DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Coloreo Particionado de Grafos

Investigación operativa
Segundo Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Confalonieri, Gisela Belén	511/11	gise_5291@yahoo.com.ar
Mignanelli, Alejandro Rubén	609/11	minga_titere@hotmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

En el presente trabajo se realizará un estudio comparativo entre las estrategias *Branch and Bound* y *Cut and Branch* sobre el problema del coloreo particionado de grafos (que se explicará en otra sección de este informe) al encararlo desde Programación Lineal Entera.

Los objetivos del trabajo son los siguientes:

- **Interacción con CPLEX:**

  CPLEX es un paquete de software comercial y académico para resolución de problemas de Programación Lineal y Programación Lineal Entera. Uno de los fines de nuestro trabajo consiste entonces en aprender a usar este paquete tanto resolviendo problemas, como reemplazando parte del trabajo que éste realiza con código propio.

- **Modelado de un PLEM:**

  El modelado de un PLEM (Problema de Programación Lineal Entero Mixto) cumple un rol muy importante. Dado que PLEM es NP-difícil, es importante ajustar lo mejor posible el conjunto de soluciones factibles, de manera de tener menos "opciones" que recorrer. Por ejemplo, en el caso de coloreo de grafos existe el problema de que tiene muchas soluciones simétricas para un mismo grafo, que al fin y al cabo representan una misma solución. Por ello, parte del objetivo de este trabajo es no sólo realizar un PLEM correcto, sino elaborar estrategias que permitan ver la menor cantidad de soluciones posibles.

- **Desigualdades válidas e implementación de planos de corte:**

  Otro de los propósitos de este trabajo es lograr un mayor entendimiento de qué son y cómo funcionan los planos de corte. Si bien en la materia los estudiamos a nivel teórico, es importante toparnos con ellos a nivel práctico e intentar implementarlos. Para esto, utilizaremos las desigualdades válidas *clique* y *odd-hole*.

- **Comparación entre distintos métodos de resolución:**

  La última meta será realizar una comparación entre algunos métodos de resolución general:

  - *Branch-and-Bound:*

    Esta técnica subdivide el problema sucesivamente en otros más pequeños, eliminando ciertas soluciones fraccionarias, y manteniendo durante el recorrido del árbol generado una cota superior y otra inferior para el óptimo buscado. Es la resolución automática de CPLEX, por lo que le quitaremos todo el preprocesamiento y cortes que este paquete le añade, con el fin de obtener un resultado basado solamente en Branch-and-Bound.

  - *Cut-and-Branch:*

    La idea es añadir cortes en el nodo raíz, para luego realizar un Branch-and-Bound clásico, que en teoría debería tomar menos tiempo dado que la formulación del PLEM es más restrictiva. Sin embargo, procesar un problema con demasiadas restricciones también puede empeorar la performance, por lo cual un subobjetivo en esta etapa será encontrar una cantidad apropiada de cortes a agregar, de manera que favorezca la resolución.

2. Problema del Coloreo Particionado de Grafos

El problema de coloreo de grafos ha sido ampliamente estudiado y aparece en numerosas aplicaciones de la vida real, como por ejemplo en problemas de scheduling, asignación de frecuencias, secuenciamiento, etc. Formalmente, el problema puede ser definido de la siguiente forma:

Dado un grafo $G = (V, E)$ con $n = |V|$ vértices y $m = |E|$ aristas, un coloreo de G consiste en una asignación de colores o etiquetas a cada vértice $p \in V$ de forma tal que todo par de vértices $(p, q) \in E$ poseen colores distintos. El problema de coloreo de grafos consiste en encontrar un coloreo que utilice la menor cantidad posible de colores distintos.

A partir de diferentes aplicaciones, surgieron variaciones o generalizaciones de este problema, como el problema de coloreo particionado de grafos. En este problema el conjunto V se encuentran dividido en una partición $V_1, ..., V_k$, y el objetivo es asignar un color a sólo un vértice de cada partición, de manera tal que dos vértices adyacentes no reciban el mismo color, minimizando la cantidad de colores utilizados.

