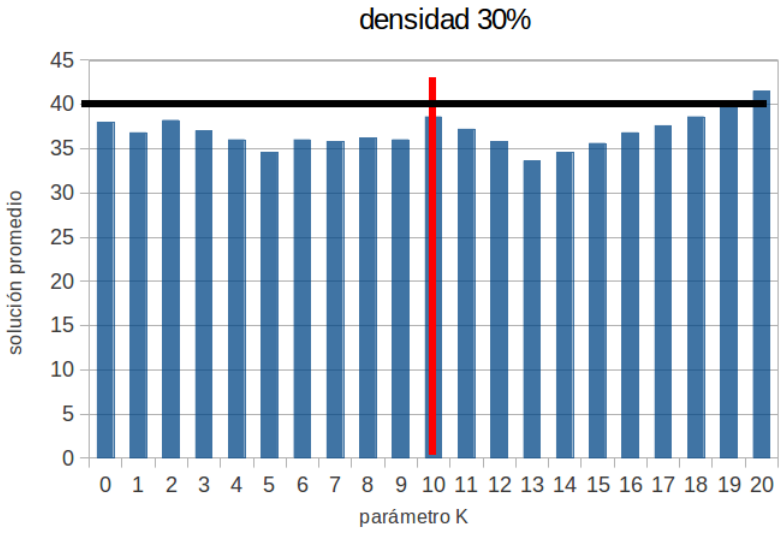

Fig. 5.7: HGA para grafos de densidad 30 %.

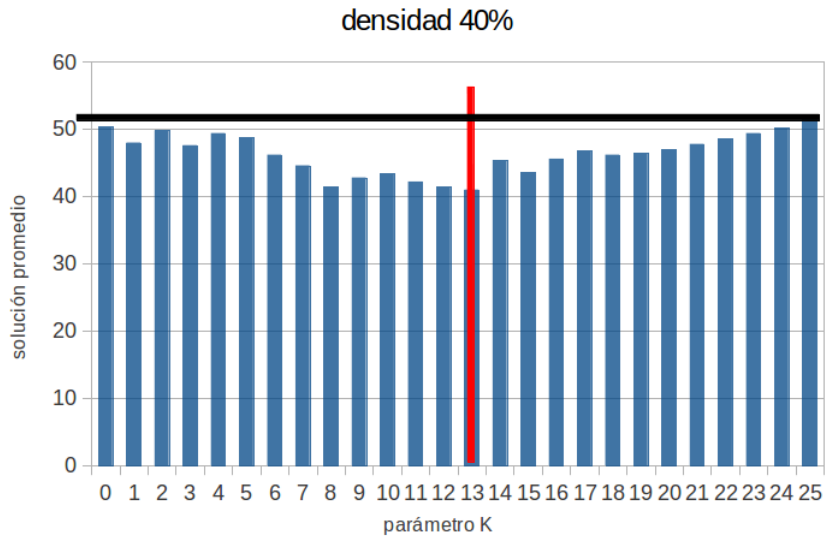

Fig. 5.8: HGA para grafos de densidad 40 %.

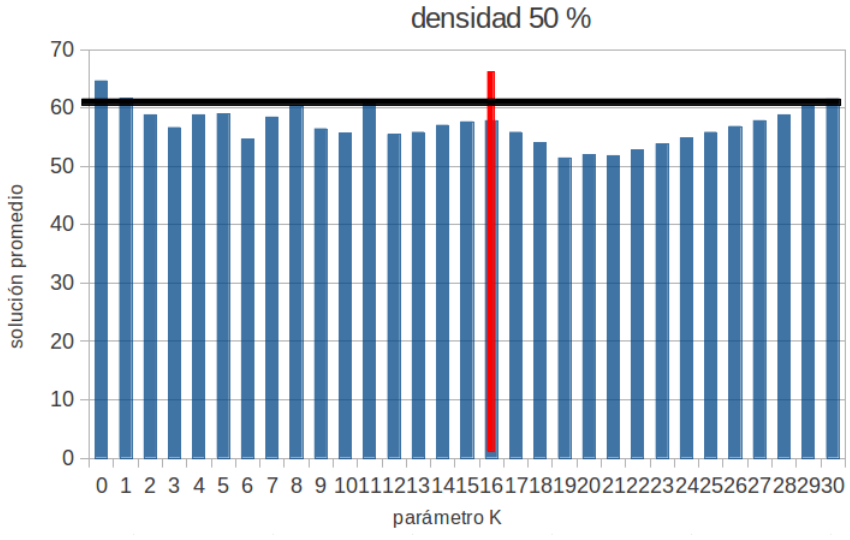

Fig. 5.9: HGA para grafos de densidad 50 %.

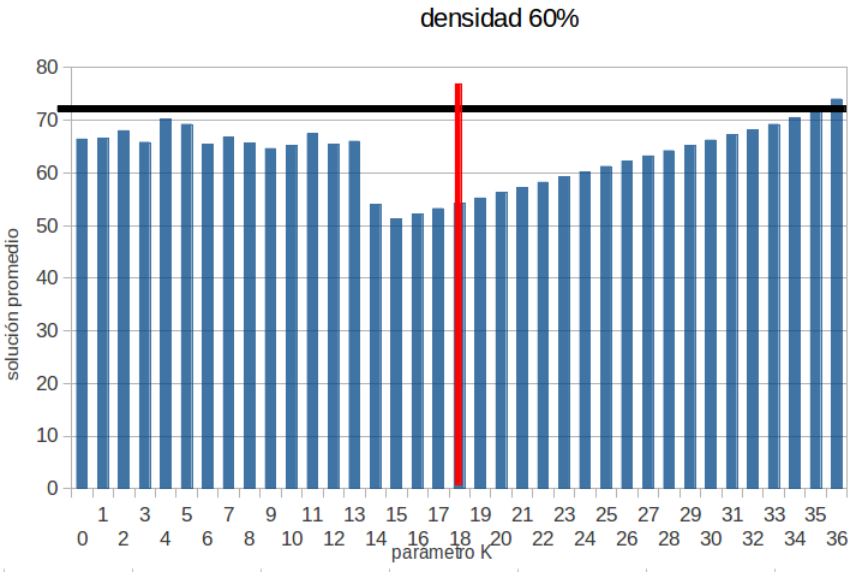

Fig. 5.10: HGA para grafos de densidad 60 %.

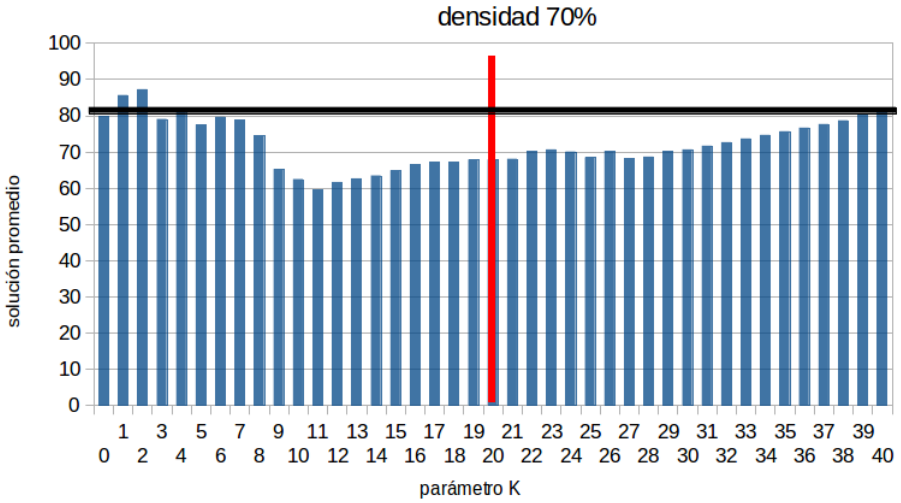

Fig. 5.11: HGA para grafos de densidad 70 %.

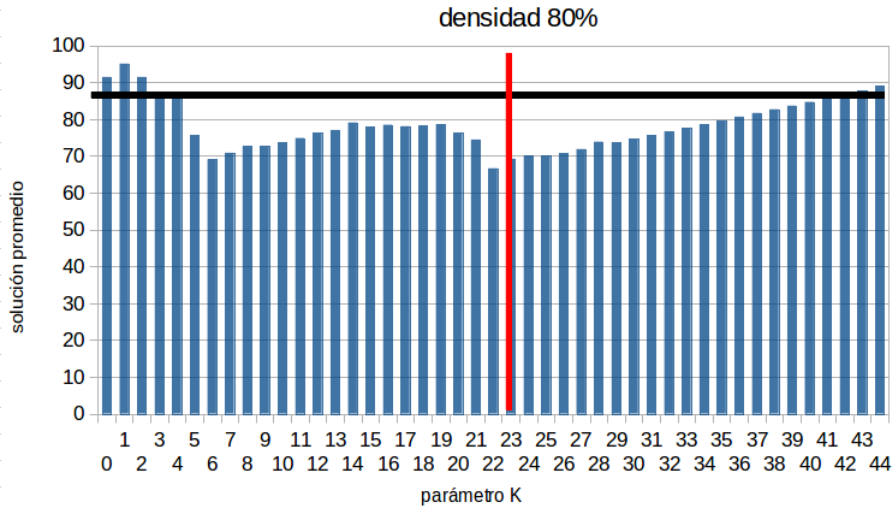

Fig. 5.12: HGA para grafos de densidad 80 %.

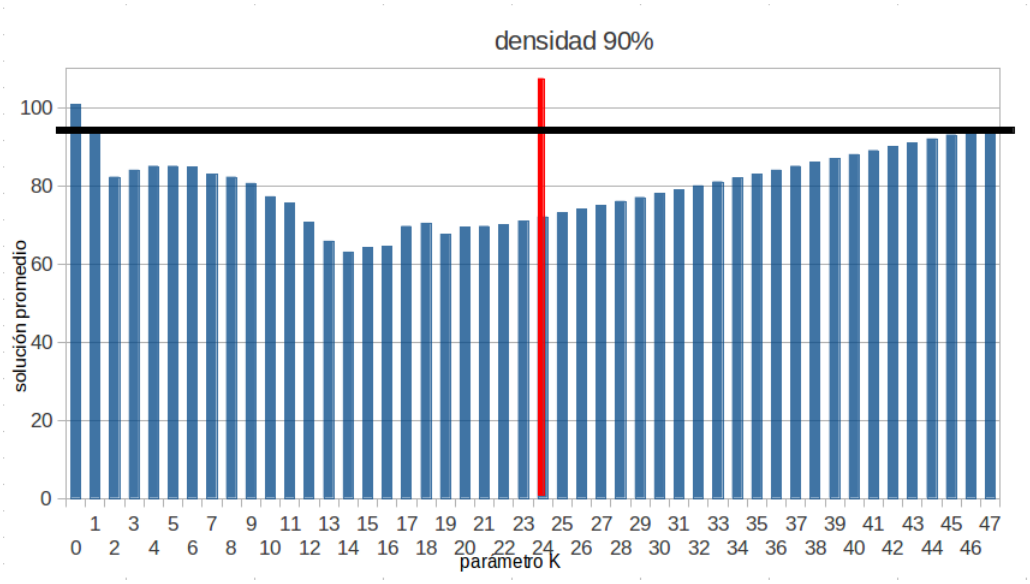

Fig. 5.13: HGA para grafos de densidad 90 %.

	HCV		HGA	
densidad	tiempo (seg)	cantidad de soluciones satisfactorias	tiempo (seg)	cantidad de soluciones satisfactorias
baja	0.089	13	0.553	15
media	0.429	8	4.433	15
alta	1.02	0	15.049	15

Tab. 5.5: Comparación entre la HCV y HGA.

