Búsqueda Dispersa para el problema de Ordenación Lineal de Corte Mínimo

Eduardo G. Pardo, Abraham Duarte, Juan J. Pantrigo

Dpto. de Ciencias de la Computación

ETS Ingeniería Informática

Universidad Rey Juan Carlos

C/Tulipán s/n, 28933 Móstoles (Madrid)

{eduardo.pardo,abraham.duarte,juanjose.pantrigo}@urjc.es

Resumen

El problema de optimización de Ordenación Lineal de Corte Mínimo (CM) consiste en encontrar un etiquetado de los vértices de un grafo, de modo que se minimice el máximo número de aristas que sobrepasan el espacio entre cada dos vértices consecutivos, al ordenar los vértices etiquetados sobre una línea recta. Este problema puede ser formulado como un problema de programación entera y se encuadra dentro de la categoría NP-Difícil. Las primeras aplicaciones prácticas del problema se remontan a los años setenta, destacando su utilización en el diseño de circuitos integrados. En este trabajo se propone un método heurístico basado en Búsqueda Dispersa para obtener soluciones aproximadas al problema. Los resultados obtenidos por los métodos propuestos han sido comparados con los métodos presentes en el estado del arte.

1. Introducción

El problema de optimización de Ordenación Lineal de Corte Mínimo (CM) consiste en encontrar un etiquetado de los vértices de un grafo, de modo que se minimice el máximo número de aristas que sobrepasan el espacio entre cada dos vértices consecutivos, al ordenar los vértices etiquetados sobre una línea recta. Sea $G(V, E)$ un grafo no dirigido y no ponderado tal que $V(|V| = n)$ es el conjunto de vértices y $E(|E| = m)$ es el conjunto de aristas del mismo. Un etiquetado f de G asigna los enteros 1, 2,... n a los vértices de G, de modo que cada vértice recibe una etiqueta diferente. Para dicha ordenación, el valor del corte de un vértice concreto ($C_f(v)$), que se denominará en lo que sigue corte posicional de v en f, viene determinado por el número de aristas $(u, w) \in E$ tales que $f(u) \leq f(v) < f(w)$, es decir:

$$C_f(v) = |(u, w) \in E : f(u) \leq f(v) < f(w)|$$

De este modo, el valor de la función objetivo para la ordenación f de G sería:

$$CM_f(G) = max_{v \in V} C_f(v)$$

En la Figura 1 se muestra un ejemplo de ordenación de los vértices de un grafo y el valor del corte de cada vértice en dicha ordenación. El valor de la función objetivo del ejemplo, viene determinado por el corte que genera el vértice que se encuentra en segunda posición.

El valor óptimo del problema del CM para un grafo, sería el menor valor de la función objetivo para todas las posibles ordenaciones de los vértices del grafo, y quedaría definido de la siguiente manera:

$$CM(G) = min\, CM_f(G)\ \forall f \in \Pi_n$$

Este problema es un problema NP-Difícil [7, 6] y puede ser expresado como un problema de programación entera [14]. El problema de la Ordenación Lineal de Corte Mínimo, también conocido como problema de la Anchura del

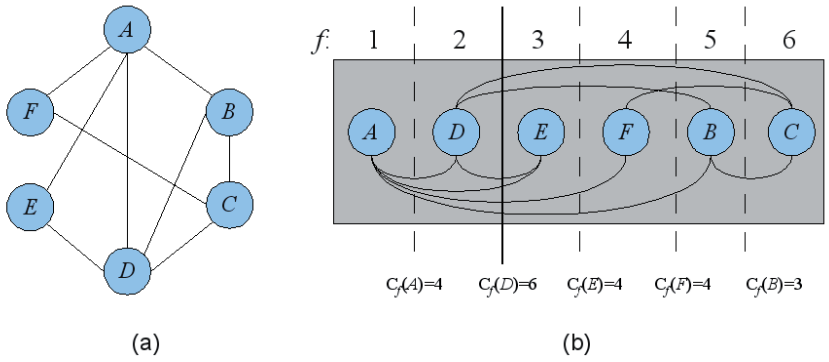

Figura 1: (a) Representación de un grafo de seis vértices y nueve aristas. (b) Representación de una ordenación de los vértices del grafo en (a) y del valor del corte para cada vértice en dicha ordenación.