3. Modelado del problema

Sea un grafo $G = (V, E)$, con $n = |V|$. Definimos las siguientes variables binarias:

- Para $i = 1, ..., n, j = 1, ..., n$

$$x_{ij} = \begin{cases} 1 & \text{si el nodo } i \text{ es pintado con el color } j \\ 0 & \text{caso contrario} \end{cases}$$

- Para $j = 1, .., n$

$$w_j = \begin{cases} 1 & \text{si algun nodo es pintado con el color } j \\ 0 & \text{caso contrario} \end{cases}$$

Sea C el conjunto de colores posibles de utilizar. Buscamos minimizar la cantidad de colores distintos usados:

$$min \sum_{j \in C} w_j$$

Sujeto a:

- La variable w_j es verdadera sii algún vértice usa el color j:

$$x_{ij} \leq w_j \quad \forall j \in C, \forall i \in V$$

- Dos vecinos no pueden usar el mismo color:

$$x_{ij} + x_{kj} \leq 1 \quad \forall j \in C, \forall (i, k) \in E$$

- Cada conjunto p de la partición P tiene exactamente un color asignado:

$$\sum_{x_i \in p} \sum_{j \in C} x_{ij} = 1 \quad \forall i \in V, p \in P$$

- No se permite usar un color hasta que no se hayan usado todos los anteriores:

$$w_j \geq w_{j+1} \quad \forall 1 \leq j < |C|$$

- Ninguna partición puede estar coloreada con un color de etiqueta mayor a su índice:

$$x_{ij} = 0 \quad \forall j > p(i) + 1$$

4. Desigualdades válidas

A continuación presentaremos dos familias desigualdades y demostraremos que son válidas para el problema de coloreo particionado.

Desigualdad Clique: Sea $j_0 \in \{1, ..., n\}$ y sea K una clique maximal de G. La desigualdad clique está definida por:

$$\sum_{p \in K} x_{pj_0} \leq w_{j_0} \tag{1}$$

Demostración:

Queremos ver que es desigualdad válida. Es decir, que toda solución factible del PLEM planteado cumple (1).

Supongamos que no, o sea que $\exists s$ solución factible tal que:

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

- Si $w_{j_0} = 0$, no cumple con la restricción $x_{ij} \leq w_j$: *ABSURDO*.
- Si $w_{j_0} = 1$ entonces

$$\sum_{p \in K} x_{pj_0} > 1$$

O sea que hay al menos dos nodos en K con color j_0, lo que significa que $\exists i, h \in K$ tal que $x_{ij_0} = 1$ y $x_{hj_0} = 1$. Pero como K es una clique, $(i, h) \in E$, así que no se cumple la restricción $x_{ij} + x_{kj} \leq 1 \quad \forall j \in C, \forall (i, k) \in E$: *ABSURDO*.

Estos absurdos provienen de suponer que existe s solución factible que no cumple con la desigualdad de clique. Por lo tanto, la desigualdad resulta válida para toda solución factible del PLEM.

Desigualdad Odd-hole: Sea $j_0 \in \{1, ..., n\}$ y sea $C_{2k+1} = v_1, ..., v_{2k+1}, k \geq 2$, un agujero de longitud impar. La desigualdad odd-hole está definida por:

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

Demostración:

Queremos ver que es desigualdad válida. Es decir, que toda solución factible del PLEM planteado cumple (2).

Supongamos que no, o sea que $\exists s$ solución factible tal que:

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

- Si $w_{j_0} = 0$, no cumple con la restricción $x_{ij} - w_j \leq 0$: *ABSURDO*.
- Si $w_{j_0} = 1$ entonces

$$\sum_{p \in C_{2k+1}} x_{pj_0} > k$$

O sea que hay al menos $k + 1$ nodos en C_{2k+1} con color j_0. Por el **Lema 1**, esto significa que $\exists i, h \in C_{2k+1}$ tal que $x_{ij_0} = 1$ y $x_{hj_0} = 1$ con $(i, h) \in E$. Pero esto no cumple la restricción $x_{ij} + x_{kj} \leq 1 \quad \forall j \in C, \forall (i, k) \in E$: *ABSURDO*.

Estos absurdos provienen de suponer que existe s solución factible que no cumple con la desigualdad de agujero impar. Por lo tanto, la desigualdad resulta válida para toda solución factible del PLEM.