En la Tab. 5.5 se observan los resultados globales para comparar la HVC con la HGA. Recordar que por cada grupo hay 15 grafos. Se va a tomar como una solución satisfactoria los etiquetados donde la etiqueta de máximo valor sea menor o igual a Δ. En este experimento se puede notar que en general la HCV no funciona bien para grafos de media a alta densidad (de hecho, para este conjunto de grafos de densidad alta de aristas, no encuentra buenas soluciones). En cambio, en la mayoría de los grafos de densidad baja se logra generar un etiquetado con máxima etiqueta menor o igual a Δ. Sin embargo, la HGA logra encontrar etiquetados satisfactorios para cada grafo. Se puede ver además, que los tiempos de ejecución de la HCV son menores que los tiempo de la HGA. Esto se debe a que en esta última heurística se hace un mayor esfuerzo para buscar un etiquetado satisfactorio ya que se recorre el rango de valores elegidos para K para generar un buen etiquetado.

5.2. Heurísticas primales

Además de la posibilidad de introducir cortes en los nodos del árbol del algoritmo de *Branch & Bound*, se pueden utilizar las denominadas heurísticas primales. Al igual que las heurísticas iniciales, el objetivo de las heurísticas primales es encontrar buenas soluciones enteras para el problema. La diferencia yace en que las heurísticas primales tratan de encontrar una solución entera en cada nodo del árbol utilizando la información provista por el nodo actual, es decir, la relajación lineal del nodo. En cambio, las iniciales sirven para contar con una solución entera al comienzo del algoritmo.

El punto de la relajación lineal en un nodo del árbol puede aportar información relevante para obtener una buena solución entera para el problema. En general, a medida que se avanza en el árbol se debería ir aproximando a una solución entera (o incluso a la solución óptima), por lo que se podría aprovechar esto para generar soluciones de calidad.

Por otra parte, se deben utilizar heurísticas primales rápidas porque en principio se ejecutan en cada uno de los nodos del árbol.

5.2.1. Heurística primal - modelo 1

Para esta heurística se va a utilizar una idea similar a la que se usó para la HGA inicial y al concepto del que surgieron las desigualdades 6.7 y 6.8. La intuición es que las variables δ^* de la relajación del nodo van *marcando* el orden de los colores de las aristas. Es decir, si para un nodo u los δ^*_{vur} son muy chicos, tendría mayor probabilidad de recibir un color chico.

En primer lugar se genera el grafo auxiliar como en la HGA y se etiqueta los nodos de ese grafo de la misma manera. En este momento, se obtiene un etiquetado de los nodos del grafo, por lo que falta etiquetar las aristas con la restricción de que se forme un coloreo de aristas. Sin embargo, aún no se está aprovechando la información que provee el nodo actual.

Como se mencionó, en la heurística se van a utilizar los valores de la relajación lineal del nodo de las variables δ^*, para guíar el orden de elección de las aristas para el etiquetado final (en lugar de tomar las aristas por el orden de lectura como en la HGA). El procedimiento es el siguiente: por cada arista se realiza la suma de los valores δ^* que la involucran. Luego, se procede a etiquetar las aristas tomándolas en orden creciente según esta suma. Como se mencionó, valores chicos de δ^*, y por lo tanto de la suma, podrían indicar que la arista tiene un color chico. Por eso se decide etiquetarla primero porque tiene más probabilidad de recibir una etiqueta de valor pequeño.

La experimentación con las heurísticas consistió en aportar una solución entera antes que empiece el *Branch & Bound* mediante la HGA y luego ejecutar el algoritmo de *Branch & Cut* para el modelo, utilizando la heurística primal en los nodos con el objetivo de bajar el valor de la cota superior (y por lo tanto, achicar el *gap*).

Nuevamente, se probó con los grafos de distintas densidades. Con los grafos de baja y media densidad no se observaron diferencias. Sin embargo, se mejoraron en una unidad las cotas en el 40 % de las instancias de los grafos de densidad alta.

5.2.2. Heurística primal - modelo 2

La heurística primal para el modelo 2 sigue la misma línea que la heurística primal del modelo 1 en el sentido que primero se etiquetan los vértices y después se etiquetan las aristas tomándolas en algún orden particular y cuidando que se genere un coloreo de aristas.

Entonces, la primera parte es etiquetar los vértices. Para esto, se asigna a cada vértice v la etiqueta j, donde j es la etiqueta de la variable x^*_{vj} con mayor valor en la relajación. Intuitivamente, si el valor de x_{vj} es el más alto, la etiqueta j es la que tiene más chances de ser asignada. De esta manera se etiquetan todos los vértices.

La segunda parte consiste en etiquetar las aristas y se decidió utilizar el siguiente criterio: para cada arista (u, v), primero se consideran los j tales que w^*_{uvj} sean las de máximo valor (de la misma manera que se realizó para los vértices) y a ese j se le suma el valor de las etiquetas de los vértices u y v conseguidas en el paso anterior. Entonces, por cada arista se obtiene un valor que representaría el color estimado que debería tener. El problema es que probablemente no se consiga un coloreo. Por este motivo, se ordenan las aristas según este valor calculado, de menor a mayor, y luego se las etiqueta con un algoritmo secuencial clásico de manera que se forme un coloreo.

Como en la experimentación del modelo 1, se utilizaron los grafos de distintas densidades, primero tomando una solución inicial con la HGA y luego ejecutando la heurística primal en los nodos del árbol del algoritmo de *Branch & Cut*. En este caso no se observaron diferencias con el algoritmo original, con excepción de unos pocos casos de grafos con densidad alta cuya cota superior bajó una unidad.

6. DESIGUALDADES VÁLIDAS - MODELO 1

En este apartado se presentan las desigualdades válidas que se identificaron para el primer modelo propuesto del problema. Recordemos que el modelo 1 tiene variables x_v y x_{uv} enteras que representan el valor de etiqueta asignada para cada vértice v y arista (u, v) del grafo. También posee variables δ_{ujv} utilizadas para indicar que aristas adyacentes (u, j), (j, v) del grafo reciben colores diferentes. Por último, la variable z mantiene el valor de la máxima etiqueta.

A continuación se exhiben distintas familias de desigualdades válidas para este modelo y su correspondiente explicación. Todas ellas fueron probadas en experimentos computacionales para establecer su efectividad como planos de corte.

`Familia 1` Las desigualdades tienen la siguiente forma

- Desigualdades tipo 1

$$x_u + x_{vu} + \delta_{uvr} \geq 3 \qquad \forall (u, v), (v, r) \in E(G) \tag{6.1}$$

- Desigualdades tipo 2

$$x_r + x_{vr} - \delta_{uvr} \geq 2 \qquad \forall (u, v), (v, r) \in E(G) \tag{6.2}$$

`Explicación` Sea $v \in V(G)$ y supongamos que tiene vértices adyacentes u y $r \in V(G)$ (podría tener más vértices adyacentes). Empecemos con las desigualdades 6.1. Notemos que si u y (v, u) reciben los dos una etiqueta con valor igual a 1, necesariamente el color $w(vu)$ tiene que ser menor que el color $w(vr)$ ya que ambas aristas comparten la etiqueta de v para constituir su color y como las etiquetas son mayores o iguales que 1, la arista (v, j) o el vértice j debe recibir una etiqueta mayor que 1 (para que los colores sean distintos). De esta manera, $w(vu) < w(vr)$.

Entonces, si $x_u = x_{vu} = 1$, fuerzan a que $\delta_{uvr} = 1$ que indica justamente que $w(vu) < w(vr)$.

El razonamiento es similar para las desigualdades 6.2, pero tomando en cuenta la otra arista. Si r y (v, r) reciben los dos una etiqueta con valor igual a 1, necesariamente el color $w(vu)$ tiene que ser mayor que el color $w(vr)$. De esta manera, si $x_r = x_{vr} = 1$, se fuerza a que $\delta_{uvr} = 0$ que indica justamente que $w(vu) > w(vr)$.

`Familia 2` La siguiente desigualdad describe la familia

- Desigualdades tipo 1

$$x_u + x_{vu} + \delta_{uvr} \leq 2\Delta \qquad \forall (u, v), (v, r) \in E(G) \tag{6.3}$$

- Desigualdades tipo 2

$$x_r + x_{vr} - \delta_{uvr} \leq 2\Delta - 1 \qquad \forall (u,v),(v,r) \in E(G) \tag{6.4}$$

`Explicación` Esta familia surgen del mismo concepto que las desigualdades 6.1 y 6.2, siguiendo la misma lógica en el caso en que se usen las etiquetas con valor de la cota máxima (es decir, etiquetas con valor Δ).

Sea $v \in V(G)$ con vértices adyacentes u y $r \in V(G)$. Veamos primero la desigualdad 6.3. Si u y (v,u) reciben los dos una etiqueta con valor Δ, necesariamente el color $w(vu)$ tiene que ser mayor que el color $w(vr)$. Luego, si $x_j = x_{vr} = \Delta$, fuerzan a que $\delta_{uvr} = 0$ que indica precisamente que $w(vu) > w(vr)$.

La desigualdad 6.4 analiza el caso en que la arista (r,v) reciba el mayor color. Es decir, si r y (v,r) reciben los dos una etiqueta con valor Δ, el color $w(vu)$ tiene que ser menor que el color $w(vr)$. En otras palabras, si $x_r = x_{vr} = \Delta$ el valor de $\delta_{uvr} = 1$ que indica que $w(vu) < w(vr)$.

Dado cualquier vértice v del grafo se pueden acotar los valores de los colores de las aristas incidentes a él, tanto inferior como superiormente. Como se tiene que cumplir con la condición de coloreo, las aristas tienen una configuración mínima para los valores que pueden recibir. Además, como el modelo restringe el valor máximo de cada etiqueta, también existe una configuración máxima de colores. Para las siguientes cuatro desigualdades, se va a considerar la descripción que muestra la Fig. 6.1, dónde el vértice v tiene k vértices adyacentes (es decir, el grado de v es igual a k).

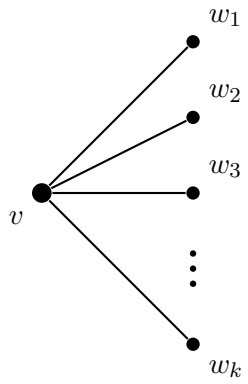

Fig. 6.1: Vecindad de un vértice v.

`Familia 3` Las desigualdades se enuncian de la siguiente forma

- Desigualdades de cota inferior

$$\sum_{i=1}^{k} x_{vw_i} + \sum_{i=1}^{k} x_{w_i} \geq \frac{k^2+3k}{2} \qquad \forall v \in V(G) \tag{6.5}$$

- Desigualdades por cota superior

$$\sum_{i=1}^{k} x_{vw_i} + \sum_{i=1}^{k} x_{w_i} \leq k(2\Delta+1) - \frac{k(k+1)}{2} \qquad \forall v \in V(G) \tag{6.6}$$

`Explicación` Sea $v \in V(G)$ cualquier vértice de G y supongamos que $w_1, w_2, \ldots, w_k$ son todos los vértices adyacentes a v. Es decir, $\{(v,w_1),(v,w_2),\ldots,(v,w_k)\} \subseteq E(G)$. Observemos que estas aristas tienen que recibir colores diferentes para poder conformar un coloreo de aristas (al compartir el vértice v, todas estas aristas son adyacentes entre sí).