Corte, fue utilizado en los años setenta como modelo teórico para determinar el número de canales en un circuito electrónico [1, 15]. En esta misma línea, [13] y [16] establecieron una relación entre el valor del corte mínimo de un grafo y el área necesaria para representar la conexión determinada por dicho grafo en un circuito con gran escala de integración. Más recientemente, ha sido utilizado en la determinación de la fiabilidad de redes [11], en tareas de recuperación de información [4] y en la migración de estaciones de comunicaciones [2].

En el estado del arte, caben destacar los métodos heurísticos incluidos en [5] donde se proponen diversos algoritmos constructivos y un conjunto de métodos de mejora. Los autores emplean la técnica del Enfriamiento Simulado (SA, del inglés *Simulated Annealing*) para combinar su mejor estrategia constructiva y su mejor método de mejora. En [2] los autores plantean un algoritmo tipo GRASP (del inglés, *Greedy Randomized Adaptive Search Procedure*) combinado con un mecanismo de Reencadenamiento de Trayectorias Evolutivo.

En este trabajo se propone un algoritmo heurístico de Búsqueda Dispersa (BD) para encontrar soluciones aproximadas al problema del CM. En la Sección 2 se describe el conjunto de algoritmos constructivos propuestos. En las secciones 3 y 4 se presentan los métodos de mejora y combinación. La Sección 5 está dedicada al método completo de BD. Por último, en las secciones 6 y 7 se expondrán los resultados experimentales y las conclusiones.

2. Algoritmos constructivos

En esta sección se describen los algoritmos constructivos propuestos para la generación de soluciones para el problema del CM. Cabe destacar que cualquier ordenación de los vértices del grafo considerado, constituye una solución al problema. Es deseable que un algoritmo constructivo que forme parte de un esquema de BD, genere soluciones de calidad, pero también soluciones diversas. Se proponen dos enfoques para el diseño del algoritmo constructivo: un enfoque estructural y un enfoque basado en la función objetivo. Ambos enfoques crean, en cada iteración, una lista de vértices candidatos a ser etiquetados (LC) formada por aquellos vértices con, al menos, un vértice adyacente etiquetado o, si esto no fuera posible (como es el caso inicial) formada por todos los vértices del grafo, y una Lista de Candidatos Restringida (LCR) que se forma atendiendo a diferentes criterios para cada algoritmo. En cada enfoque se construye una solución al problema añadiendo, en cada iteración, un nuevo vértice a la solución. Este vértice es seleccionado en función de un nuevo criterio dependiente del algoritmo, y recibe la etiqueta más baja que esté disponible en ese momento. El proce-

so se repite hasta que todos los vértices hayan sido etiquetados.

2.1. Constructivo basado en la función objetivo (C1)

En el enfoque basado en la función objetivo, se estudia cuánto aportaría a la función objetivo cada vértice candidato, si fuera el siguiente en ser etiquetado. Este algoritmo presenta, a su vez, dos variantes:

- Versión voraz-aleatorizada (C1-VA): En la versión voraz-aleatorizada, se evalúa cuánto aporta a la función objetivo cada uno de los vértices de la LC. A partir de la LC, se formará la LCR con aquellos vértices que menos aporten a la función objetivo, más aquellos cuya aportación esté dentro de un porcentaje por encima de la aportación mínima. De la LCR, se escoge un vértice aleatoriamente.
- Versión aleatorizada-voraz (C1-AV): En la versión aleatorizada-voraz, se selecciona aleatoriamente un subconjunto de vértices de la LC, formando una LCR. A continuación se evalúa cuánto aporta a la función objetivo cada uno de los vértices de la LCR, seleccionando el vértice que menos aporte.

