DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico

Damian Eliel Aleman

28 de noviembre de 2013 — Investigacion Operativa

Integrante	LU	Correo electrónico
Aleman, Damian Eliel	377/10	damianealeman@gmail.com

Corrector:

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

El presente informe apunta a documentar el proceso de desarrollo del Trabajo Práctico de la materia Investigación Operativa, cursada correspondiente al segundo cuatrimestre del año 2013.

Este trabajo práctico consiste en el análisis de diferentes algoritmos para problemas de programcion lineal entera, para encontrar el conjunto independiente maximo de un grafo y evaluar los resultados de los distintos algoritmos sobre instancias de benchmark utilizadas en la literatura e instancias generadas aleatoriamente.

Dado un grafo $G = (V, E)$, un conjunto independiente de G es un subconjunto K de los nodos de V tal que para todo par de nodos de K, no existe una arista en E que los une. Es decir,

$$sea\ G = (V, E),\ K \subseteq V\ es\ conjunto\ independiente\ de\ G \Leftrightarrow (v, w) \notin E\ \forall\ v, w \in K$$

Para el desarrollo de los algoritmos se utilizarán las librerías que provee el paquete CPLEX, en particulae la Callable Library. Los algoritmos Cut and Branch y Branch and Cut requieren el desarrollo de algoritmos de planos de corte, que incluyen las siguientes desigualdades válidas:

- Cortes Clique
- Cortes Ciclo impar

Para resolver este problema se pide implementar:

- Un algoritmo branch and bound.
- Un algoritmo branch and cut.
- Un algoritmo cut and branch.

Luego de haber terminado con la programación de los algoritmos , se realizará' una experimentación con el objetivo de verificar y comparar tanto el gap como los resultados de los programas extrayendo así conclusiones sobre la performance y la optimalidad de los diferentes métodos propuestos en este trabajo práctico.

Para comenzar se describirá el modelo de un conjunto independiente que se eligió para las implementaciones:

$$\begin{array}{lll} \textbf{max} & \sum_{j=1}^{n} x_j & \\ \textbf{s.a} & x_i + x_j \leq 1 & \forall i, j \in E(G) \\ & x_j \geq 0 & j \in [1..n] \end{array}$$

Para los algoritmos de branch and cut y de cut and branch, se necesitarán planos de corte, que tengan desigualdades válidas. En este trabajo se realizarán los cortes clique y los cortes de ciclo impar.

2. Generador de Cliques

Para generar los los cliques, se implemento una heuristica que genera cliques Maximales, (no necesariamente óptimos). La heurística consiste en: Teniendo un conjunto de nodos tales que se pueden conectar con la clique actual (que se inicializa con todos los nodos adyacentes a un nodo), se van agregando nodos que pertenecen a este conjunto y se elimina del conjunto a los nodos no adyacentes al recién agregado. Asi obtenenmos una clique, ya que todos los nodos son adyacentes de a pares, que es maximal (ya que dejo de agregar nodos cuando el conjunto de seConectanConCliqueActual es vacio).

Los cortes clique son muy utiles a la hora de agregar desigualdades al problema debido a que restringen más a las posibles soluciones del problema generando así podas en el arbol de soluciones.

Debido a que en una clique todos los nodos son adyacentes de a pares, si alguno de los nodos de esa clique, pertenece a un conjunto independiente de G, luego ningun nodo de esa clique pertenece a ese conjunto independiente, es decir:

$$\forall K \sum_{i \in K} x_i \leq 1$$

con K una clique de G.

3. Generador de Ciclos Impares

Otro tipo de cortes que se generaron en este trabajo son los cortes de Ciclos Impares. En el grafo se encontraron ciclos impares haciendo dfs (depth first search) y cada vez que encuentro un ciclo, me fijo si es impar. Como se recorre el grafo en profundidad, es probable que los ciclos que encuentre sean de muchos nodos.

El corte del ciclo impar se modela de la siguiente manera:

$$\forall C \sum_{i \in C} x_i \leq \frac{|C| - 1}{2}$$

con C un ciclo impar de G.