Lema 1: Sea C_{2k+1} circuito impar $(k \geq 2)$ y H subconjunto de nodos de C_{2k+1} tal que $|H| \geq k+1$. Entonces $\exists v, w \in H$ tal que $(v, w) \in E(C_{2k+1})$.

Demostración:

Supongamos que no. O sea que existe un H subconjunto de nodos de C_{2k+1} tal que $|H| \geq k+1$ y tal que no existe $v, w \in H$ tal que $(v, w) \in E(C_{2k+1})$. Esto significa que el grafo inducido por H es un grafo sin aristas.

Ahora, a partir del subgrafo inducido por H, intentemos construir un circuito sin conectar a los elementos de H entre sí y añadiendo la mínima cantidad de nodos extra. Para esto, basta con tomar alternadamente un elemento de H y un elemento no perteneciente a H. Para obtener un circuito, entonces, necesitamos al menos la misma cantidad de nodos extra como elementos existentes en H, y como H tiene al menos $k+1$ elementos necesitaremos al menos $k+1$ vértices más. Esto quiere decir que, para que el subgrafo inducido por H no contenga aristas, el circuito al que pertenece debe tener longitud al menos $2k+2$. ***ABSURDO***, pues el circuito al que pertenece H tiene exactamente $2k+1$ elementos.

El absurdo provino de suponer que el subgrafo inducido por H no contenía aristas, por lo cual probamos que si $|H| \geq k+1$ entonces $\exists v, w \in H$ tal que $(v, w) \in E(C_{2k+1})$.

5. Algoritmos de separación

En esta sección explicaremos las heurísticas de separación implementadas para cada familia de desigualdades válidas. En ambos casos se utilizó una estrategia *Greedy* para generar los cortes en función de la solución obtenida en la relajación lineal del problema, tratando de encontrar cliques y agujeros impares que utilicen los nodos con mayor valor, ya que intuitivamente es más probable que la desigualdad hallada separe efectivamente a la solución de la relajación.

5.1. Cliques

Para cada color disponible con valor mayor a 0[1] en la solución de la relajación (o sea, para cada color "usado"), empezando desde el de menor etiqueta hasta el de mayor etiqueta, se ordenan los nodos de mayor a menor según el valor que tienen en la solución de la relajación para el color fijado.

Entonces, se toma el nodo con mayor valor como nodo "inicial" y se busca el siguiente de mayor valor con el cual tenga una arista; luego se busca el siguiente nodo que se relacione con ambos, y así sucesivamente hasta recorrer todos los nodos. El resultado es una clique maximal del grafo original, ya que si al revisar un nodo, éste no se conectaba con todos los incluidos en la clique hasta ese momento, el mismo era descartado; es decir, todo nodo que no pertenezca a la clique hallada no tiene arista con al menos uno de los elementos de la misma, así que la clique es maximal.

Durante el armado de la clique se va acumulando la suma de los valores de los nodos en la solución de la relajación. Si en algún momento del proceso se llega a considerar agregar a la clique un nodo con valor 0, esto significa que a partir de ese momento todos los nodos a considerar tendrán el mismo valor (porque estamos consultando los nodos de manera golosa). Por lo tanto, a partir de ese momento el valor de la suma no se verá afectada, así que para evitar procesamiento innecesario se chequea que la suma obtenida hasta el momento supere el valor que tiene la variable w_j para el j que se fijó, de manera que la clique que se está armando efectivamente separe la solución hallada en la relajación. De ser así, se continúa hasta hallar la clique maximal, si no, la clique se descarta y se prosigue con la siguiente.

Una vez hallada una clique maximal, se busca como nodo "inicial" de la siguente al nodo de mayor valor que no haya sido utilizado en una clique anterior ni considerado como nodo "inicial" de una clique descartada. A partir de ahí, se vuelven a recorrer los nodos de manera golosa armando la clique bajo el mismo criterio que antes. Este procedimiento se repite hasta que se encuentra la cantidad de cliques solicitada o hasta que todos los nodos fueron, o bien utilizados en una clique válida anterior, o bien considerados como nodo "inicial" de alguna clique, haya resultado válida o no.