2.2. Constructivo estructural (C2)

En el enfoque estructural, el criterio de selección del siguiente vértice a etiquetar se basa en la estructura del grafo para el que se va a construir la solución y, más concretamente, en el número de adyacentes no etiquetados que tenga cada vértice candidato. Al igual que la versión basada en la función objetivo, este algoritmo presenta dos variantes:

- Versión voraz-aleatorizada (C2-VA): En la versión voraz-aleatorizada, se seleccionan, de entre los vértices de la LC, aquellos que tengan un menor número de vértices adyacentes no etiquetados, formando así una LCR. A continuación, se elige un vértice de la LCR de manera aleatoria.
- Versión aleatorizada-voraz (C2-AV): Se elige, de manera aleatoria, un subconjunto de vértices de la LC, formando así la LCR. De esta lista se selecciona el vértice que tenga menos adyacentes no etiquetados.

La representación en pseudocódigo de los algoritmos constructivos propuestos puede verse en Algoritmo 1. Éste es un esquema general, siendo necesario que cada algoritmo constructivo particularice tanto el criterio de selección de los vértices que formarán parte de la LCR, como el criterio de selección del siguiente vértice a etiquetar, según lo anteriormente descrito.

Procedimiento `Constructivo`
`1. Sean` S `y` N `los conjuntos de vértices etiquetados y no etiquetados respectivamente`
`2. Inicialmente` $S = \oslash$ `y` $N = V$
`3. LC =` N
`4. Selección de un vértice` $u \in$ `LC en base a un criterio de selección`
`5. Asignación de la etiqueta k=1 a` u
`6.` $S = \{u\}, N = N \backslash \{u\}$
Mientras $N \neq \oslash$
`7. k = k + 1`
`8. LC =`$\{v \in N / (v,w) \in E \; \forall w \in S\}$
`9. LCR =`$\{v \in LC /$ `v cumpla criterio`$\}$
`10. Selección de un vértice` $u \in LCR$ `en base al criterio de selección`
`11. Asignación de la etiqueta k al vértice u`
`12.` $S = S \cup \{u\}, N = N \backslash \{u\}$

Algoritmo 1: Esquema constructivo general

3. Métodos de mejora

Los métodos de mejora (o búsqueda local) permiten encontrar soluciones de mejor calidad, explorando la vecindad de una solución. Dado que la estructura de una solución para el problema del CM constituye una ordenación, el mecanismo para encontrar soluciones de mejor calidad, consiste en realizar movimientos entre los elementos de la misma. En general, existen dos estrategias a la hora de mejorar una solución de este tipo: intercambios e inserciones.

3.1. Búsqueda local basada en intercambios

La estrategia de mejora basada en intercambios consiste en seleccionar dos vértices de la ordenación e intercambiar sus posiciones. Es necesario, por lo tanto, determinar qué vértices se desea intercambiar. En [2] se propone una búsqueda local exhaustiva basada en intercambios (BL1) que consiste en examinar todos los posibles intercambios en la ordenación. Puede verse el pseudocódigo de esta búsqueda local en Algoritmo 2.

Procedimiento Búsqueda-Local-BL1(S)
1. Sea S una solución al problema y K' el valor de la función objetivo (F.O.) de S
2. mejora $\leftarrow$ cierto
Mientras mejora = cierto
3. mejora $\leftarrow$ falso
Para todos pares (i,j)
4. S' =intercambio (i,j) en S
5. $K'' \leftarrow$ valor F.O. de S'
6. Si $K'' < K'$
7. $S = S'$
8. mejora $\leftarrow$ cierto