4. Experimentación

Se generaron tres tipos de algoritmos:

- Un algoritmo branch and bound.
- Un algoritmo branch and cut.
- Un algoritmo cut and branch.

Es preciso aclarar que para los tres algoritmos se deshabilitaron todos los tipos de corte y preprocesamiento proporcionado por CPLEX.

Algoritmo Branch and Bound: Este algoritmo resuelve el problema aplicando sucesivamente el algoritmo simplex en la relajación lineal del modelo. Luego elige una variable x_j que no es entera y llama a la funcion recursiva para seguir resolviendo de dos maneras, con $\lceil x_j \rceil$ y con $\lfloor x_j \rfloor$ hasta que la solución de la relajación lineal sea entera.

Algoritmo Branch and Cut: Este algoritmo resuelve el problema aplicando sucesivamente el algoritmo simplex en la relajación lineal del modelo. Luego evalua los cortes proporcionados (los cortes clique y los de ciclo impar) y si hay algúna desigualdad violada se agrega a las restricciones. Por como está implementado se pude agregar solo algunas restricciones por cada iteracioón, en especial se agregan las que violan la desigualdad con mayor diferencia. Posteriormente como en branch and bound, se elige una variable x_j que no es entera y llama a la función recursiva para seguir resolviendo de dos maneras, con $\lceil x_j \rceil$ y con $\lfloor x_j \rfloor$ hasta que la solución de la relajación lineal sea entera.

Algoritmo Cut and Branch: Se basa en aplicar los cortes unicamente en el nodo raiz y luego ejecutar un branch and bound.

5. Resultados

Los algoritmos tienen tres parámtros. El primero es la cantidad de cliques que se buscan, el segundo la cantidad de ciclos imapres. El tercer parámetro es el límite de cortes que se pueden agregar por cada iteración. En el caso de que se encuentren menos cliques o ciclos impares que los pasados por parámetro se busca la mayor cantidad posible.

En la proxima tabla se muestra los resultados sobre la instancia frb30-15-1.mis [1].

Algoritmo	Parametros	Gap nodo Raiz	Gap Nodo FInal	Solucion	nodos
Branch and Bound		971	800	25	865148
Branch and Cut	100 100 10	643	548	27	71162
Branch and Cut	100 100 200	681	606	25	58921
Branch and Cut	450 12 460	741	505	25	49559
Branch and Cut	100 100 5	938	667	23	54314
Cut and branch	500 12 512	741	505	25	49559
Cut and branch	200 12 212	751	394	25	1233045

En la proxima tabla se muestra los resultados sobre la instancia $\text{dsjc}_2 50_5.in^2$.

Algoritmo	Parametros	Gap Nodo Final	Solucion
Branch and Bound		638	12
Branch and Cut	100 100 10	418	12
Branch and Cut	100 100 200	0(fin en 3128s)	12
Branch and Cut	100 100 5	364	12
Cut and branch	500 2000 2500	0(termina en 2176)	12
Cut and branch	200 2000 2200	0(fin 2216s)	12

6. Análisis de resultados y Conclusiones

En primer lugar se puede observar la diferencia entre Branch and Bound y Cut and Branch.

Como era esperable el gap en el nodo raiz en Branch and Bound es mayor que el de los demás algoritmos ya que no cuenta con los cortes.

Si bien en en Cut and Branch solamente se aplican los cortes en el nodo raiz, en el test random se observa que es más eficiente y por lo general el gap en el último nodo (cuando no termina el agoritmo) es mucho menor en comparación con branch and bound.

Comparando Branch and Bound con Cut and Branch, se ve (en el test random) que el Cut and Branch es más eficiente, teniendo por lo general un gap mas chico en el nodo final. Igualmente la diferencia no es tan grande como con Branch and Bound.

[1] grafo de http://www.nlsde.buaa.edu.cn/ kexu/benchmarks/graph-benchmarks.htm

[2] grafo de dsjc_250_5 de http://www.info.univ-angers.fr/pub/porumbel/graphs/