

Heurísticos para el Problema del Etiquetado Lineal Mínimo

Juan José Pantrigo, Abraham Duarte, Vicente Campos, Rafael Martí

Resumen—El problema del etiquetado lineal mínimo (*Minimum Linear Arrangement problem* - MinLA) consiste en encontrar un etiquetado u ordenación de los vértices de un grafo tal que minimice la suma de los valores absolutos de las diferencias entre las etiquetas de vértices adyacentes. Éste es un problema NP-duro que constituye un reto para los métodos basados en optimización heurística. En este trabajo se han estudiado diferentes heurísticos con el objetivo de descubrir aquéllos más efectivos para abordar este problema. En concreto, proponemos la adaptación de métodos constructivos, de búsqueda local, GRASP y *Path Relinking* al problema del MinLA. Finalmente, se presenta experimentación para, en primer lugar, encontrar los valores más adecuados de los parámetros críticos de la búsqueda y, posteriormente, presentar una comparativa de nuestra propuesta con otras previas encontradas en la literatura.

Palabras clave—GRASP, *Path Relinking*, MinLA.

I. INTRODUCCIÓN

Sea $G = (V, E)$ un grafo con un conjunto de vértices V ($|V| = n$) y un conjunto de aristas E ($|E| = m$). Un etiquetado f de G asigna los enteros $1, 2, \dots, n$ a los vértices de G . Sea $f(v)$ la etiqueta del vértice v , donde cada vértice tiene una etiqueta diferente. La contribución $L(v, f)$ de un vértice v a la función objetivo se calcula como la suma de los valores absolutos de las diferencias entre $f(v)$ y las etiquetas de sus vértices adyacentes. Esto es:

$$L(v, f) = \sum_{u \in N(v)} |f(v) - f(u)| \quad (1)$$

donde $N(v)$ es el conjunto de vértices adyacentes a v . Entonces, el valor del etiquetado lineal de un grafo G con respecto a un etiquetado f es:

$$LA(G, f) = \frac{1}{2} \sum_{v \in V} L(v, f) \quad (2)$$

Por tanto, el etiquetado lineal mínimo $LA(G)$ de un grafo G es el mínimo valor de $LA(G, f)$ sobre todos los posibles etiquetados f . De otro modo, el problema del etiquetado lineal mínimo (MinLA) consiste en encontrar un etiquetado f que minimice $LA(G, f)$. Este problema NP-duro [2] está relacionado con otros dos problemas: el problema del ancho de banda (*bandwidth*) y el problema de la minimización del perfil (*profile minimization*). Sin embargo, como apunta [11], una solución óptima para uno de esos problemas no lo es necesariamente para los

otros. Por lo tanto, en este trabajo, restringimos nuestra atención a métodos y estrategias desarrollados específicamente para el MinLA. Este problema fue establecido por [5] y, desde entonces, se han propuesto numerosos algoritmos para resolverlo.

En [6] se introduce el método *Spectral Sequencing* (SSQ). Este método calcula los autovectores de la matriz laplaciana de G . A continuación, ordena los vértices de acuerdo al segundo autovector más pequeño. Como se estableció en [12], el método SSQ es efectivo porque tiende a asignar números próximos a vértices adyacentes, proporcionando así una buena solución para el MinLA.

En [11] se propone un método constructivo que etiqueta a los vértices en orden consecutivo. Los vértices se seleccionan de acuerdo a su grado con respecto a los etiquetados previamente.

Petit [12] revisa cotas inferiores y métodos heurísticos para el MinLA, propone nuevos algoritmos e introduce un nuevo conjunto de 21 instancias pequeñas y medianas ($62 \leq n \leq 1024$) para el MinLA. Con respecto a los heurísticos, Petit considera un método constructivo y una búsqueda local. El heurístico voraz de aumento sucesivo (*Successive Augmentation* - SAG) construye una solución paso a paso extendiendo un etiquetado parcial hasta que se etiquetan todos los vértices. En cada paso, se asigna a cada vértice la mejor etiqueta disponible. Se examinan los vértices en el orden dado por una búsqueda en anchura. Una vez se construye una solución, se aplican diferentes métodos de mejora: *Hill-climbing*, *Full-search* y recocido simulado (SA). En el primer método se seleccionan movimientos aleatorios. En el segundo se examina una vecindad completa de la solución y, en cada iteración, se busca el mejor movimiento posible. Finalmente, el SA implementa el parámetro de temperatura como en [7] para la selección del movimiento. Petit considera dos vecindades, denominadas *flip2* y *flip3*. El primero intercambia las etiquetas de dos vértices seleccionados, mientras que el segundo “rota” las etiquetas de tres vértices.