Observación: Cuando esta heurística es utilizada en sucesivas iteraciones de un algoritmo de planos de corte, no es posible que en distintas iteraciones se encuentren cliques "repetidas", ya que si en alguna iteración una clique es considerada como corte, la solución de cualquier relajación lineal posterior al agregado de esa restricción ya la estará cumpliendo y, por lo tanto, no podrá ser propuesta como corte nuevamente. A su vez, por este motivo no se encuentran cliques de tamaño 1 (K_1) que separen, puesto que es lo mismo que la restricción $x_{ij} \leq w_j$ del PLEM y la solución de la relajación ya lo cumple.

5.2. Odd-holes

Al igual que con las cliques, para cada color disponible "usado" en la solución de la relajación, empezando desde el de menor etiqueta hasta el de mayor etiqueta, se ordenan los nodos de mayor a menor según el valor que tienen en la solución de la relajación para el color fijado.

Entonces, se toma el nodo con mayor valor como nodo "inicial" y se busca el siguiente de mayor valor con el cual tenga una arista; luego se busca el siguiente nodo que se relacione sólo con el último de los nodos elegidos hasta el momento, y así sucesivamente hasta que se recorren todos los nodos ó se encuentra un nodo que también cierre el circuito con el nodo "inicial". Si se encuentra un nodo que cierra el circuito, se chequea que el agujero hallado tenga longitud impar mayor ó igual a 5, y de no ser así se descarta el último nodo y se continúa en busca de otro agujero que contenga a los anteriores

[1] Dado que los valores de las variables en este momento son continuas entre 0 y 1, y dada la aritmética finita de la computadora, las comparaciones se hacen con una tolerancia de 10^{-5}.

seleccionados. En caso de que no se logre hallar dicho agujero, se descarta esta selección de nodos y se busca otro.

Durante el armado del agujero se va acumulando la suma de los valores de los nodos en la solución de la relajación. Si en algún momento del proceso se llega a considerar agregar al agujero un nodo con valor 0, se chequea que la suma obtenida hasta el momento supere el valor que tiene la variable w_j para el j que se fijó, de manera que el agujero que se está armando efectivamente separe la solución hallada en la relajación. De ser así, se continúa hasta hallar el agujero completo, si no, la selección se descarta y se prosigue con la siguiente.

Una vez hallado un agujero impar de longitud apropiada, se busca como nodo "inicial" del siguente al nodo de mayor valor que no haya sido utilizado en un agujero anterior ni considerado como nodo "inicial" de una selección descartada. A partir de ahí, se vuelven a recorrer los nodos de manera golosa armando el agujero bajo el mismo criterio que antes. Este procedimiento se repite hasta que se encuentra la cantidad de agujeros solicitada o hasta que todos los nodos fueron, o bien utilizados en un agujero válido anterior, o bien considerados como nodo "inicial" de algún agujero, haya resultado válido o no.

Observación: Para asegurarnos de que la búsqueda de agujeros tiene sentido, antes de que se comience a ejecutar el algoritmo de planos de corte se hace una "poda" del grafo de la siguiente manera: se quitan los nodos de grado 1, se actualiza el grado de los nodos restantes y se repite hasta que no haya más hojas. Esto quita las "ramas colgantes" del grafo que no llevarían a ningún agujero (en particular, era un árbol). Si el grafo resultante de esta poda es vacío, significa que no existían agueros en el grafo original, así que en ese caso el algoritmo de planos de corte no utiliza la heurística de agujero impar. Si bien en caso de que el grafo podado no sea vacío no significa que contenga agujeros, al utilizar esta heurística sobre el grafo podado se reduce la cantidad de opciones a chequear.

6. Experimentación

A continuación presentaremos las pruebas realizadas, las instancias utilizadas y los resultados obtenidos.

La experimentación consta de dos partes: una enfocada en evaluar las familias de desigualdades válidas utilizadas como cortes y una enfocada en comparar la resolución mediante Branch and Bound contra la resolución mediante Cut and Branch.

Todas las corridas se realizaron sobre una CPU Intel Core i3 de 4 núcleos a 2,20 GHZ con 4GB RAM, utilizando la versión 12.6 de CPLEX.