Analicemos primero la desigualdad 6.5. La mínima asignación de colores que se puede hacer para las aristas es $\{3,4,\ldots,k+2\}$. Sin embargo, como la etiqueta de v está presente en los colores de cada una de las aristas, es equivalente decir que $\{2,3,\ldots,k+1\}$ es la mínima asignación que se puede hacer para la suma de las etiquetas de cada arista (v,w_i) y su vértice correspondiente w_i. Por lo tanto la suma de estas etiquetas tiene que ser necesariamente mayor o igual que la suma de este conjunto de asignaciones:

$$\begin{aligned}\sum_{i=1}^{k} x_{vw_i} + \sum_{i=1}^{k} x_{w_i} &\geq 2+3+\cdots+k+1 \\ &= \sum_{l=1}^{k}(l+1) \\ &= \frac{k(k+1)}{2} + k \\ &= \frac{k^2+3k}{2}\end{aligned}$$

De manera similar se puede deducir la desigualdad 6.6, pero acotando superiormente los colores de las aristas. Es decir, esta desigualdad toma en cuenta la máxima asignación de colores que se puede realizar para las aristas incidentes a v. Ya que la cota superior para las etiquetas de vértices y aristas en el modelo es Δ, la máxima asignación posible es $\{3\Delta, 3\Delta-1, \ldots, 3\Delta-k+1\}$. Como la etiqueta de v está presente en los colores de cada una de las aristas, resulta lo mismo tomar $\{2\Delta, 2\Delta-1, \ldots, 2\Delta-k+1\}$ como la máxima asignación que se puede hacer para la suma de las etiquetas de cada (v,w_i) y el vértice correspondiente w_i. Entonces, la suma de estas etiquetas tiene que ser menor o igual que la suma de este conjunto de asignaciones:

$$\sum_{i=1}^{k} x_{vw_i} + \sum_{i=1}^{k} x_{w_i} \leq 2\Delta + (2\Delta - 1) + \cdots + (2\Delta - k + 1)$$
$$= \sum_{l=1}^{k} (2\Delta - l + 1)$$
$$= \sum_{l=1}^{k} (2\Delta + 1) - \sum_{l=1}^{k} l$$
$$= k(2\Delta + 1) - \frac{k(k+1)}{2}$$

`Familia 4` Las desigualdades tienen la siguiente forma

- Desiguadades por cota inferior

$$x_{vw_i} + x_{w_i} \geq 2 + \sum_{j=1}^{i-1} \delta_{w_j v w_i} + \sum_{j=i+1}^{k} (1 - \delta_{w_i v w_j}) \qquad \forall v \in V(G),\ i = 1, \ldots, k, \forall w_i \in N(v) \tag{6.7}$$

- Desigualdades por cota superior

$$x_{vw_i} + x_{w_i} \leq 2\Delta - \sum_{j=1}^{i-1} (1 - \delta_{w_j v w_i}) - \sum_{j=i+1}^{k} \delta_{w_i v w_j} \qquad \forall v \in V(G),\ i = 1, \ldots, k, \forall w_i \in N(v) \tag{6.8}$$

`Explicación` Estas desigualdades son una generalización de las familias 1 y 2 propuestas, ya que en lugar de analizar solo dos aristas incidentes a un vértice v, toman en cuenta todas las aristas incidentes a v simultáneamente.

En este caso se acota individualmente cada arista (v, w_i) aprovechando que se cuenta con variables que indican si el color de una arista es mayor que el color de otra arista adyacente. Veamos las desigualdades 6.7. En principio, el color de una arista como mínimo es 3. Pero esta cota inferior se incrementa en uno por cada arista adyacente que posea un color menor. Esta información la codifica las variables δ_{uvr}. Al igual que en la familia anterior, no es necesario tomar en cuenta el valor de la etiqueta de v porque es común al color de cada arista. Entonces, se elimina del miembro izquierdo la variable x_v y se arranca a contar de 2 en adelante por la suma del valor de la etiqueta de la arista y del otro vértice extremo.

Una observación: en el lado derecho se separa en dos sumatorias simplemente por el orden de lectura del grafo. Con el orden de lectura $\delta_{w_j v w_i}$, significa que si el valor de esta variable es 1 el color de (w_j, v) es menor que el color de (v, w_i) y por lo tanto se debe

incrementar en uno el valor de la cota inferior. De la misma manera, con un orden de lectura $\delta_{w_i v w_j}$, si el valor de esta variable es 0 el color de (w_i, v) es mayor que el color de (v, w_j), con lo que se debe sumar uno a la cota inferior.

Se puede ver que este tipo de desigualdades implican la desigualdades 6.5. Basta con sumar las desigualdades por cada arista incidente a v. Esto equivale a sumar sobre cada vértice adyacente a v en ambos lados de la desigualdad:

$$\begin{aligned}
\sum_{i=1}^{k}(x_{vw_i} + x_{w_i}) &\geq \sum_{i=1}^{k}\left(2 + \sum_{j=1}^{i-1}\delta_{w_j v w_i} + \sum_{j=i+1}^{k}(1 - \delta_{w_i v w_j})\right) \\
&= \sum_{i=1}^{k}\left(2 + \sum_{j=1}^{i-1}\delta_{w_j v w_i} + \sum_{j=i+1}^{k}1 - \sum_{j=i+1}^{k}\delta_{w_i v w_j}\right) \\
&= \sum_{i=1}^{k}\left(2 + k - i + \sum_{j=1}^{i-1}\delta_{w_j v w_i} - \sum_{j=i+1}^{k}\delta_{w_i v w_j}\right) \\
&= \sum_{i=1}^{k}(2 + k - i) + \sum_{i=1}^{k}\sum_{j=1}^{i-1}\delta_{w_j v w_i} - \sum_{i=1}^{k}\sum_{j=i+1}^{k}\delta_{w_i v w_j} \\
&= 2k + k^2 - \frac{k(k+1)}{2} + \sum_{s=1}^{k}\sum_{l=1}^{s-1}\delta_{w_l v w_s} - \sum_{i=1}^{k}\sum_{j=i+1}^{k}\delta_{w_i v w_j} \\
&= 2k + k^2 - \frac{k^2+k}{2} + \sum_{1\leq l<s\leq k}\delta_{w_l v w_s} - \sum_{1\leq i<j\leq k}\delta_{w_i v w_j} \\
&= \frac{4k + 2k^2 - k^2 - k}{2} \\
&= \frac{k^2 + 3k}{2}
\end{aligned}$$

Finalmente:

$$\sum_{i=1}^{k}x_{vw_i} + \sum_{i=1}^{k}x_{w_i} \geq \frac{k^2+3k}{2}$$

que es exactamente la forma de la desigualdad 6.5.

Las desigualdades 6.8 siguen la misma línea que las desigualdades 6.7, con la diferencia de que se acota superiormente el color de la arista. Al igual que en las desigualdades 6.7 se acota cada arista (v, w_i) sabiendo cuáles aristas adyacentes tienen un color mayor y cuáles tienen color menor. Como se acota superiormente se debe decrementar en uno la cota por cada arista adyacente que presente un color mayor. Recordar que en el lado derecho de la desigualdad se separa en dos sumatorias por el orden de lectura del grafo.

Con una cuenta similar a la empleada para demostrar que las desigualdades 6.7 implican a las desigualdades 6.5 se puede ver que las desigualdades 6.8 implican a las desigualdades 6.6:

$$\begin{aligned}
\sum_{i=1}^{k}(x_{vw_i}+x_{w_i}) &\leq \sum_{i=1}^{k}\left(2\Delta-\sum_{j=1}^{i-1}(1-\delta_{w_jvw_i})-\sum_{j=i+1}^{k}\delta_{w_ivw_j}\right)\\
&=\sum_{i=1}^{k}\left(2\Delta-\sum_{j=1}^{i-1}1+\sum_{j=i+1}^{k}\delta_{w_jvw_i}-\sum_{j=i+1}^{k}\delta_{w_ivw_j}\right)\\
&=\sum_{i=1}^{k}\left(2\Delta-i+1+\sum_{j=1}^{i-1}\delta_{w_jvw_i}-\sum_{j=i+1}^{k}\delta_{w_ivw_j}\right)\\
&=\sum_{i=1}^{k}(2\Delta-i+1)+\sum_{i=1}^{k}\sum_{j=1}^{i-1}\delta_{w_jvw_i}-\sum_{i=1}^{k}\sum_{j=i+1}^{k}\delta_{w_ivw_j}\\
&=2\Delta k-\frac{k(k+1)}{2}+k+\sum_{s=1}^{k}\sum_{l=1}^{s-1}\delta_{w_lvw_s}-\sum_{i=1}^{k}\sum_{j=i+1}^{k}\delta_{w_ivw_j}\\
&=k(2\Delta+1)-\frac{k(k+1)}{2}+\sum_{1\leq l<s\leq k}\delta_{w_lvw_s}-\sum_{1\leq i<j\leq k}\delta_{w_ivw_j}\\
&=k(2\Delta+1)-\frac{k(k+1)}{2}
\end{aligned}$$

Luego:

$$\sum_{i=1}^{k}x_{vw_i}+\sum_{i=1}^{k}x_{w_i}\leq k(2\Delta+1)-\frac{k(k+1)}{2}$$

que es justamente la descripción de la desigualdad 6.7.

`Análisis` Existen a lo sumo $2m$ desigualdades para cada tipo de desigualdad de esta familia. Por este motivo, se decidió incluir explícitamente cada tipo de desigualdad al modelo en la experimentación para ver si el valor de la relajación lineal se incrementa con estas desigualdades como parte del modelo. Además cabe destacar que no se incluye la experimentación con las familias 1, 2 y 3, porque esta familia de desigualdades es más fuerte que cada una de dichas familias. Sin embargo, no se descarta usar la familia 3 ya que tiene menos desigualdades.

Siguiendo con la experimentación con los grafos de distintas densidades, se observó que el valor óptimo de la relajación del modelo original no cambia al incluir las desigualdades 6.8. La razón es que el valor 2Δ suele ser un número grande y los valores de δ^* muy pequeños. Entonces, el punto óptimo de la relajación del modelo original también satisface las nuevas desigualdades.

En cambio, la inclusión de las desigualdades 6.7 al modelo hacen que los valores de la relajación mejoren. Para registar las mejoras, en la Tab. 6.1 se muestran los *gaps* obtenidos para el modelo original y los que se obtienen al incluir este tipo de desigualdades.

Densidad	*gap* (%) promedio del modelo original	*gap* (%) promedio del modelo original + desigualdades
Baja	67.56	36.72
Media	81.76	42.45
Alta	86.96	46.84

Tab. 6.1: Comparación entre los *gaps* del modelo original y el modelo resultante de añadir las desigualdades 6.7.

Recordemos que el valor de la función objetivo en el óptimo de la relajación para el modelo original es 1 para cualquier grafo de entrada, por lo que es lógico que tenga *gaps* altos. Sin embargo, al incluir esta familia de desigualdades se puede observar que se reduce el *gap* para cada densidad.

Al analizar la estructura de las desigualdades 6.7, se puede deducir una cota inferior para el óptimo de la relajación del modelo con estas desigualdades incluidas. Veamos el valor de dicha cota. Para calcular la cota inferior se usarán las desigualdades 6.5 por simplicidad en las cuentas, ya que se demostró que las desigualdades 6.7 son más fuertes que las desigualdades 6.5. Sea v un vértice de grado máximo, entonces

$$\sum_{i=1}^{\Delta} x_{vw_i} + \sum_{i=1}^{\Delta} x_{w_i} \geq \frac{\Delta^2 + 3\Delta}{2}$$

Por la definición del modelo 1, sabemos que $z \geq x_u$, $\forall u \in V(G)$ y $z \geq x_{ur}$, $\forall (u,r) \in E(G)$, por lo tanto

$$\begin{aligned}
\sum_{i=1}^{\Delta} z + \sum_{i=1}^{\Delta} z \geq \sum_{i=1}^{\Delta} x_{vw_i} + \sum_{i=1}^{\Delta} x_{w_i} \geq \frac{\Delta^2 + 3\Delta}{2} &\Longrightarrow \\
2\sum_{i=1}^{\Delta} z \geq \frac{\Delta(\Delta+3)}{2} &\Longrightarrow \\
2\Delta z \geq \frac{\Delta(\Delta+3)}{2} &\Longrightarrow \\
z \geq \frac{\Delta+3}{4}&
\end{aligned}$$

Esto indica que el valor óptimo de la relajación lineal será al menos $\frac{\Delta+3}{4}$. En las pruebas realizadas siempre se cumplió por igualdad.