La experimentación en [12] muestra que la vecindad basada en el intercambio de dos etiquetas (*flip2*) produce mejores resultados que la rotación (*flip3*). Además, concluye que SA mejora a los demás métodos, aunque emplea mucho más tiempo de ejecución (que no se reporta en el trabajo). Por lo tanto, el autor recomienda emplear el método *Hill-climbing* así como los métodos de *Spectral Sequencing*. En [13] se presenta un SA más elaborado, donde se introduce la vecindad, *flipN*, basada en la distribución Normal de las distancias entre las etiquetas de los vértices. Además, para acelerar el método, la solución inicial se obtiene con el método SSQ. La com-

Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos. E-mail: juanjose.pantrigo@urjc.es

Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos. E-mail: abraham.duarte@urjc.es

Departamento de Estadística e Investigación Operativa, Universidad de Valencia. E-mail: vicente.campos@uv.es

Departamento de Estadística e Investigación Operativa, Universidad de Valencia. E-mail: rafael.marti@uv.es

binación SSQ+SAN mejora los métodos anteriores.

En dos trabajos recientes se utiliza el paradigma multiescala. Estos algoritmos, transforman iterativamente un problema multidimensional en varios de menor dimensión. Este proceso, denominado crecimiento (*coarsening*), permite resolver problemas multidimensionales de manera exacta. La solución se retro-proyecta a dimensiones más altas obteniendo una solución al problema original. En [8] se propone el algoritmo MS que combina el método espectral con técnicas multiescala. En [17] se introduce el método *Algebraic Multi-Grid scheme* (AMG), basado también en el paradigma multiescala pero con agregación ponderada de los nodos del grafo. Ambos métodos obtienen soluciones de gran calidad en tiempo lineal. Los autores experimentaron sobre 9 grafos extremadamente grandes ($78136 \leq n \leq 1017253$). Los tiempos de ejecución se encuentran entre los 27 y los 520 minutos.

Rodríguez-Tello et al. [16] proponen un algoritmo basado en la metodología SA, llamado *Two-Stage Simulated Annealing* (TSSA) que consta de dos etapas. En la primera, se construye una solución con el método de McAllister [11]. En la segunda, se ejecuta un SA basado en intercambio de etiquetas. Este método introduce dos nuevos elementos para resolver el MinLA: una vecindad combinada y una nueva función de evaluación. Dado un vértice v , la primera vecindad (que se selecciona con una probabilidad de 0.9) examina los vértices u cuya etiqueta $f(u)$ es cercana a la de la mediana de las etiquetas de los vértices adyacentes a v (con una distancia máxima de 2). La segunda vecindad, seleccionada con probabilidad 0.1, intercambia las etiquetas de dos vértices seleccionados aleatoriamente, con el objetivo de diversificar. El método evalúa las soluciones con una función que es más discriminante que la original LA . Los autores comparan su método con los mejores conocidos, concluyendo que el suyo es muy competitivo.

II. MÉTODOS CONSTRUCTIVOS

Un modo directo de construir una solución consiste en asignar sucesivamente a los vértices la menor etiqueta disponible. El método denominado “*frontal increase minimization*” (FIM) comienza creando una lista U de vértices sin etiquetar que, inicialmente, estará formada por todos los vértices del grafo (es decir, inicialmente $U = V$). Se selecciona el primer vértice aleatoriamente y se le asigna la etiqueta 1. En los siguientes pasos, se construye una lista de candidatos CL formada por todos los vértices en U con, al menos, un adyacente etiquetado. Se selecciona un vértice v de la CL , se le asigna la siguiente etiqueta disponible y se elimina de U . El método finaliza cuando todos los vértices han sido etiquetados.