6.1. Instancias de prueba

La idea original del trabajo era incluir en las corridas pertinentes algunas instancias DIMACS, pero dado el tamaño de las mismas se decidió trabajar sólo sobre instancias propias con tamaños manejables por los algoritmos utilizados, con el objetivo de poder visualizar en un tiempo razonable las diferencias entre las diferentes opciones.

Hemos generado un set de instancias aleatorias con distintas cantidades de nodos, densidades de aristas y tamaño de partición. En el Cuadro 1 se muestra la información de las mismas.

Instancia	Cant Nodos	Densidad	Tam Partición
10_2	10	20 %	10
10_4	10	40 %	10
10_6	10	60 %	10
10_8	10	80 %	10
20_2	20	20 %	20
20_4	20	40 %	20
20_6	20	60 %	20
20_8	20	80 %	20
30_2_30	30	20 %	30
30_4_30	30	40 %	30
30_6_30	30	60 %	30
30_8_30	30	80 %	30
30_2_15	30	20 %	15
30_4_15	30	40 %	15
30_6_15	30	60 %	15
30_8_15	30	80 %	15
30_2_10	30	20 %	10
30_4_10	30	40 %	10
30_6_10	30	60 %	10
30_8_10	30	80 %	10
30_2_5	30	20 %	5
30_4_5	30	40 %	5
30_6_5	30	60 %	5
30_8_5	30	80 %	5

Cuadro 1: Información de las instancias

6.2. Resultados

6.2.1. Planos de Corte

Hemos corrido un algoritmo de planos de corte utilizando las heurísticas de clique y odd-hole. Se configuró para que se realicen 5 iteraciones de planos de corte, y en cada una se busque hasta 10 cortes

de cada familia. El Cuadro 2 se muestra la evolución del valor objetivo en cada iteración.

Instancia	Corte	Rel 1	Rel 2	Rel 3	Rel 4	Rel 5	Opt
10_2	clique	2(sol entera)	-	-	-	-	2
	agujero	2	-	-	-	-	
	ambos	2(sol entera)	-	-	-	-	
10_4	clique	2	2,5	3	3	-	3
	agujero	2,33333	2,42857	2,5	2,5	2,5	
	ambos	2,5	2,66667	3	3	3	
10_6	clique	4(sol entera)	-	-	-	-	4
	agujero	2	2,6	3	3,36364	3,36364	
	ambos	4(sol entera)	-	-	-	-	
10_8	clique	5	5	6(sol entera)	-	-	6
	agujero	4	-	-	-	-	
	ambos	5	6(sol entera)	-	-	-	
20_2	clique	2	2	2,14286	2,28571	2,38095	4
	agujero	2	2	2,09063	2,2125	2,25949	
	ambos	2	2,26316	2,44	2,66667	3	
20_4	clique	2	2,25	2,5	2,66667	3	5
	agujero	3	3,05	3,16667	3,21739	3,21739	
	ambos	2	2,2	2,38462	2,4	2,4	
20_6	clique	2	2,5	3	3,5	4	8
	agujero	3	3	3,16667	3,2	3,3	
	ambos	4	4,10714	4,25	4,43704	4,475	
20_8	clique	2	2,91667	3	3,9	4,33333	9
	agujero	2	2	2,33333	2,5	2,5	
	ambos	5	5,38462	5,80303	6,08929	6,43636	
30_2_30	clique	2	2	2,2	2,2	2,44444	4
	agujero	2	2,33333	2,38095	2,66667	2,75	
	ambos	2	2,5	2,5	2,625	3	
30_4_30	clique	2	2,33333	2,5	2,5	3	6
	agujero	2	2,25	2,25	2,33333	2,375	
	ambos	2	2,5	2,5	3	3	
30_6_30	clique	2	2,5	2,5	3	3	?
	agujero	2	2,11111	2,11111	2,125	2,16667	
	ambos	3	3,2	3,2	3,27273	3,33333	
30_8_30	clique	6	6,5	6,5	7,5	8,5	?
	agujero	4	4,4	4,5	4,5	4,6	
	ambos	3	3,5	3,8	4,25	4,5	
30_2_15	clique	1	1,5	1,6	2	2	3
	agujero	1	1	1,07692	1,12821	1,21053	
	ambos	2	2	2	2	2	
30_4_15	clique	2	2	2	2	2	3
	agujero	1	1,07143	1,15	1,1875	1,33333	
	ambos	2,125	2,14634	2,22222	2,23529	2,28571	
30_6_15	clique	2	2,16667	2,16667	2,22222	2,66667	5
	agujero	1	1,16667	1,25	1,27273	1,27778	
	ambos	3	3,5	3,5	3,5	3,5	
30_8_15	clique	2	3	3,5	4	4	6
	agujero	1	1,16667	1,33333	1,33333	1,33333	
	ambos	3	3,07692	3,07692	3,33333	3,5	
30_2_10	clique	0,5	0,75	0,857143	0,894737	0,914798	2
	agujero	1	1,125	1,16667	1,28	1,29167	