Las siguientes dos familias de desigualdades surgen de estudiar los ciclos que contiene el grafo. En particular, nos vamos a concentrar en ciclos de tres vértices por cuestiones de complejidad temporal, pero como se observa en la descripción de las desigualdades se puede generalizar a ciclos de cualquier longitud.

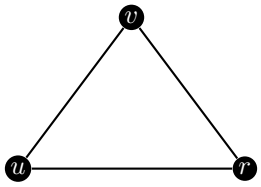

Fig. 6.2: ciclo de longitud 3.

`Familia 5` Las desigualdades se enuncian de la siguiente forma

Sea cualquier ciclo $C = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{l-1}, v_l), (v_l, v_1)\}$.

- Si la longitud l del ciclo es par

$$\sum_{(u,r)\in C} x_{ur} + 2\sum_{i=1}^{l} x_{v_i} \geq (3+4)\frac{l}{2} \tag{6.9}$$

- Si la longitud l del ciclo es impar

$$\sum_{(u,r)\in C} x_{ur} + 2\sum_{i=1}^{l} x_{v_i} \geq (3+4)\frac{l-1}{2} + 5 \tag{6.10}$$

`Explicación` El objetivo de esta desigualdad es aportar una cota inferior de los colores que pueden recibir las aristas de un ciclo. Primero observemos que el miembro de la izquierda de las desigualdades corresponde con la suma de los colores de las aristas pertenecientes al ciclo. Se suma las etiquetas de cada arista y dos veces las etiquetas de los vértice, ya que la etiqueta de cada vértice está presente en los colores de las dos aristas que une.

Veamos primero el caso en que la longitud del ciclo sea par. La mínima asignación de colores que se puede hacer es alternar el color 3 y 4 a lo largo del ciclo. De esta manera, la mitad de las aristas del ciclo recibirían el color 3 y la otra mitad el color 4.

En el caso de que la longitud del ciclo sea impar, el procedimiento para alternar los colores 3 y 4 es similar con la excepción de que la última arista que se debe colorear tiene adyacentes aristas que recibirían una color 3 y la otra color 4. Por lo tanto, debemos emplear un nuevo color para dicha arista y el menor que podemos usar es el color 5.

`Análisis` Como se mencionó, dada la complejidad de encontrar los ciclos de cualquier longitud en un grafo, se decidió trabajar con ciclos de longitud 3 como el que se observa en la Fig. 6.2. Por lo tanto, si los vértices u, v y r conforman el ciclo, las desigualdades son de la siguiente forma:

$$x_{uv} + x_{vr} + x_{ru} + 2x_u + 2x_v + 2x_r \geq 3 + 4 + 5 = 12$$

Además, para la experimentación se incluyeron las desigualdades en el modelo original para ver si mejoraba el valor de la relajación lineal. Se pudo observar que en los vértices v y aristas (u, v) que pertenecen al ciclo, $x_v^* = x_{uv}^* = \frac{4}{3}$ para cumplir con este tipo de desigualdades. Para el resto de los vértices y aristas que no pertenecen a ciclos de longitud 3, los valores de sus respectivas variables seguían siendo 1 como en el óptimo de la relajación del modelo original. Los δ^* se acomodaban para que se siguieran cumpliendo las restricciones del modelo original. De esta manera, para los grafos que contienen ciclos de longitud 3 el valor de la relajación lineal será $\frac{4}{3}$ y para el resto de los grafos continuará siendo 1.

`Familia 6` Esta familia se describe a través de la siguiente desigualdad

Sea cualquier ciclo $C = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{l-1}, v_l), (v_l, v_1)\}$. Para esta familia tenemos dos clases de desigualdades, que indican un tipo particular de transitividad. Por simplicidad de notación suponemos que existen tanto δ_{uvr} como δ_{rvu}.

- transitividad por menor

$$\delta_{v_1v_2v_3} + \delta_{v_2v_3v_4} + \cdots + \delta_{v_{l-1}v_lv_1} \leq l - 2 + \delta_{v_2v_1v_l} \tag{6.11}$$

- transitividad por mayor

$$\delta_{v_1v_2v_3} + \delta_{v_2v_3v_4} + \cdots + \delta_{v_{l-1}v_lv_1} \geq \delta_{v_2v_1v_l} \tag{6.12}$$

`Explicación` Como se mencionó, estas desigualdades aportan una noción de transitividad en los colores de un ciclo.

La desigualdad 6.11 indica que si $w(v_1v_2) < w(v_2v_3) < \ldots < w(v_{l-1}v_l) < w(v_lv_1)$, necesariamente $w(v_1v_2) < w(v_lv_1)$. Observemos que si se cumple la condición mencionada, todos las variables δ del miembro izquierdo de la desigualdad estarán en 1. Como hay $l-1$ variables en el lado izquierdo, quedará de la siguiente forma:

$$l - 1 \leq l - 2 + \delta_{v_2v_1v_l} \iff 1 \leq \delta_{v_2v_1v_l}$$

Es decir, fuerza a que la variable $\delta_{v_2v_1v_l}$ sea 1. Esto es precisamente lo que se deseaba representar, ya que si $\delta_{v_2v_1v_l}$ es 1 significa que el color de la arista (v_1, v_2) es menor que el color de la arista (v_l, v_1).

Análogamente, la desigualdad 6.12 indica que si $w(v_1v_2) > w(v_2v_3) > \ldots > w(v_{l-1}v_l) > w(v_lv_1)$, se tiene que cumplir $w(v_1v_2) > w(v_lv_1)$. Luego, si se satisface la condición deseada, todos las variables δ del miembro izquierdo de la desigualdad tomarán valor 0 y se tendrá lo siguiente:

$$0 \geq \delta_{v_2v_1v_l}$$

Es decir, fuerza a que la variable $\delta_{v_2 v_1 v_l}$ sea 0. Esto es justamente lo que se quería representar, ya que si $\delta_{v_2 v_1 v_l}$ es 0 significa que el color de la arista (v_1, v_2) es mayor que el color de la arista (v_l, v_1).

`Análisis` En este caso también se decidió trabajar con ciclos de longitud 3 para evitar la complejidad de revisar todos los ciclos de cualquier longitud del grafo. Entonces, si los vértices u, v y r son los que forman el ciclo, las desigualdades son las siguientes:

1. $\delta_{uvr} + \delta_{vru} \leq 1 + \delta_{vur}$
2. $\delta_{uvr} + \delta_{vru} \geq \delta_{vur}$

Se puede ver fácilmente que el óptimo de la relajación cumple con estas desigualdades. Como los valores δ^* son todos iguales en el óptimo de la relajación del modelo original, de la primera desigualdad se obtiene que $\delta^* \leq 1$ y de la segunda desigualdad se deduce que $\delta^* \geq 0$ que son restricciones propias del modelo original.

Por último, también se puede notar que los colores de las aristas incidentes sobre un nodo configuran un ordenamiento lineal, dado por los valores que toman las variables δ. Por lo tanto se pueden extraer y reformular algunas desigualdades propuestas para el problema de ordenamiento lineal. Las primeras dos familias que se presentan están orientadas a prevenir ciclos en el orden de los colores. Por ejemplo, si se tiene una situación como se muestra en la Fig. 6.3, si las variables δ indican que el color de la arista (u, v) es menor que el de la arista (r, v) y el color de la arista (r, v) es menor que el de (z, v) se debe garantizar que también indiquen que el color de (u, v) es menor que el de (z, v).

Las últimas desigualdades, denominadas *k-fence*, fueron estudiadas por Grötschel [11] [12] como facetas del problema de ordenamiento lineal.

Nuevamente, se va a asumir que existen tanto δ_{uvr} como δ_{vru} para simplificar la notación.

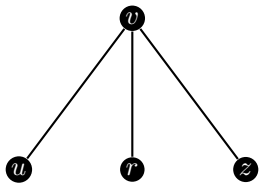

Fig. 6.3: no puede existir ciclos en el orden de los colores.

`Desigualdades anti-ciclo 1` La familia se representa de la siguiente manera

$$\delta_{uvr} + \delta_{rvz} \leq 1 + \delta_{uvz} \qquad \forall (u, v), (v, r), (v, z) \in E(G) \tag{6.13}$$

`Explicación` Sea $v \in V(G)$ y supongamos que tiene vértices adyacentes u, r y $z \in V(G)$. Es fácil de ver que si $w(vu) < w(vr)$ y $w(vr) < w(vz)$, por transitividad $w(vu)$ tiene que ser menor que $w(vz)$.

Por la formulación del modelo, el hecho de que $w(vu) < w(vr)$ se refleja en que $\delta_{uvr} = 1$. De la misma manera, si $w(vr) < w(vz)$ implica que $\delta_{rvz} = 1$. Entonces, si se dan estas condiciones a la vez, la desigualdad fuerza que $\delta_{uvz} = 1$. Esto indica que $w(vu) < w(vz)$, que es justamente lo que se quería representar.

`Desigualdades anti-ciclo 2` Esta familia está dada por la siguiente desigualdad

$$\delta_{uvr} + \delta_{rvz} \geq \delta_{uvz} \qquad \forall (u,v),(v,r),(v,z) \in E(G) \tag{6.14}$$

`Explicación` Esta familia describe lo mismo que la familia anterior, pero analizando la transitividad de los colores de las aristas en el otro sentido.

Sea $v \in V(G)$ y supongamos que tiene vértices adyacentes u, r y $z \in V(G)$. Si $w(vu) > w(vr)$ y $w(vr) > w(vz)$, por transitividad $w(vu)$ tiene que ser mayor que $w(vz)$.

Por la formulación del modelo, el hecho de que $w(vu) > w(vr)$ se refleja en que $\delta_{uvr} = 0$. De la misma manera, si $w(vr) > w(vz)$, entonces $\delta_{rvz} = 0$. Por lo tanto, si se dan estas condiciones a la vez, la desigualdad fuerza que $\delta_{uvz} = 0$. Esto indica que $w(vu) > w(vz)$, que es lo que se quería representar.

`Análisis de Desigualdades anti-ciclo 1 y 2` Para estas desigualdades también se puede ver que el óptimo de la relajación del modelo original las cumple.

Veamos primero la desigualdad anti-ciclo 1. Como los valores δ^* son todos iguales en óptimo de la relajación del modelo original, de la desigualdad anti-ciclo 1 se deriva que $\delta^* \leq 1$ que siempre es cierto por las restricciones del modelo.

El análisis es análogo para la desigualdad anti-ciclo 2. En el óptimo de la relajación del modelo original los valores de δ^* son todos iguales, por lo tanto, de la desigualdad anti-ciclo 2 se deduce que $\delta^* \geq 0$ lo cual es siempre cierto por las restricciones del modelo.