Algoritmo 2: Esquema Búsqueda Local BL1

3.2. Búsqueda local basada en inserciones

La estrategia de mejora basada en inserciones consiste en seleccionar un vértice de la ordenación e insertarlo en una posición determinada. Para ello, es necesario determinar qué vértice se desea mover y la posición a la que moverlo. En este trabajo se propone una búsqueda local basada en inserciones que trata de determinar tanto los vértices, como las posiciones más adecuadas. Se denomina Vértice Crítico (VC) a un vértice susceptible a ser movido y Posición Candidata (PC) a una posición destino a las que mover un VC. Dada una ordenación de los vértices del grafo, es necesario determinar el valor del corte de cada posición, para crear la lista de VC. Dicha lista estará compuesta por aquellos vértices que ocupen las posiciones en las que el valor del corte sea el mayor de la ordenación, o esté por encima de un porcentaje de éste. Para cada VC se determinan un conjunto de PC en las que se insertará el vértice, manteniendo el movimiento si el valor de la función objetivo se reduce, o bien se reduce el número de vértices en la lista de VC. Las PC de un VC se determinan en base a las etiquetas de los vértices adyacentes al VC. En concreto, las PC se sitúan en un rango alrededor de la mediana de las posiciones de los adyacentes. Puede verse el pseudocódigo de esta búsqueda local (BL2) en Algoritmo 3.

Procedimiento Búsqueda-Local-BL2(S)
1. Sea S una solución al problema y K' el valor de la función objetivo (F.O.) de S
2. mejora $\leftarrow$ cierto
Mientras mejora = cierto
3. mejora $\leftarrow$ falso
4. LVC = lista de VC de S
5. $\forall\, v \in$ LVC
6. LPC = lista de PC de v
7. $\forall\, i \in$ LPC
8. S' =inserción (v,i) en S
9. $K'' \leftarrow$ valor F.O. de S'
10. LVC' = lista de VC de S'
11. Si $(K'' < K')$ || tam(LVC') $<$ tam(LVC)
12. $S = S'$
13. mejora $\leftarrow$ cierto
14. Volver al paso 5

Algoritmo 3: Esquema Búsqueda Local BL2

4. Métodos de combinación

La combinación de soluciones consiste en construir una solución (denominada solución resultante) a partir de la combinación de dos o más soluciones previamente dadas (denominadas soluciones de referencia). Los mecanismos de combinación difieren en función del tipo solución que se desee combinar. En particular, cuando la estructura de la solución es una ordenación (como es el caso del problema del CM) es común utilizar el denominado mecanismo de combinación por votos [9]. En

este trabajo se proponen tres alternativas distintas de combinación por votos.

4.1. Combinación por votos clásica (CV-C)

En el mecanismo de combinación por votos clásico, las soluciones de referencia votan para que las características de sus soluciones aparezcan en la solución combinada. En cada iteración, cada solución de referencia vota por el elemento de su propia solución con posición menor, que no esté ya incluido en la solución combinada. Si la posición de los elementos por la que vota cada solución coincide, se elige aleatoriamente cuál de los dos elementos es añadido a la solución combinada. En cambio, si las soluciones votan por elementos que están en distintas posiciones, se añade a la solución combinada el elemento que ocupe una posición inferior.

4.2. Combinación por votos basada en el corte posicional (CV-CP)

Este mecanismo está basado en el corte generado en cada posición de las soluciones de referencia. En cada paso, cada solución de referencia vota por el elemento que ocupe una posición más baja en su propia solución, y que no haya sido previamente añadido a la solución combinada. En cada paso, se añade el elemento con menor valor de corte posicional. En caso de empate se selecciona uno de ellos de forma aleatoria.

4.3. Combinación por votos basada en la función objetivo (CV-FO)

El mecanismo de combinación por votos basado en la función objetivo pretende añadir a la solución combinada, en cada paso, los elementos que menos aporten a la función objetivo. Cada solución de referencia trata de añadir el elemento de su propia solución que se encuentre en una posición menor y que no haya sido previamente añadido. El elemento elegido es el que aporta menos a la función objetivo. En caso de empate se selecciona uno de ellos de forma aleatoria.