	ambos	1	1,11111	1,17143	1,25	1,33333	
30_4_10	clique	0,5	0,75	0,823529	0,875	0,97619	2
	agujero	1	1	1,05882	1,15152	1,15152	
	ambos	1	1,25	1,33333	1,33333	1,33333	
30_6_10	clique	0,5	1	1,33333	1,4	1,5	3
	agujero	1	1	1	1,125	1,3	
	ambos	1	1,5	1,5	1,5122	1,54545	
30_8_10	clique	0,5	1,5	1,71429	1,75	1,875	4
	agujero	1	1	1,125	1,125	1,125	
	ambos	1	1,5	1,75	1,96296	2	
30_2_5	clique	0,5	0,5	0,5	0,5	-	1
	agujero	0,33333	0,33333	0,33333	0,350515	0,388889	
	ambos	0,333333	0,333333	0,363636	0,405882	0,440945	
30_4_5	clique	0,5	0,66667	1	1	1	1
	agujero	0,333333	0,375	0,448718	0,472362	0,490862	
	ambos	0,333333	0,352941	0,5	0,538462	0,581081	
30_6_5	clique	0,5	0,5	0,615385	0,639535	0,767857	2
	agujero	0,333333	0,4	0,473684	0,486322	0,503571	
	ambos	0,333333	0,466667	0,75	0,806349	0,820189	
30_8_5	clique	0,5	0,8	1,1828	1,21605	1,22222	2
	agujero	0,333333	0,363636	0,4	0,428571	0,5	
	ambos	0,333333	0,583333	0,9	1,13889	1,25	

Cuadro 2: Comparación de cortes

Se puede observar que la heurística del agujero impar arroja, en general, mejoras menos significativas del valor objetivo. Esto se puede deber a que un agujero impar de al menos 5 nodos no es tan fácil de encontrar en comparación a una clique de 2 ó 3 nodos, sobre todo siendo que las instancias utilizadas se generaron de manera aleatoria y sin ninguna característica particular. De hecho, cuando se trata de grafos con pocos nodos, nuestra heurística encuentra pocos o ningún agujero apropiado. También es de notar el hecho de que si bien los mejores resultados parecen ser arrojados por la combinación de ambas heurísticas, en algunos casos usar sólo cliques lleva a una mejor solución de la relajación. No nos queda muy en claro por qué sucede esto, pero se podría aventurar que el cambio producido por las sucesivas soluciones de la relajación al realizar ambas estrategias, llevaron al hallazgo de cliques y agujeros de menor importancia para estos casos particulares.

Por otro lado, cabe resaltar que la tabla parecería indicar que cuanto más denso es el grafo, las soluciones dadas al aplicar nuestros planos de corte se ubican más lejos de la solución original. Esto podría deberse a que al ser más denso el grafo, estamos tratando con un problema más complejo (aumento de la cantidad de restricciones), y en este contexto los cortes generados no impactan de manera significativa.

6.2.2. Métodos de resolución

Cada instancia fue resuelta una vez con *Branch and Bound* y una vez con *Cut and Branch*. En el primer caso, se desactivaron todos los preprocesamientos que CPLEX realiza automáticamente, de manera de tener un *Branch and Bound* puro, y en el segundo se aplicaron los cortes clique y odd-hole sobre el nodo raíz (5 iteraciones de planos de corte, hasta 10 cortes de cada familia por iteración).