McAllister [11] propuso un refinamiento para el FIM. Sea L el conjunto de vértices etiquetados y sea $d(v)$ el grado del vértice v . Entonces, $d_L(v)$ representa el número de vértices adyacentes a v que han sido etiquetados y $d_U(v)$ es el número de vértices adyacentes a v que aún no han sido etiquetados. Esta variante del constructivo FIM se basa en el cálculo de $sf(v) = d_U(v) - d_L(v)$ como me-

```

1. Inicialmente  $L = \emptyset$  y  $U = V$ .
2. Seleccionar aleatoriamente un vértice  $u$  de  $U$ .
3. Asignar a  $u$  la etiqueta  $l=1$ .  $L = \{u\}$ ,  $U = U - \{u\}$ 
WHILE ( $U \neq \emptyset$ )
4.  $l = l+1$ 
5. Construir  $CL = \{v \in U / (w,v) \in E \forall w \in L\}$ 
6. Calcular  $sf(v) = d_U(v) - d_L(v) \forall v \in CL$ 
7. Seleccionar el vértice  $u \in CL$  con menor  $sf(u)$ 
8. Etiquetar  $u$  con la etiqueta  $l$ 
9.  $U = U - \{u\}$ ,  $L = L \cup \{u\}$ 
ENDWHILE

```

Fig. 1. Pseudocódigo del método constructivo C1

didada del atractivo de un vértice v para ser etiquetado. La figura 1 muestra un pseudocódigo de este método, que llamaremos C1. En cada paso de C1, se selecciona el vértice con menor valor sf y se le asigna la menor etiqueta disponible. El procedimiento especifica además un mecanismo de desempate, en el paso 7 de la Figura 1. Cuando más de un vértice en CL tiene el mínimo valor sf , el método selecciona aquél con un número máximo de iteraciones en la CL . McAllister [11] mejora los constructivos basados en la estrategia FIM.

Nosotros proponemos una construcción GRASP basada también en el cálculo de $sf(v)$. GRASP (del inglés *Greedy Randomized Adaptive Search Procedure*), es un proceso iterativo multiarranque en el que cada iteración consta de dos etapas: construcción y búsqueda local. La fase de construcción propone una solución factible, y posteriormente se aplica una fase de búsqueda local que explora la vecindad de esa solución hasta que se encuentra un óptimo local [15]. La Figura 2 muestra el pseudocódigo del constructivo GRASP, al que llamamos C2.

En la construcción GRASP anterior, el parámetro th representa un umbral en la calidad de los elementos. Los elementos en CL con un valor $sf \leq th$ se incluyen en la RCL . Este parámetro de búsqueda se calcula como porcentaje α del rango de sf in CL :

$$th = msf + \alpha(Msf - msf) \quad (3)$$

donde

$$msf = \min_{v \in CL} sf(v), Msf = \max_{v \in CL} sf(v) \quad (4)$$

Se debe notar que si $\alpha = 0$, entonces $th = msf$, y el constructivo GRASP es equivalente al método de McA-

```

1. Inicialmente  $L = \emptyset$  y  $U = V$ .
2. Seleccionar aleatoriamente un vértice  $u$  de  $U$ .
3. Asignar a  $u$  la etiqueta  $l=1$ .  $L = \{u\}$ ,  $U = U - \{u\}$ 
WHILE ( $U \neq \emptyset$ )
4.  $l = l+1$ 
5. Construir  $CL = \{v \in U / (w,v) \in E \forall w \in L\}$ 
6. Calcular  $sf(v) = d_U(v) - d_L(v) \forall v \in CL$ 
7. Construir  $RCL = \{v \in CL / sf(v) \leq th\}$ 
8. Seleccionar aleatoriamente un vértice  $u \in RCL$ 
9. Asignar a  $u$  la etiqueta  $l$ 
10.  $U = U - \{u\}$ ,  $L = L \cup \{u\}$ 
ENDWHILE

```

Fig. 2. Pseudocódigo del método constructivo C2

1. Inicialmente $L = \emptyset$ y $U = V$.
2. Seleccionar aleatoriamente un vértice u de U .
3. Asignar a u la etiqueta $l=1$. $L = \{u\}$, $U = U - \{u\}$
- WHILE ($U \neq \emptyset$)
 4. $l = l+1$
 5. Construir $CL = \{v \in U / (w,v) \in E \ \forall w \in L\}$
 6. Calcular $sf(v) = d_U(v) - d_L(v) \ \forall v \in CL$
 7. Construir $CL_{msf} = \{v \in CL / sf(v) = msf\}$
 8. Construir $RCL = \{v \in CL_{msf} / C(v) \leq th_C\}$
 9. Seleccionar aleatoriamente un vértice $u \in CL$
 10. Asignar a u la etiqueta l
 11. $U = U - \{u\}$, $L = L \cup \{u\}$
- ENDWHILE

Fig. 3. Pseudocódigo del método constructivo C3

lister. Por el contrario, si $\alpha = 1$, entonces $RCL = CL$ y el constructivo GRASP equivale a la estrategia FIM.