5. Búsqueda Dispersa

La BD es un método metaheurístico evolutivo que ha sido aplicado con éxito a un gran número de problemas de optimización. La primera descripción del método fue publicada por Fred Glover en [8] aunque las bases del esquema actual se sentaron en [10]. Se basa en la combinación de un conjunto de soluciones, denominado conjunto de referencia, sobre el que se realiza una exploración sistemática. Dicha combinación tiene como objetivo generar centroides, debiendo seleccionar pesos adecuados para la combinación y no valores al azar. Además, el método integra la combinación de soluciones con la búsqueda local.

En este trabajo se propone un algoritmo basado en BD para el problema del CM. La BD está compuesta por un generador de soluciones diversas, un método de mejora y un método de combinación, seleccionados experimentalmente de entre los métodos propuestos en las secciones 2, 3 y 4 respectivamente. Los resultados experimentales de estas comparaciones pueden consultarse en la Sección 6. Es necesario también, fijar el tamaño y la composición del conjunto de referencia. Para la creación de dicho conjunto, se han seleccionado inicialmente las cinco soluciones de mejor calidad de un conjunto de cien soluciones y las cinco más diversas a las primeras. La diversidad de una solución respecto a un conjunto se calcula evaluando la distancia entre la solución y cada una de las soluciones del conjunto. Se considera la distancia de una solución al conjunto como la mínima de todas las distancias anteriores. La función de distancia empleada ha sido el mecanismo de permutaciones relativas. Sean p, q dos soluciones del problema. La distancia relativa entre p y q, denotada por $d(p, q)$ contabiliza el número de veces que el elemento p_{i+1} no sucede al elemento p_i en q para $i = 1, \ldots, n-1$.

En cuanto a la generación de subconjuntos para su combinación se refiere, se han creado todos los posibles subconjuntos de dos soluciones a partir del conjunto de referencia. La actualización del conjunto de referencia se lleva a cabo en alguno de los siguientes supuestos:

- Cuando durante el proceso de combinación de soluciones se genere una solución de mejor calidad que la mejor del

conjunto de referencia, se añade al conjunto eliminando la más parecida a ésta.

- Cuando durante el proceso de combinación de soluciones se genere una solución de mejor calidad que la peor del conjunto de referencia, se añade al conjunto si es suficientemente diversa, sustituyendo a la más parecida.
- Cuando tras haber concluido el proceso de combinación de todos los pares disponibles en el conjunto de referencia no se haya añadido ninguna solución nueva al conjunto, se eliminan las cinco soluciones de peor calidad y se añaden las cinco más diversas a las soluciones que se conserven. Las nuevas soluciones se toman del conjunto inicial de soluciones diversas, generando nuevas soluciones en el momento que éste se agote.

6. Resultados experimentales

En esta sección se describen los resultados experimentales obtenidos durante la evaluación de los algoritmos propuestos. Inicialmente se presentan una serie de experimentos previos que permiten decidir cuáles, de entre los métodos propuestos para construir, mejorar o combinar soluciones, deben formar parte del algoritmo final de BD. Una vez justificada la composición de dicho algoritmo, se comparan los resultados obtenidos por éste, sobre un conjunto de instancias de referencia, con los métodos presentes en el estado del arte.

6.1. Conjunto de instancias utilizado

Para evaluar el rendimiento del algoritmo propuesto se ha empleado un subconjunto de instancias de la colección de matrices dispersas Harwell-Boeing[1]. Esta colección está formada por matrices provenientes de diferentes disciplinas científicas, de las cuales se han tomado todas aquellas menores de 400 vértices (55 en total). De este conjunto de evaluación, se han seleccionado el 15 % de las instancias (un total de 8) para llevar a cabo los experimentos preliminares.