Se estableció un tiempo límite de ejecución de 15 minutos y se experimentó con distintas estrategias de recorrido del árbol de enumeración y de selección de la variable de branching, a saber:

- *Recorrido:*
  - Best Bound Search: Recorre el árbol eligiendo cada vez el nodo con la mejor función objetivo para la relajación lineal del problema.

- Depth First Search: Recorre el árbol en profundidad, eligiendo cada vez el nodo más recientemente creado.

- *Branching:*
  - Minimum Infeasibility: Branchea sobre la variable con valor fraccionario más cercano a un entero.
  - Maximum Infeasibility: Branchea sobre la variable con valor fraccionario más lejano a un entero.

Las combinaciones utilizadas, entonces, fueron:

- Branch and Bound con Best Bound Search y Minimum Infeasibility. (B&B bb_min)
- Cut and Branch con Best Bound Search y Minimum Infeasibility. (C&B bb_min)
- Branch and Bound con Best Bound Search y Maximum Infeasibility. (B&B bb_max)
- Cut and Branch con Best Bound Search y Maximum Infeasibility. (C&B bb_max)
- Branch and Bound con Depth First Search y Minimum Infeasibility. (B&B dfs_min)
- Cut and Branch con Depth First Search y Minimum Infeasibility. (C&B dfs_min)
- Branch and Bound con Depth First Search y Maximum Infeasibility. (B&B dfs_max)
- Cut and Branch con Depth First Search y Maximum Infeasibility. (C&B dfs_max)

Los cuadros 3 y 4 muestran en comparación los tiempos de ejecución y la cantidad de nodos recorridos.

Instancia	B&B bb_min	C&B bb_min	B&B bb_max	C&B bb_max	B&B dfs_min	C&B dfs_min	B&B dfs_max	C&B dfs_max
10_2	0.00817704	0.00396299	0.00784397	0.00377202	0.00295782	0.00356007	0.00702	0.00376296
10_4	0.00556493	0.003582	0.00507998	0.00317883	0.00423908	0.00440192	0.00598311	0.00358891
10_6	0.00516009	0.00465703	0.00512791	0.00373411	0.00352597	0.00340414	0.00368905	0.00392509
10_8	0.00447989	0.00491095	0.00998902	0.003829	0.018265	0.006953	0.00925493	0.00416493
20_2	0.0121379	0.0125961	0.026387	0.0280669	0.11162	0.0244279	0.047262	0.03755
20_4	0.0472369	0.0542049	0.162116	0.053056	0.116251	1.01158	3.65766	0.0483739
20_6	289.089	2.20215	7.91918	42.5035	14.1231	3.82658	90.7334	36.3658
20_8	3.62331	108.505	1.80254	23.6802	1.53601	0.032114	2.87022	24.8268
30_2_30	1.7933	0.264302	1.80697	0.291555	6.27657	0.091531	21.6015	0.10904
30_4_30	EXCEEDED	56.9797	812.904	158.88	EXCEEDED	686.48	126.995	709.174
30_6_30	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED
30_8_30	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED	EXCEEDED
30_2_15	0.0639958	0.073993	0.0686121	0.0707431	0.0141408	0.015296	0.0890319	0.0394049
30_4_15	0.473762	0.285445	0.185458	0.0899439	0.0504482	0.108304	0.078253	0.146229
30_6_15	0.238924	0.468219	0.16859	4.55121	1.60669	0.204959	1.71758	1.25974
30_8_15	10.8477	45.9958	26.855	2.69702	6.27454	18.8419	4.8434	27.6312
30_2_10	0.052624	0.0360329	0.0632908	0.072603	0.00950503	0.0100551	0.00671983	0.00903082
30_4_10	0.037679	0.0424449	0.142905	0.0297949	0.0303299	0.0165229	0.0340099	0.023515
30_6_10	0.125693	0.122483	0.122681	0.118635	0.125298	0.064764	0.0432501	0.0670681
30_8_10	0.283003	1.48016	0.394975	0.141603	0.410445	0.849894	0.136766	0.07359
30_2_5	0.0100551	0.00908589	0.0103738	0.025409	0.00478601	0.0079329	0.00349712	0.00430799
30_4_5	0.020565	0.029243	0.0140061	0.042057	0.00444293	0.0161209	0.00715399	0.016217
30_6_5	0.0213292	0.030966	0.022845	0.107575	0.00678015	0.0124071	0.01565	0.0312059
30_8_5	0.0788221	0.0691769	0.0811901	0.0746622	0.00953984	0.028486	0.0813901	0.0927289