`Desigualdades k-fence` Esta familia está dada por la siguiente desigualdad

Sean los conjutos de vértices disjuntos $U = \{u_1, \ldots, u_k\}$, $L = \{l_1, \ldots, l_k\} \subseteq V(G)$ de cardinalidad $3 \leq k \leq \frac{n}{2}$. Se denomina *k-fence* al conjunto de arcos

$$A = \bigcup_{i=1}^{k} \left(\{(u_i, l_i)\} \cup \{(l_i, u_j) : j \in \{1, \ldots, k\}, j \neq i\} \right)$$

Para adaptar este tipo de desigualdades a nuestro problema, se necesita que cada vértice de los conjuntos U y L estén unidos con un vértice en común v. Luego, la desigualdad *k-fence* para el *PCAET* es:

$$\sum_{(u_i,l_i) \in A} \delta_{u_i v l_i} + \sum_{(l_i,u_j) \in A} \delta_{l_i v u_j} \leq k^2 - k + 1 \tag{6.15}$$

`Análisis` Dada la alta complejidad de obtener todas las desigualdades de esta familia y en base a los resultados computacionales reportados por Grötschel [12], sólo se generan las desigualdades *3-fence* mediante una rutina de separación en un algoritmo de planos de corte. Como en las pruebas anteriores, se usaron los grafos de 15 nodos y diferentes densidades. Para esta experimentación se observó que no se introducen cortes y por lo tanto el valor óptimo de la relajación del modelo no cambia. En este caso se deduce que el punto óptimo de la relajación del modelo también satisface las desigualdades *3-fence* generadas en la rutina de separación ya que los valores de δ^* son muy pequeños.

La última experimentación de este capítulo consistió en combinar todas las desigualdades válidas. Es decir, se incluyeron las desigualdades de las familias 4 (recordar que no se utilizan las familias 1, 2 y 3 ya que son más débiles), 5, 6, anti-ciclo 1 y 2 explícitamente en el modelo y sobre este nuevo modelo se ejecutó un algoritmo de planos de corte generando las desigualdades *3-fence* mediante una rutina de separación.

En esta prueba se obtuvieron los mismos resultados en los valores de las relajaciones que los obtenidos al agregar únicamente las desigualdades 6.7 al modelo original, a pesar de que se introducen cortes de las desigualdades *3-fence*.

7. DESIGUALDADES VÁLIDAS - MODELO 2

A continuación se presentan las desigualdades válidas que se lograron determinar para el segudo modelo del problema. El modelo 2 tiene variables x_{vj}, w_{uvj} binarias que indican si el vértice v y arista (u, v) reciben la etiqueta j, respectivamente. La variable z_{uvk} indica si la arista (u, v) tiene color k. Por último, la variable y mantiene el valor de la máxima etiqueta.

En [9], se propone utilizar las denominadas desigualdades *blossom* para resolver la formulación multi-matching para coloreo de aristas mediante un algoritmo de planos de corte que genere dichas desigualdades cuando se lo requiera. Esta familia de desigualdades puede ser adaptada para resolver la formulación del segundo modelo en el algoritmo *Branch & Cut*.

`Desigualdades blossom` Esta familia de desigualdades se enuncia de la siguiente forma

$$\sum_{(u,v)\in E(G[U])} z_{uvk} \leq \left\lfloor \frac{|U|}{2} \right\rfloor \qquad \forall k \in \{3, \ldots, 3\Delta\},\, U \subseteq V(G) \tag{7.1}$$

`Explicación` Las variables z_{uvk} se utilizan para indicar el color que reciben las aristas y para establecer que efectivamente se genere un coloreo de aristas. En el coloreo de aristas generado, el conjunto de aristas que define un color particular forman un matching. Esta desigualdad relaciona coloreo de aristas con matching máximo.

Sea G un grafo. Dado un subconjunto de vértices $U \subseteq V(G)$, la máxima cantidad de aristas que se pueden colorear con un color determinado en el grafo inducido de G por U (o sea, $G[U]$) está dado por el tamaño del matching máximo. Es decir, si M es un matching máximo para $G[U]$, entonces se puede colorear a lo sumo $|M|$ aristas con un color.

La desigualdad *blossom* permite establecer una cota superior para el matching máximo de $G[U]$ para cualquier $U \subseteq V(G)$ y por lo tanto para la cantidad de aristas que se pueden colorear con cualquier color k en $G[U]$. No se usa $|M|$ porque obtener la dimensión del matching máximo es un problema `NP-completo`.

`Análisis` El problema de esta familia de desigualdades es que la complejidad de conseguir todos los subconjuntos $U \subseteq V(G)$ es exponencial, por lo que se decidió desarrollar una heurística para generar un subconjunto de las desigualdades. La intuición de la heurística es la siguiente: dado un conjunto $U \subseteq V(G)$ sería deseable que exista la mayor cantidad posible de aristas que unen los vértices de U porque habría más probabilidades que la desigualdad corte el punto óptimo de la relajación. Por este motivo, se procede a buscar heurísticamente *cliques* en G (subgrafos completos) y generar las desigualdades *blossom* a partir de las *cliques* encontradas.

La experimentación consiste en ejecutar un algoritmo de planos de corte agregando todas las desigualdades *blossom* generadas mediante un algoritmo de separación heurística. A pesar de que en varias instancias se introducen cortes, en ninguna se lograba subir el óptimo de la relajación lineal del modelo original.

`Desigualdades etiqueta necesaria` La siguiente desigualdad describe la familia

$$\sum_{k=3}^{3\Delta} \lceil k/3 \rceil \, z_{uvk} \leq y \qquad \forall (u,v) \in E(G) \tag{7.2}$$

`Explicación` Vale la pena recordar que suma de las variables z_{uvk} es 1 para cada arista ya que cada una puede recibir un solo color. Supongamos entonces que la arista (u, v) recibió el color k y por lo tanto la variable $z_{uvk} = 1$. Esta desigualdad indica que para que suceda esta situación se debe utilizar necesariamente una etiqueta con valor mayor o igual a $\lceil k/3 \rceil$. Es decir, si la arista (u, v) tiene color k, entonces el alguno entre u, v o (u, v) debe tener una etiqueta con valor mayor o igual a $\lceil k/3 \rceil$.

`Análisis` Para analizar la performance de esta familia se decidió incluirlas explícitamente en el modelo y calcular las relajaciones lineales de la nueva formulación, ya que son m desigualdades nuevas. Al hacer el experimento se puede observar que la inclusión de estas desigualdades suben el valor del óptimo de la relajación del modelo original. La Tab. 7.1 muestra la comparación entre los *gaps* obtenidos para el modelo original y los obtenidos al incorporar las desigualdades. Se puede notar que los *gaps* promedios son menores para cada densidad al agregar al modelo las desigualdades 7.2.

Densidad	*gap* (%) promedio del modelo original	*gap* (%) promedio del modelo original + desigualdades
Baja	59.33	36.79
Media	76.33	49.65
Alta	82.95	56.01

Tab. 7.1: Comparación entre los *gaps* del modelo original y el modelo resultante de añadir las desigualdades 7.2.

Finalmente, se hace el análisis utilizando las dos desigualdades identificadas para el modelo. Es decir, la experimentación consiste en ejecutar un algoritmo de planos de corte agregando las desigualdades 7.1 con un algoritmo de separación sobre el modelo original con las desigualdades 7.2 incorporadas explícitamente.

Se obtuvieron los mismos resultados que la experimentación realizada individualmente para las desigualdades 7.2. Aunque se introducían cortes con las desigualdades 7.1, no se lograba ajustar más los *gaps*.

8. RESULTADOS

El propósito de implementar buenas heurísticas iniciales y primales, e identificar y estudiar desigualdades válidas para cada modelo es desarrollar un buen algoritmo *Branch & Cut* y compararlo con el algoritmo original. En el presente capítulo se exponen los resultados que se obtuvieron al experimentar con todos los elementos analizados para cada modelo.

Dado que las cotas superiores mejoraron con la ejecución de la HGA y la incorporación de las heurísticas primales en los algoritmos, se usarán siempre para las pruebas. La HGA se usará de la siguiente manera: si *bestSol* es el valor de la etiqueta máxima usada en la solución encontrada con la HGA para una instancia, se parte de un modelo donde $bestSol - 1$ se utiliza como cota para el valor de las etiquetas en dicha instancia. Se realiza de esta forma para tratar de encontrar una solución cuya máxima etiqueta tenga valor $bestSol - 1$ o menos, ya que conocemos una solución con etiqueta máxima *bestSol* gracias a la HGA.

Además, para la experimentación se decidió utilizar las estrategias de *branching* y de selección de próximo nodo a explorar que vienen por defecto en el paquete de CPLEX y establecer como tiempo máximo de resolución 1 hora.

8.1. Resultados - Modelo 1

Para el modelo 1 se identificó una serie de familias de desigualdades válidas y se analizó su rendimiento en el contexto de la relajación inicial. Antes de plantear una configuración final para el algoritmo *Branch & Cut* para este modelo, se desea ver el rendimiento de las desigualdades en la ejecución del algoritmo *Branch & Cut*. Luego, se comparará el algoritmo resultante con el algoritmo por defecto que utiliza CPLEX.

En el apartado de desigualdades válidas del modelo 1 las desigualdades 6.7 se destacaron por su rendimiento. Como no son muchas desigualdades (a lo sumo $2m$), se incluirán directamente en la formulación del modelo para las pruebas. En esta experimentación se descartan las desigualdades 6.8 ya que se considera que no van a servir para mejorar el rendimiento del algoritmo. En las pruebas realizadas individualmente con estas desigualdades se observaba que los valores de las variables δ^* eran muy bajos. Además, como se trata de minimizar el valor de las etiquetas difícilmente se supere la cota superior impuesta por la desigualdad. Entonces, las configuraciones de desigualdades válidas con las que se experimentó fueron las siguientes.

- CPLEX-BC1: Las desigualdades 6.7 como restricciones del modelo original.
- CPLEX-BC2: Las desigualdades 6.7 como restricciones del modelo original y la incorporación de las familias 5 y 6 en una rutina de separación.
- CPLEX-BC3: Las desigualdades 6.7 como restricciones del modelo original y la incorporación de las desigualdades 3-fence en una rutina de separación.

- CPLEX-BC4: Las desigualdades 6.7 como restricciones del modelo original y la incorporación de las desigualdades anti-ciclo en una rutina de separación.

- CPLEX-BC5: Las desigualdades 6.7 como restricciones del modelo original y la incorporación de las desigualdades 3-fence, anti-ciclo y las familias 5 y 6 en una rutina de separación.

En el análisis de los resultados de estas pruebas llevarán a la elección de una configuración para establecer el algoritmo de *Branch & Cut*. La comparación se va a realizar en base a los *gaps* obtenidos en la ejecución de las pruebas.

Como se viene haciendo a lo largo del trabajo, se van a utilizar instancias de grafos generadas aleatoriamente con distintas densidades de aristas. Sin embargo, para esta experimentación se utilizó un sobconjunto de los grafos de densidad media y alta, ya que son las instancias más difíciles de resolver.

Entonces, en las Tab. 8.1 y 8.2 se pueden observar valores promedio para los *gaps* alcanzados y cantidad de nodos explorados por las configuración mencionadas, clasificados por la densidad de las instancias.

	CPLEX-BC1		*CPLEX-BC2*		*CPLEX-BC3*	
Densidad	gap (%)	#nodos	gap (%)	#nodos	gap (%)	#nodos
40 %	0	409568.5	0	440293	0	306285.5
50 %	0	3031078.75	8.32	8682714.25	7.14	234413
60 %	8	1493102.5	9.29	1307134.5	17.96	297034
70 %	28.57	4625182.5	34.15	3643489.75	42.01	85504
80 %	37	3600815.5	42.71	1963157.75	46.87	42099.5
90 %	42.71	2489160	45.82	1038507.5	48.61	19017

Tab. 8.1: Resultados obtenidos para las configuraciones CPLEX-BC1, CPLEX-BC2 y CPLEX-BC3.