[1]Las instancias pueden ser descargadas en http://www.uv.es/rmarti/paper/results/Harwell-Boeing %20BRP %20Instances.zip

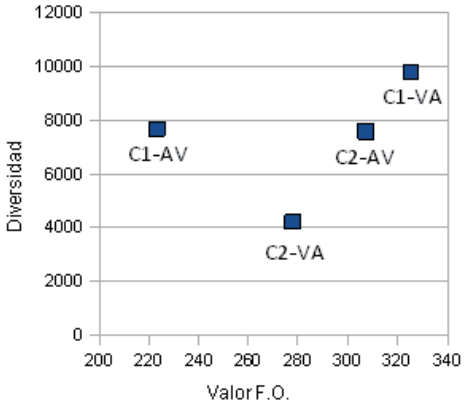

Figura 2: Representación del promedio de la calidad de las soluciones de los constructivos frente a la diversidad de las mismas.

6.2. Experimento previo: Constructivos

Utilizando las instancias de entrenamiento, se han evaluado los cuatro constructivos propuestos en la Sección 2 (C1-VA, C1-AV, C2-VA y C2-AV). Cada constructivo se ha ejecutado un total de 100 veces, calculando la media del valor de la función objetivo de las construcciones (calidad) y la media de la diversidad de cada solución con respecto al resto (medida mediante la distancia expuesta en la Sección 5). En la Figura 2 se representa la calidad frente a la diversidad promedio de las soluciones generadas con cada algoritmo. De entre los algoritmos propuestos, se selecciona el algoritmo C1-AV ya que ofrece soluciones de mejor calidad, siendo el segundo más diverso.

6.3. Experimento previo: Búsquedas locales

A continuación se evaluará el impacto de las dos búsquedas locales propuestas en la Sección 3 sobre las soluciones generadas por cada uno de los algoritmos constructivos propuestos en la Sección 2. Para ello, se han realizado 100 construcciones y 100 mejoras sobre las instancias de entrenamiento, reportando la media del mejor valor alcanzado por cada algoritmo para cada instancia. Como puede verse en el

resumen presentado en la Tabla 1, la combinación del constructivo C1-AV junto con BL2, es la que obtiene mejores resultados.

	C1-AV	C1-VA	C2-AV	C2-VA
BL 1	172.4	175.4	203.6	209.0
BL 2	170.1	171.8	174.9	182.9

Tabla 1: Promedio del mejor valor alcanzado por cada algoritmo constructivo junto con una búsqueda local tras 100 ejecuciones.

En la Tabla 1, se puede observar también que, tras aplicar las búsquedas locales, el constructivo C1-AV sigue siendo el que mejores resultados alcanza. Asimismo, el método de mejora BL2, parece funcionar mejor que BL1 en todos los casos.

6.4. Experimento previo: Mecanismos de combinación

Una vez seleccionado el constructivo C1-AV y la búsqueda local BL2 a incluir en el esquema algorítmico de la BD, es necesario seleccionar un mecanismo de combinación de soluciones. Para ello, se han evaluado los distintos mecanismos de combinación propuestos en la Sección 4. Cada mecanismo se ha incluido en un esquema de BD junto con el constructivo C1-AV y la búsqueda local BL2. Se ha ejecutado cada algoritmo sobre las instancias de entrenamiento, permitiendo realizar construcciones, mejoras y combinaciones durante 30 segundos para cada instancia. El promedio del valor de la mejor solución encontrada por cada algoritmo puede verse en la Tabla 2. El valor que aparece encabezando cada columna de dicha tabla, representa un porcentaje respecto al valor de diversidad de la solución de diversidad máxima generada (DMax) y se emplea para determinar si una solución es lo suficientemente diversa como para entrar en el conjunto de referencia.

Como se puede observar en la Tabla 2, el método de combinación utilizado no tiene un gran impacto sobre la calidad de las soluciones. No obstante, el método de combinación basado en el valor de la función objetivo (CV-FO) proporciona resultados ligeramente mejores, considerando una solución como suficientemente diversa si su diversidad es mayor que (DMax - 0.2·DMax).

	10 %	20 %	30 %	40 %	50 %
CV-C	164.6	166.4	166.3	165.1	166.1
CV-FO	164.9	162.5	164.0	164.5	164.4
CV-CP	163.8	164.6	166.3	165.3	164.4

Tabla 2: Comparativa de los diferentes métodos de combinación junto con los porcentajes de diversidad empleados en la actualización del conjunto de referencia.