Cuadro 3: Tiempos de ejecución B&B y C&B

Instancia	B&B bb_min	C&B bb_min	B&B bb_max	C&B bb_max	B&B dfs_min	C&B dfs_min	B&B dfs_max	C&B dfs_max
10_2	0	0	0	0	0	0	0	0
10_4	0	0	0	0	0	0	0	0
10_6	0	0	0	0	0	0	0	0
10_8	0	0	0	0	39	0	0	0
20_2	0	0	7	0	323	0	60	0
20_4	38	0	262	0	184	1116	6843	0
20_6	196583	1001	14606	50729	27132	6569	161853	51037
20_8	2292	42067	2896	22205	2422	0	4439	33485
30_2_30	1393	10	792	0	6400	0	24016	0
30_4_30	(215522)	12687	421360	27420	(282400)	204199	90835	179715
30_6_30	(140238)	(76117)	(88764)	(101331)	(349096)	(323320)	(262261)	(404507)
30_8_30	(176566)	(151798)	(172051)	(119726)	(466448)	(241704)	(268941)	(253541)
30_2_15	4	0	0	0	0	0	0	0
30_4_15	225	0	17	0	7	0	0	0
30_6_15	10	40	6	1463	2008	118	1980	801
30_8_15	4213	9741	9690	498	5274	12795	3303	17965
30_2_10	3	0	6	0	0	0	0	0
30_4_10	0	0	10	0	3	0	0	0
30_6_10	9	0	9	0	174	0	5	7
30_8_10	65	453	107	0	500	880	69	0
30_2_5	0	0	0	0	0	0	0	0
30_4_5	0	0	0	0	0	0	0	0
30_6_5	0	0	0	13	0	0	0	0
30_8_5	10	0	18	0	0	0	19	0

Cuadro 4: Nodos recorridos B&B y C&B

Se puede observar, como es lógico, una cierta correspondencia entre la cantidad de nodos que se visitó en cada caso con el tiempo de ejecución consumido.

Por otro lado, se ve que para Best Bound, con ambas estrategias de branching, C&B parece recorrer en general menos nodos y tomar menos tiempo que B&B. Sin embargo, cuando se corre con DFS esta diferencia no es tan evidente. Esto puede deberse a que el DFS recorre el árbol en profundidad, lo que puede conducir a un recorrido de “fuerza bruta” que no permita podar las posibles soluciones de manera eficaz, independientemente de la selección de variable de branching. De todos modos, para los casos de coloreo con mayor cantidad de nodos, C&B muestra una mejor performance.

7. Conclusiones

Luego del trabajo realizado y de los resultados obtenidos y analizados, se desprenden las siguientes conclusiones:

Planos de corte: Se puede observar que la utilización de las desigualdades estudiadas producen una mejora en la solución de la relajación lineal,iteración a iteración, pareciendo tener mayor impacto la desigualdad clique. Quedaría pendiente ver si este impacto podría acrecentarse con una combinación diferente de cantidad de iteraciones de planos de corte y cantidad de cortes clique/agujero buscados por iteración.

Branch-and-Bound vs Cut-and-Branch: Si bien de antemano se podría considerar que Cut-and-Branch tendría una mejor performance en comparación con Branch-and-Bound, nos encontramos con que esta diferencia es mínima e incluso discutible. Esto podría deberse a una implementación demasiado ingenua de los algoritmos de separación, o bien a que las familias utilizadas para separar y la cantidad de cortes introducidos en el nodo raíz no sean suficientes para plasmar una diferencia importante.

Instancias DIMACS: Si bien la idea original del trabajo contemplaba realizar pruebas sobre instancias de coloreo DIMACS, no se logro trabajar con ellas ya que los tiempos de ejecución eran muy elevados.