	CPLEX-BC4		*CPLEX-BC5*	
Densidad	gap (%)	#nodos	gap (%)	#nodos
40 %	0	69768.5	10.71	27059.75
50 %	4.54	1044601.25	7.14	158097.75
60 %	7.14	456833	26.39	9638
70 %	49.72	16752.75	47.99	73080.25
80 %	48.61	339455.25	48.61	79712.5
90 %	48.61	179488.5	48.61	35623.75

Tab. 8.2: Resultados obtenidos para las configuraciones CPLEX-BC4 y CPLEX-BC5.

De las Tab. 8.1 y 8.2 se puede observar que la configuración CPLEX-BC1 es la que produce mejores resultados. En casi todas las densidades obtiene mejores *gaps* que las otras configuraciones (excepto para las instancias con densidad 60 % en dónde CPLEX-BC4 consigue un *gap* menor). También se puede observar que es el algoritmo que logra recorrer más nodos del árbol. Esto se debe a que las otras configuraciones consumen tiempo

al ejecutar las rutinas de separación, los cortes encontrados no ayudan a mejorar el valor de las relajaciones y además, las relajaciones se hacen más pesadas.

La resolución en los nodos del árbol no parece ser un factor importante para consumir tanto tiempo en las configuraciones CPLEX-BC1, CPLEX-BC2 y CPLEX-BC4, ya que no se agregaban muchas restricciones al modelo. En cambio, sí puede llegar a ser determinante en CPLEX-BC3 y CPLEX-BC5. El algoritmo CPLEX-BC3 utiliza las desigualdades 3-fence y en cada nodo se introducen varias desigualdades al modelo. Esto puede hacer que la resolución en los nodos se vuelva pesada dado que el tamaño del modelo se va incrementando mucho. El algoritmo CPLEX-BC5 no sólo usa las desigualdades 3-fence sino también el resto. Por esta razón, se puede ver que son los algoritmos que menos nodos del árbol recorren en el tiempo límite.

Entonces, analizando los resultados se decidió que CPLEX-BC1 será la configuración del algoritmo final. Se va a comparar contra el algoritmo original: la implementación del modelo 1 usando el algoritmo *Branch & Cut* que viene por defecto en CPLEX. Al algoritmo original lo denominamos CPLEX-D1. La Tab. 8.3 muestra esta comparación.

	CPLEX-D1		CPLEX-BC1	
Densidad	tiempo (seg)	gap (%)	tiempo (seg)	gap (%)
Baja	129.96	1.167	0.169	0
Media	2569.19	26.90	890.19	5.476
Alta	3600	67.58	3600	39.28

Tab. 8.3: Comparación de los resultados obtenidos con CPLEX-D1 y CPLEX-BC1.

De la Tab. 8.3 se evidencia que el algoritmo CPLEX-BC1 mejora notablemente los *gaps*. De hecho, llega a resolver todos las instancias de baja densidad y casi todas las de media densidad. Ninguno de los algoritmos logra resolver alguna instacia de los grafos de alta densidad, pero CPLEX-BC1 presenta un valor promedio de *gap* mucho menor que CPLEX-D1.

También se puede notar que, a excepción de las instancias de alta densidad en dónde ambos algoritmos consumen el tiempo límite para cada instancia, al algoritmo CPLEX-BC1 consume mucho menos tiempo para resolver las instancias.

8.2. Resultados - Modelo 2

En este apartado se presentarán los resultados obtenidos para el modelo 2 utilizando las dos desigualdades identificadas. Es decir, el algoritmo *Branch & Cut* definitivo para el modelo 2 hace uso de los dos desigualdades encontradas (a diferencia del modelo 1, en dónde se probaron distintas configuraciones ya que se identificaron más cantidad de familias).

Nuevamente, el objetivo es comparar el rendimiento del algoritmo original con el algoritmo que utiliza las heurísticas y las familias de desigualdades encontradas. En este caso, CPLEX-D2 será el algoritmo original, es decir, la implementación del modelo 2 ejecutado con el algoritmo *Branch & Cut* que viene por defecto en CPLEX. Por otro lado, CPLEX-M2 será el algoritmo *Branch & Cut* que se logró conformar: el modelo 2 con las

desigualdades *etiqueta necesaria* como restricciones del modelo y utilizando las desigualdades *blossom* en una rutina de separación (además de ejecutar la HGA al principio del algoritmo y la heurística primal en los nodos del árbol). Los resultados se pueden ver en la Tab. 8.4.

	CPLEX-D2		*CPLEX-M2*	
Densidad	tiempo (seg)	gap (%)	tiempo (seg)	gap (%)
Baja	262.5	2.56	176.54	0
Media	3361.72	43.05	3302.55	25.78
Alta	3600	67.27	3600	34.58

Tab. 8.4: Comparación de los resultados obtenidos con CPLEX-D2 y CPLEX-M2.

Al igual que lo sucedido con el modelo 1, se puede observar que el algoritmo propuesto no sólo mejora notablemente los *gaps* sino también baja los tiempos de resolución. El algoritmo CPLEX-M2 logra resolver todas las instancias de baja densidad y algunas pocas de las de media densidad.

8.3. Comparación de los resultados de cada modelo

En el panorama individual se puede ver que ambos algoritmos propuestos mejoran en gran medida los respectivos algoritmos iniciales. Ambos algoritmos logran *gaps* más ajustados y mejores tiempos de resolución.

Por los resultados parece que el de mejor rendimiento es el algoritmo CPLEX-BC1 que resuelve muy rápido las instancias de baja densidad y casi todas las de media densidad en un tiempo razonable. Sin embargo, el algoritmo CPLEX-M2 se hace fuerte en instancias de densidad alta ya que, a pesar de no lograr resolver instancias en el tiempo establecido, consigue buenas cotas superiores haciendo que mejoren los *gaps*. De hecho, se puede notar que el valor promedio de *gaps* para las instancias de alta densidad de CPLEX-M2 es menor que el de CPLEX-BC1. Esto se debe justamente a que CPLEX-M2 consigue mejores cotas superiores que CPLEX-BC1.

9. CONCLUSIONES Y TRABAJO A FUTURO

El trabajo realizado permite sacar varias conclusiones acerca de la resolución del problema. Los capítulos están organizados de manera de seguir los pasos que se transitaron para llegar a los algoritmos finales para cada modelo. Primero un relevamiento de la literatura sobre el problema e interiorizar los conceptos fundamentales de programación lineal entera. Luego, elaborar las formulaciones y comparar los rendimientos con la finalidad de elegir una de ellas para estudiar en profundidad. En este caso, como la comparación no arrojó grandes diferencias, se siguió con los dos modelos a lo largo del trabajo. Por último, se hizo un estudio de los modelos para mejorar los rendimientos con distintas herramientas (por ejemplo, con heurísticas y planos de corte) y llegar a un algoritmo final competitivo para cada modelo.

Cabe destacar que CPLEX es un software poderoso para resolver este tipo de problemas. CPLEX lograba resolver muchas de las instancias chicas (lo que denominamos grafos de baja densidad). No obstante, para resolver la totalidad de las instancias chicas dentro del tiempo límite y mejorar los *gaps* de las instancias más grandes se necesitó hacer un estudio particular del problema. Esto permitió mejorar el rendimiento de los algoritmos de resolución. De todas maneras, el estudio particular del problema está respaldado por las herramientas generales que ofrece CPLEX, que son muy importantes al momento de resolver problemas de optimización.

Lo primero que se puede destacar es que se afrontó un problema relativamente nuevo, que solo había sido analizado bajo un enfoque de teoría de grafos, estudiándolo desde la programación lineal entera. En este sentido, se formularon dos modelos que resuelven el problema. En principio, el objetivo era comparar el rendimiento de los modelos para elegir uno y estudiarlo más en profundidad. Esta comparación se realizó siguiendo dos criterios. En primera instancia se consideró el valor de la relajación lineal de cada modelo. Como segunda medida, se observó el comportamiento de cada modelo al ser resuelto mediante el algoritmo *Branch & Cut* por defecto de CPLEX. Sin embargo, en los resultados obtenidos pudimos notar que la performance era similar. Por este motivo se decidió continuar con el estudio de ambos.

Se dedicó bastante trabajo a diseñar una heurística para el PCAET. Se implementó el método constructivo propuesto por en el trabajo original para generar un coloreo de aristas por un Δ-etiquetado. Sin embargo, en el trabajo se demuestra que en el etiquetado óptimo el valor de la máxima etiqueta es cercano a $\frac{\Delta}{2}$, por lo que en esta etapa se probaron diferentes algoritmos para tratar de acercarse a ese valor. Con este objetivo, se desarrollaron dos heurísticas, la HCV y la HGA, y se comparó su rendimiento para elegir una. Se pudo observar que la HGA lograba producir un etiquetado con valor de etiqueta máxima menor a Δ en todas las instancias probadas, resultando la mejor alternativa. Entonces, la HGA se usó como heurística inicial en los modelos para proveer una solución inicial y disminuir las cotas de las etiquetas.

En cuanto a las heurísticas primales de cada modelo, resultó difícil dar con un algo-

ritmo que diera buenas soluciones. Como primer paso para el desarrollo de las heurísticas se analizó la estructura de los puntos en las relajaciones de los nodos. La primera aproximación de redondear los valores de las variables al entero más cercano no sirve ya que no se genera un coloreo. Además, no resultó posible corregir los valores de las etiquetas sin superar la cota establecida. Por este motivo, se decidió incorporar a las ideas de la HGA los valores de las variables en los óptimos de las relajaciones de las heurísticas primales.

La incorporación de las heurísticas a los algoritmos logró en algunos casos mejorar en una unidad las cota superiores. Por eso, se incorporaron las heurísticas para el algoritmo final en cada modelo.

El estudio poliedral de cada modelo fue determinante para lograr las mejoras en el algoritmo. Se lograron identificar varias familias de desigualdades válidas para cada modelo que logran ajustar las relajaciones lineales originales.

Se analizó y experimentó con cada desigualdad válida individualmente, y luego juntas para evaluar cuáles podían servir para el algoritmo final y en qué grado. En el caso del modelo 1, resultó interesante ver que se destacaba una desigualdad en el análisis individual, que luego terminó siendo la única desigualdad que se agregó al modelo por los buenos resultados que produjo. Por el lado del modelo 2, también se destacó en rendimiento una desigualdad, pero ya que solamente se encontraron dos desigualdades, se resolvió usar ambas en el algoritmo.

Lo interesante de aprovechar la programación lineal entera para resolver este tipo de problemas es que tiene una gran combinación de decisiones que se pueden tomar que producen diferentes algoritmos de resolución. Resulta imposible explorar todas las combinaciones de opciones, pero deja abierto un gran campo para trabajar a futuro, como por ejemplo en los siguientes aspectos:

- **los modelos** Como se mencionó, no se encontraron estudios del PCAET bajo un enfoque de programación lineal entera. Esto quiere decir que hay muchas potenciales formulaciones que modelen el problema. En este trabajo se exploraron nada más que dos. Se consideró en algún momento una formulación que mezclara las ideas de los modelos propuestos en el trabajo. Además, nuevos modelos dan pie a nuevos análisis poliedrales, heurísticas primales, etc. Es decir, hay espacio para aumentar las combinaciones posibles.

- **las heurísticas** Este aspecto es uno de los que queda más abierto a seguir progresando. Debe haber mucho trabajo por hacer para lograr diseñar heurísticas generales que logren soluciones de calidad que estén cerca de la óptima. Es preciso decir que las heurísticas desarrolladas en el trabajo surgieron de conceptos simples. Con técnicas heurísticas más sofisticadas probablemente se puede llegar a dar con heurísticas más efectivas.

  Lo mismo se puede decir en cuanto a las heurísticas primales. Con un estudio más profundo de la estructura de los puntos de las relajaciones en los nodos se podría encontrar algún patrón o intuición que lleven a buenas heurísticas primales.