Además, se propone otra variante, denominada C3, que considera la contribución del vértice a la solución como criterio de selección. En concreto, sea $C(v, l)$ la contribución de v a la solución actual cuando se le asigna la etiqueta l (y las etiquetas 1 a $l-1$ han sido ya asignadas). En términos matemáticos:

$$C(v, l) = \sum_{u \in N(v) \cap L} |f(u) - l| \quad (5)$$

El constructivo C3 considera únicamente los vértices con valor mínimo de sf en cada iteración. La RCL se calcula como los vértices con menor valor de sf y con un valor de C por debajo de un umbral th_C . Así, C se usa como un mecanismo de desempate cuando más de un vértice en CL alcanza el mínimo valor de sf . La Figura 3 muestra un pseudocódigo de este método.

El parámetro th_C en C3 es el umbral que establece los vértices con una contribución relativa pequeña a la solución actual. Se computa como un porcentaje β del rango en la lista de candidatos CL_{msf} :

$$th = dm_L + \beta(dm_L - dm_L) \quad (6)$$

donde

$$dm_L = \min_{v \in CL_{msf}} C(v, l), \quad dM_L = \max_{v \in CL_{msf}} C(v, l) \quad (7)$$

Si el parámetro $\beta = 1$, entonces $th_C = dM_L$ y todos los vértices en CL_{msf} estarán en RCL, y se llevará a cabo una selección aleatoria entre ellos. Por otro lado, $\beta = 0$, entonces $th_C = dm_L$ convirtiéndose en una selección voraz. En la Sección VI se comparan los tres constructivos C1, C2 and C3 y se estudia la influencia de los parámetros en su rendimiento.

III. MÉTODO DE MEJORA

En nuestra implementación, el mecanismo principal utilizado para ir de una solución a otra es el intercambio de etiquetas. Hemos desarrollado un mecanismo de mejora basado en la metodología de la cadena de eyección (*ejection chain*). Dada una etiqueta f y dos vértices u y v con etiquetas $f(u)$ y $f(v)$ respectivamente, se define $move(u, v)$ como el intercambio de las etiquetas $f(u)$ y $f(v)$. Sea g el etiquetado resultante de hacer $move(u, v)$.

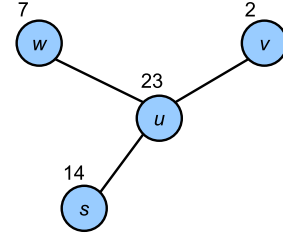


Fig. 4. Ilustración de un movimiento.

Es posible calcular el valor de g , $LA(G, g)$, del valor de f , $LA(G, f)$, como sigue:

$$LA(G, g) = LA(G, f) - MoveValue(u, v) \quad (8)$$

donde $MoveValue(u, v) = L(u, f) + L(v, f) - L(u, g) - L(v, g)$. Por lo tanto, conforme mayor es el valor de $MoveValue$, mejor es el movimiento.

Algunos heurísticos previos para el MinLA [16] también se basan en movimientos de intercambio y consideran que, dado un vértice u , la mejor etiqueta para él es la etiqueta mediana de sus adyacentes. Aunque la mediana minimiza la suma de las diferencias de etiquetas en valor absoluto, no siempre es la mejor elección. Considérese el grafo parcial de la Figura 4, en el que el vértice u tiene una contribución relativamente grande $LA(u, f) = |23 - 2| + |23 - 7| + |23 - 14| = 46$, y suponemos que intercambiamos su etiqueta (23) por otra.