6.5. Comparación con métodos previos

A tenor de los resultados de los experimentos previos. El algoritmo de BD final estará formado por el constructivo C1-AV, el método de mejora BL2 y el método de combinación CV-FO. Una vez decidida la composición del algoritmo de BD, se ha llevado a cabo una comparación con los métodos previos presentes en el estado del arte. En la Tabla 3 se muestran los resultados comparativos obtenidos al ejecutar los algoritmos durante 30 segundos sobre cada una de las 55 instancias descritas en la Sección 6.1.

	BD	Sahni [5]	Resende [3]
Media F.O.	122.95	140.52	160.51
Desviación	0.01	0.24	0.66

Tabla 3: Comparativa con los métodos del estado del arte, ejecutando cada algoritmo durante 30 segundos por instancia sobre las instancias descritas en la Sección 6.1.

7. Conclusiones

En este trabajo se ha propuesto un algoritmo de BD para el problema de la Ordenación Lineal de Corte Mínimo. En la propuesta, se incluyen diferentes métodos constructivos, de mejora y de combinación de soluciones. El esquema de BD propuesto incluye un método constructivo, uno de mejora y otro de combinación de soluciones, escogidos mediante pruebas experimentales de entre los métodos propuestos. El algoritmo de BD final obtiene

mejores resultados que los métodos presentes en el estado del arte, para el conjunto de instancias evaluadas.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia e Innovación (TIN2008-06890-C02-02, TIN2009-07516).

Referencias

[1] Adolphson, D. y T.C. Hu. *Optimal linear ordering*. SIAM Journal on Applied Mathematics, 25(3):403-423, 1973.

[2] Andrade, D.V. y M.G.C. Resende. *GRASP with path-relinking for network migration scheduling*. Proceedings of International Network Optimization Conference, 2007.

[3] Andrade, D.V. y M.G.C. Resende. *GRASP with evolutionary path-relinking* . Proceedings of Seventh Metaheuristics International Conference (MIC), 2007.

[4] Botafogo, R.A. *Cluster analysis for hypertext systems*. 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, pages 116-125, 1993.

[5] Cohoon, J. y S. Sahni. *Heuristics for the Board Permutation Problem*. Journal of VLSI and Computer Systems, 2, pages 37-61, 1987.

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

[7] Gavril, F.L. *Some NP-complete problems on graphs*. In Proceedings of the 11th conference on information Sciences and Systems, pages 91-95, 1977.

[8] Glover, F. *Heuristic for integer programming using surrogate constraints*. Decision Sciences, 8:533-549, 1977.

[9] Glover, F. *Tabu search for non-linear and parametric optimization (with links to genetic algorithms)*. Discrete Applied Mathematics 49, pages 231-255, 1994.

[10] Glover, F. *A template for scatter search and path relinking*. Lecture Notes in Computer Science 1363, pages 1-51, 1998.

[11] Karger, D.R. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. SIAM Journal on Computing, 29(2):492-514, 1999.

[12] Laguna, M. y R. Martí *Scatter Search methodology and implementations in C*. Kluwer Academic Publishers, 2003.

[13] Lengauer, T. *Upper and lower bounds on the complexity of the Min-Cut Linear Arrangement problem on trees*. SIAM. Journal on Algebraic and Discrete Methods Volume 3, Issue 1, pp. 99-113, 1982.

[14] Luttamaguzi, J., M. Pelsmajer, Z. Shen y B. Yang. *Integer programming solutions for several optimization problems in graph theory*. Technical report, DIMACS, 2005.

[15] Makedon, F. y I.H. Sudborough. *On minimizing width in linear layouts*. Discrete Applied Math., 23(3):243-265, 1989.

[16] Raspaud, A., O. Sýkora y I. Vrt'o. *Cutwidth of the de Bruijn graph*. RAIRO, Theoretical Informatics and Applications 26, 509-514, 1995.