- **estudio poliedral** Con respecto a los dos modelos, se pudo observar que bastó una desigualdad válida en cada modelo para achicar en gran medida los *gaps* de la primera relajación. Por este motivo, sería importante poner foco en identificar

más familias de desigualdades que hagan más eficientes los modelos. Por supuesto, lo ideal sería encontrar *facetas* o desigualdades lo más cerca posible a la cápsula convexa del modelo.

Un aspecto que quedó pendiente es tratar de caracterizar los puntos óptimos de la relajación lienal de los modelos resultantes de agregar algunas de las desigualdades válidas. Por ejemplo, al incorporar las desigualdades 6.7 al modelo 1 el valor óptimo daba $\frac{\Delta+3}{4}$ para todas instancias. Pudimos ver que el óptimo era mayor o igual que este valor, pero sospechamos que se cumple por igualdad.

- **opciones no exploradas** Existen parámetros que no fueron estudiados en el trabajo. Por ejemplo, se hicieron pocas pruebas con respecto a estrategias de *branching* que no dieron resultados determinantes. Por eso, se decidió usar las estrategias que provee CPLEX. De la misma manera, no se experimentó con las distintas estrategias de exploración de próximo nodo, se aprovecharon las que usa CPLEX. Hay bastante trabajo y experimentación para hacer con las distintas estrategias de *branching* y exploración de nodos.

10. APÉNDICE

En los capítulos 6 y 7 se presentaron familias de desigualdades válidas para cada modelo. En este apéndice se propone mostrar que efectivamente son cortes. Esto quiere decir que cada desigualdad válida no esté implicada por las restricciones lineales del modelo (ni que tampoco esté implicada por las restricciones del modelo junto con las otras desigualdades válidas). Notar que el hecho de que sea corte no significa que sea necesariamente útil para el problema en el algoritmo de *Branch & Cut*, ya que puede estar cortando un punto que no esté cerca del óptimo entero.

Un procedimiento para mostrar que las desigualdades válidas son planos de corte es el que se describe a continuación. En primer lugar, se cuenta con las restricciones del modelo $Ax \leq b$ del problema y $\pi_1 x \leq s_1$, $\pi_2 x \leq s_2$, ..., $\pi_r x \leq s_r$ las familias de desigualdades que se identificaron para el problema. Si se quiere mostrar que la familia $\pi_i x \leq s_i$ es corte, el procedimiento consiste en tomar una instancia particular del problema y resolver la siguiente formulación para dicha instancia:

$$\begin{aligned} \text{Maximizar} \quad & \pi_i x \\ \text{sujeto a} \quad & Ax \leq b \\ & \pi_1 x \leq s_1 \\ & \vdots \\ & \pi_{i-1} x \leq s_{i-1} \\ & \pi_{i+1} x \leq s_{i+1} \\ & \vdots \\ & \pi_r x \leq s_r \\ & x \in \mathbb{R}^n_{\geq 0} \end{aligned}$$

De esta manera, si se obtiene que la relajación lineal de la formulación es mayor que s_i significa que la desigualdad válida es un corte. De lo contrario, si se obtiene que el óptimo de la relajación es menor o igual que s_i, se debe probar con otras desigualdades de la misma familia para la instancia. Si se sigue manteniendo el mismo resultado, habría que pensar si la familia de desigualdades es implicada por el modelo o la instancia elegida no es la adecuada.

En el caso que la familia de desigualdades sean del estilo $\pi_i x \geq s_i$, el procedimiento es análogo excepto que en la formulación se debe minimizar la función objetivo, y luego verificar si la relajación es menor que s_i.

A continuación, se muestran, para cada modelo, la instancia del problema elegida para las pruebas, las formulaciones resultantes para cada familia y los resultados obtenidos.

10.1. Desigualdades válidas - Modelo 1

Para estas pruebas no se van a tener en cuenta las desigualdades de las familias 1, 2 y 3, en su lugar se utilizarán las desigualdades de la familia 4 ya que se demostró que son más fuertes. Asimismo, tampoco se se usarán las desigualdades traídas del *problema de ordenamiento lineal*.

La instancia de grafo $G = (V, E)$ que se eligió para probar en este modelo es el que muestra la Fig. 7.1, en donde $\Delta = 5$

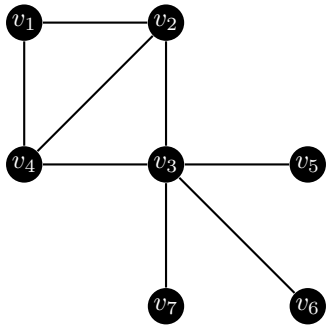

Fig. 10.1: Grafo elegido para las pruebas del modelo 1.

Cabe destacar que para las formulaciones resultantes, la variable z que se usaba en la minimización ya no es necesaria considerarla y tampoco las restricciones que la involucran.

Familia 4

- desigualdad por cota inferior

$$x_{vw_i} + x_{w_i} \geq 2 + \sum_{j=1}^{i-1} \delta_{w_j v w_i} + \sum_{j=i+1}^{k} (1 - \delta_{w_i v w_j})$$

El problema que se resolvió es el siguiente:

$$
\begin{aligned}
\text{minimizar} \quad & x_2 + x_{2,3} + \delta_{2,3,4} + \delta_{2,3,5} + \delta_{2,3,6} + \delta_{2,3,7} \\
\text{sujeto a} \quad & \\
& x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_{vw_i} + x_{w_i} \leq 2\Delta - \sum_{j=1}^{i-1}(1 - \delta_{w_j v w_i}) - \sum_{j=i+1}^{k} \delta_{w_i v w_j} && \forall v \in V(G),\ i = 1, \ldots, k \\
& x_{uv} + x_{vr} + x_{ru} + 2x_u + 2x_v + 2x_r \geq 12 && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& \delta_{uvr} + \delta_{vru} \leq 1 + \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& \delta_{uvr} + \delta_{vru} \geq \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& 1 \leq x_w, x_{uv} \leq \Delta && \forall w \in V(G) \text{ y } (u,v) \in E(G) \\
& 0 \leq \delta_{urv} \leq 1 && \forall (u,r),(r,v) \in E(G) \text{ y } u \neq v
\end{aligned}
$$

La desigualdad elegida como función objetivo de este conjunto indica que debería ser mayor o igual que 6. Corresponde a obtener una cota inferior para la arista (v_2, v_3). Al realizar la prueba, se obtuvo una relajación lineal con valor $2{,}44 < 6$, por lo que se concluye que este conjunto de desigualdades no esta implicado por otras.

- desigualdad por cota superior

$$x_{vw_i} + x_{w_i} \leq 2\Delta - \sum_{j=1}^{i-1}(1 - \delta_{w_j v w_i}) - \sum_{j=i+1}^{k} \delta_{w_i v w_j}$$

Se resolvió el siguiente problema:

$$
\begin{aligned}
\text{maximizar} \quad & x_2 + x_{2,3} + \delta_{2,3,4} + \delta_{2,3,5} + \delta_{2,3,6} + \delta_{2,3,7} \\
\text{sujeto a} \quad & \\
& x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_{vw_i} + x_{w_i} \geq 2 + \sum_{j=1}^{i-1} \delta_{w_j v w_i} + \sum_{j=i+1}^{k} (1 - \delta_{w_i v w_j}) && \forall v \in V(G),\ i = 1, \ldots, k \\
& x_{uv} + x_{vr} + x_{ru} + 2x_u + 2x_v + 2x_r \geq 12 && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& \delta_{uvr} + \delta_{vru} \leq 1 + \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& \delta_{uvr} + \delta_{vru} \geq \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& 1 \leq x_w, x_{uv} \leq \Delta && \forall w \in V(G) \text{ y } (u,v) \in E(G) \\
& 0 \leq \delta_{urv} \leq 1 && \forall (u,r),(r,v) \in E(G) \text{ y } u \neq v
\end{aligned}
$$

La desigualdad elegida como función objetivo de este conjunto indica que debería ser menor o igual que 10. Es la misma desigualdad que la usada para la formulación anterior, pero esta vez mirando la cota superior de la arista. Para la formulación, se obtuvo una relajación lineal con valor 13,56 > 10. Por lo tanto, estas desigualdades no están implicadas por el resto.

Familia 5

$$x_{uv} + x_{vr} + x_{ru} + 2x_u + 2x_v + 2x_r \geq 12$$

El problema que se resolvió es la siguiente:

$$
\begin{aligned}
\text{minimizar} \quad & x_{1,2} + x_{1,4} + x_{2,4} + 2x_1 + 2x_2 + 2x_4 \\
\text{sujeto a} \quad & \\
& x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_{vw_i} + x_{w_i} \geq 2 + \sum_{j=1}^{i-1} \delta_{w_j v w_i} + \sum_{j=i+1}^{k} (1 - \delta_{w_i v w_j}) && \forall v \in V(G),\ i = 1, \ldots, k \\
& x_{vw_i} + x_{w_i} \leq 2\Delta - \sum_{j=1}^{i-1} (1 - \delta_{w_j v w_i}) - \sum_{j=i+1}^{k} \delta_{w_i v w_j} && \forall v \in V(G),\ i = 1, \ldots, k \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& \delta_{uvr} + \delta_{vru} \leq 1 + \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& \delta_{uvr} + \delta_{vru} \geq \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& 1 \leq x_w, x_{uv} \leq \Delta && \forall w \in V(G) \text{ y } (u,v) \in E(G) \\
& 0 \leq \delta_{urv} \leq 1 && \forall (u,r),(r,v) \in E(G) \text{ y } u \neq v
\end{aligned}
$$

Para definir la función objetivo se utilizó el ciclo de longitud 3 formado por los vértices v_1, v_2 y v_4. Se obtuvo una relajación lineal con valor $10{,}5 < 12$, por lo que esta familia de desigualdades no está implicada por las otras familias.

Familia 6

- Transitividad por menor

$$\delta_{uvr} + \delta_{vru} \leq 1 + \delta_{vur}$$

Se resolvió el siguiente problema:

$$
\begin{aligned}
\text{maximizar} \quad & \delta_{2,1,4} - \delta_{1,2,4} + \delta_{1,4,2} \\
\text{sujeto a} \quad & \\
& x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_{vw_i} + x_{w_i} \geq 2 + \sum_{j=1}^{i-1} \delta_{w_j v w_i} + \sum_{j=i+1}^{k} (1 - \delta_{w_i v w_j}) && \forall v \in V(G),\ i = 1, \ldots, k \\
& x_{vw_i} + x_{w_i} \leq 2\Delta - \sum_{j=1}^{i-1} (1 - \delta_{w_j v w_i}) - \sum_{j=i+1}^{k} \delta_{w_i v w_j} && \forall v \in V(G),\ i = 1, \ldots, k \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& x_{uv} + x_{vr} + x_{ru} + 2x_u + 2x_v + 2x_r \geq 12 && \forall (u,v),(v,r),(r,u) \in E(G) \\
& \delta_{uvr} + \delta_{vru} \geq \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& 1 \leq x_w, x_{uv} \leq \Delta && \forall w \in V(G) \text{ y } (u,v) \in E(G) \\
& 0 \leq \delta_{urv} \leq 1 && \forall (u,r),(r,v) \in E(G) \text{ y } u \neq v
\end{aligned}
$$

La función objetivo se formuló a partir del ciclo de longitud 3 formado por los vértices v_1, v_2 y v_4. Se obtuvo una relajación lineal con valor 1,67 > 1, por lo que este conjunto de desigualdades no está implicado por el resto.

- Transitividad por mayor

$$\delta_{uvr} + \delta_{vru} \geq \delta_{vur}$$

Se resolvió el siguiente problema:

$$
\begin{aligned}
\text{maximizar} \quad & -\delta_{2,1,4} + \delta_{1,2,4} - \delta_{1,4,2} \\
\text{sujeto a} \quad & \\
& x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
& x_{vw_i} + x_{w_i} \geq 2 + \sum_{j=1}^{i-1} \delta_{w_j v w_i} + \sum_{j=i+1}^{k} (1 - \delta_{w_i v w_j}) && \forall v \in V(G),\ i = 1, \ldots, k \\
& x_{vw_i} + x_{w_i} \leq 2\Delta - \sum_{j=1}^{i-1} (1 - \delta_{w_j v w_i}) - \sum_{j=i+1}^{k} \delta_{w_i v w_j} && \forall v \in V(G),\ i = 1, \ldots, k \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& x_{uv} + x_{vr} + x_{ru} + 2x_u + 2x_v + 2x_r \geq 12 && \forall (u,v),(v,r),(r,u) \in E(G) \\
& \delta_{uvr} + \delta_{vru} \leq 1 + \delta_{vur} && \forall (u,v),(v,r),(r,u) \in E(G) \\
& && \text{y } u \neq v, v \neq r, r \neq u \\
& 1 \leq x_w, x_{uv} \leq \Delta && \forall w \in V(G) \text{ y } (u,v) \in E(G) \\
& 0 \leq \delta_{urv} \leq 1 && \forall (u,r),(r,v) \in E(G) \text{ y } u \neq v
\end{aligned}
$$

Al igual que en la formulación anterior, se eligió el ciclo formado por v_1, v_2 y v_4. En este caso, se obtuvo una relajación lineal con valor $0{,}67 > 0$. Esto quiere decir que el conjunto de desigualdades no están implicadas por el resto.

La familia 3 y la familia 4 no son equivalentes

En la sección de desigualdades válidas del modelo 1 se mostró que las desigualdades definidas para la familia 4 implicaban a las desigualdades de la familia 3. Sin embargo, en este apartado vamos a ver que no son equivalentes. La metodología es la misma: se incluirán al modelo original las desigualdades de la familia 3, se elige alguna instancia de las desigualdades de la familia 4 como función objetivo y se calcula la relajación del modelo resultante.

- Veamos que las desigualdades 6.5 no implican a las desigualdades 6.7. El problema que se resolvió es el siguiente:

$$\begin{aligned}
&\text{minimizar} && x_2 + x_{2,3} + \delta_{2,3,4} + \delta_{2,3,5} + \delta_{2,3,6} + \delta_{2,3,7} \\
&\text{sujeto a} \\
&&& x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
&&& x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
&&& \sum_{i=1}^{k} x_{vw_i} + \sum_{i=1}^{k} x_{w_i} \geq \frac{k^2 + 3k}{2} && \forall v \in V(G) \\
&&& 1 \leq x_w, x_{uv} \leq \Delta && \forall w \in V(G) \text{ y } (u,v) \in E(G) \\
&&& 0 \leq \delta_{urv} \leq 1 && \forall (u,r),(r,v) \in E(G) \text{ y } u \neq v
\end{aligned}$$

  La desigualdad elegida como función objetivo de este conjunto indica que debería ser mayor o igual que 6. Sirve para acotar inferiormente a la arista (v_2, v_3). Se obtuvo una relajación lineal con valor $3{,}56 < 6$, esto quiere decir que las desigualdades 6.7 aportan más información que las desigualdades 6.5.

- De igual manera, veamos que las desigualdades 6.6 no implican a las desigualdades 6.8. Se resolvió el siguiente problema para ello:

$$\begin{aligned}
&\text{maximizar} && x_2 + x_{2,3} + \delta_{2,3,4} + \delta_{2,3,5} + \delta_{2,3,6} + \delta_{2,3,7} \\
&\text{sujeto a} \\
&&& x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
&&& x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall (u,j),(j,v) \in E(G) \text{ y } u \neq v \\
&&& \sum_{i=1}^{k} x_{vw_i} + \sum_{i=1}^{k} x_{w_i} \leq k(2\Delta + 1) - \frac{k(k+1)}{2} && \forall v \in V(G) \\
&&& 1 \leq x_w, x_{uv} \leq \Delta && \forall w \in V(G) \text{ y } (u,v) \in E(G) \\
&&& 0 \leq \delta_{urv} \leq 1 && \forall (u,r),(r,v) \in E(G) \text{ y } u \neq v
\end{aligned}$$

  La desigualdad elegida como función objetivo de este conjunto indica que debería ser menor o igual que 10. La función objetivo es la misma que la formulación anterior, pero en este caso viendo la cota superior de la arista. Se obtuvo una relajación lineal con valor $12{,}44 > 10$, de lo que se concluye que las desigualdades 6.8 no están implicadas por las desigualdades 6.6.

10.2. Desigualdades válidas - Modelo 2

La instancia de grafo $G = (V, E)$ elegida para probar en el modelo 2 se muestra en la Fig. 7.2, donde $\Delta = 3$

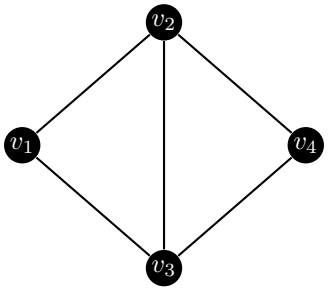

Fig. 10.2: Grafo elegido para las pruebas del modelo 2.

Para este modelo se identificaron solo dos familias de desigualdades válidas:

Desigualdades blossom

$$\sum_{(u,v)\in E(G[U])} z_{uvk} \leq \left\lfloor \frac{|U|}{2} \right\rfloor$$

Se resolvió el problema:

$$
\begin{aligned}
\text{maximizar} \quad & z_{2,3,7} + z_{2,4,7} + z_{3,4,7} \\
\text{sujeto a} \quad & \\
& \sum_{j=1}^{\Delta} x_{vj} = 1 && \forall v \in V(G) \\
& \sum_{j=1}^{\Delta} w_{uvj} = 1 && \forall (u,v) \in E(G) \\
& \sum_{k=3}^{3\Delta} z_{uvk} = 1 && \forall (u,v) \in E(G) \\
& z_{uvk} \geq x_{uj} + w_{uvr} + x_{vp} - 2 && \forall (u,v) \in E(G) \text{ y } j,r,p \in \{1,\ldots,\Delta\} \\
& && \text{tal que } k = j + r + p \\
& \sum_{\forall v \in N(v)} z_{uvk} \leq 1 && \forall u \in V(G), d(u) > 1 \text{ y } k \in \{3,\ldots,3\Delta\} \\
& y \geq \sum_{j=1}^{\Delta} j\, x_{vj} && \forall v \in V(G) \\
& y \geq \sum_{j=1}^{\Delta} j\, w_{uvj} && \forall (u,v) \in E(G) \\
& y \geq \sum_{k=3}^{3\Delta} \lceil k/3 \rceil\, z_{uvk} && \forall (u,v) \in E(G) \\
& x_{vj} \in \{0,1\} && \forall v \in V(G) \text{ y } j \in \{1,\ldots,\Delta\} \\
& w_{uvj}, z_{uvk} \in \{0,1\} && \forall (u,v) \in E(G) \text{ y } k \in \{3,\ldots,3\Delta\}, \\
& && j \in \{1,\ldots,\Delta\} \\
& 1 \leq y \leq \Delta, \text{ con } y \in \mathbb{Z}
\end{aligned}
$$

Para definir la función objetivo se tomó el triángulo derecho del grafo (conformado por los vértices v_2, v_3 y v_4). Además, se fijó en el valor 7 el color de las aristas. Entonces el tamaño del conjunto U es 3 y la cuenta del lado derecho de la desigualdad sería 1. En este caso, se obtuvo un valor de relajación lineal de 1,5 $>$ 1, por lo que la familia de desigualdades no está implicada pro el resto.

Desigualdades etiqueta necesaria

$$\sum_{k=3}^{3\Delta} \lceil k/3 \rceil\, z_{uvk} \leq y$$

El problema es la siguiente:

$$
\begin{aligned}
\text{maximizar} \quad & z_{1,3,3} + 2z_{1,3,4} + 2z_{1,3,5} + 2z_{1,3,6} + \\
& 3z_{1,3,7} + 3z_{1,3,8} + 3z_{1,3,9} - y \\
\text{sujeto a} \quad & \\
& \sum_{j=1}^{\Delta} x_{vj} = 1 && \forall v \in V(G) \\
& \sum_{j=1}^{\Delta} w_{uvj} = 1 && \forall (u,v) \in E(G) \\
& \sum_{k=3}^{3\Delta} z_{uvk} = 1 && \forall (u,v) \in E(G) \\
& z_{uvk} \geq x_{uj} + w_{uvr} + x_{vp} - 2 && \forall (u,v) \in E(G) \text{ y } j,r,p \in \{1,\ldots,\Delta\} \\
& && \text{tal que } k = j + r + p \\
& \sum_{\forall v \in N(v)} z_{uvk} \leq 1 && \forall u \in V(G), d(u) > 1 \text{ y } k \in \{3,\ldots,3\Delta\} \\
& y \geq \sum_{j=1}^{\Delta} j\, x_{vj} && \forall v \in V(G) \\
& y \geq \sum_{j=1}^{\Delta} j\, w_{uvj} && \forall (u,v) \in E(G) \\
& \sum_{(u,v) \in E(G[U])} z_{uvk} \leq \left\lfloor \frac{|U|}{2} \right\rfloor && \forall k \in \{3,\ldots,3\Delta\},\ U \subseteq V(G) \\
& x_{vj} \in \{0,1\} && \forall v \in V(G) \text{ y } j \in \{1,\ldots,\Delta\} \\
& w_{uvj}, z_{uvk} \in \{0,1\} && \forall (u,v) \in E(G) \text{ y } k \in \{3,\ldots,3\Delta\}, \\
& && j \in \{1,\ldots,\Delta\} \\
& 1 \leq y \leq \Delta, \text{ con } y \in \mathbb{Z}
\end{aligned}
$$

En este caso se decidió utilizar la arista $(1,3)$ para la expresión de la función objetivo. Se obtuvo un valor de relajación lineal de $1{,}67 > 0$, lo que significa que esta familia de desigualdades tampoco está implicada por el resto.

BIBLIOGRAFÍA

[1] Stephan Brandt, Kristína Budajová, Dieter Rautenbach, and Michael Stiebitz. Edge colouring by total labellings. *Discrete Mathematics*, 310(2):199 – 205, 2010.

[2] Martin Bača, Mirka Miller, Joseph Ryan, et al. On irregular total labellings. *Discrete Mathematics*, 307(11):1378–1388, 2007.

[3] Riadh Khennoufa, Hamida Seba, and Hamamache Kheddouci. Edge coloring total k-labeling of generalized petersen graphs. *Information Processing Letters*, 113(13):489–494, 2013.

[4] Hamida Seba and Riadh Khennoufa. Edge coloring by total labelings of 4-regular circulant graphs. *Electronic Notes in Discrete Mathematics*, 41:141–148, 2013.

[5] Laurence A Wolsey. *Integer programming*. Wiley New York, 1998.

[6] George Bernard Dantzig. Linear programming and extensions. *RAND Corporation*, 1963.

[7] CPLEX. http://www.ilog.com/products/cplex.

[8] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[9] Jon Lee and Janny Leung. A comparison of two edge-coloring formulations. *Operations research letters*, 13(4):215–223, 1993.

[10] Jayadev Misra and David Gries. A constructive proof of vizing's theorem. *Information Processing Letters*, 41(3):131–133, 1992.

[11] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations research*, 32(6):1195–1220, 1984.

[12] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33(1):43–60, 1985.