De acuerdo con la regla anterior, podemos considerar la mediana de las etiquetas de sus adyacentes $med(u)$, como la mejor etiqueta para el intercambio. En este ejemplo, $med(u) = 7$ y corresponde al vértice w . Sea g el etiquetado después del cambio ($g(w) = 23$ and $g(u) = 7$); Entonces, obtenemos $LA(u, g) = 28$, reduciendo la contribución de u a la función objetivo LA. Sin embargo, si asignamos la etiqueta 8 (en lugar de la 7) al vértice u , la contribución de u sería: $|8 - 2| + |8 - 7| + |8 - 14| = 13$. Además, se debe notar que 7 es la mejor etiqueta para u cuando sus vértices adyacentes mantienen su etiqueta actual, de modo que no sirve de nada asignar la etiqueta 7 al vértice u y, al mismo tiempo, reemplazar la etiqueta de un vértice adyacente a u . Este ejemplo muestra que es mejor considerar una etiqueta cercana a la mediana y no asignada a ningún adyacente que la mediana en sí.

La situación ilustrada en la Figura 4 aparece en todos los casos en que el vértice considerado para el intercambio tiene grado impar (y, por tanto, la mediana es la etiqueta de uno de los adyacentes). Esto puede ser particularmente problemático cuando el vértice u tiene solamente un adyacente ($|N(u)| = 1$). En ese caso, la selección de la etiqueta mediana podría ciclar la búsqueda. En línea con esto, los trabajos anteriores [16], no limitan el movimiento, pero examinan un conjunto de etiquetas cercanas a la mediana con un distancia máxima de 2. Por lo tanto, indirectamente, resuelven esta situación. Este trabajo propone extender este conjunto de “buenas etiquetas” para el intercambio incluyendo un parámetro de búsqueda *width* y evitando las etiquetas de vértices adyacentes. El conjunto $CL(u)$ contiene las etiquetas candidatas para u :

$$CL(u) = \{l / |l - med(u)| \leq width, l \neq f(v) \forall v \in N(u)\}. \quad (9)$$

Nuestro método de mejora está basado en la metodología de la cadena de eyección (*ejection chain*). En [4] se introduce la noción de movimiento compuesto, contruidos a partir de una serie de componentes más simples. Este tipo de métodos también se conocen como métodos de profundidad variable. Dentro de ellos, los procedimientos de cadena de eyección se han demostrado útiles recientemente. Como se describe en [3], “los procedimientos de cadena de eyección se basan en la idea de generar secuencias compuestas de movimientos, que llevan de una solución a otra a través de pasos encadenados en los cuales los cambios en los elementos seleccionados propician que otros elementos sean expulsados se su estado actual, posición o valor de asignación”. En esta sección se adapta este concepto al problema del MinLA.

Supongamos que queremos intercambiar la etiqueta de un vértice u por la etiqueta $f(v)$ de otro vértice v , porque ese intercambio produce una reducción de $L(u, f)$. Sin embargo, este intercambio produce un aumento en el valor de $L(v, f)$, resultando en un movimiento que no mejora la calidad de la solución ($MoveValue(u, v) < 0$). Podemos entonces mantener la etiqueta $f(v)$ para u pero, en lugar de etiquetar v con $f(u)$, examinar otro vértice w y probar si su etiqueta $f(w)$ es adecuada para v (reduciendo $L(v, f)$). En términos de cadena de eyección, se puede decir que la asignación de $f(v)$ a u provoca que $f(u)$ sea “expulsada” de u a w (asignando $f(w)$ a v y definiendo un movimiento compuesto de profundidad dos que podemos representar como $move(u, v) + move(v, w)$). Es posible construir cadenas más largas aplicando la misma lógica.

En nuestro procedimiento de búsqueda local basada en la cadena de eyección, EC, definimos $move_{EC}(u, depth)$ a la cadena de eyección que empieza buscando una buena etiqueta para el vértice u . Para restringir la búsqueda y hacerla más eficiente, sólo se escanean etiquetas en $CL(u)$. Sea v el vértice con una etiqueta asociada $f(v) \in CL(u)$ con máximo $MoveValue(u, v)$. La cadena comienza haciendo $move_{EC}(u, depth) = move(u, v)$, es decir, intercambiando las etiquetas $f(u)$ y $f(v)$. Si $move(u, v)$ es un movimiento de mejora, se ejecuta y el proceso se detiene. En otro caso, se selecciona el vértice $v \in CL(u)$ tal que, cuando le asignamos la etiqueta $f(v)$ a u , la contribución de u a la función objetivo $L(u, f)$ se minimiza. Entonces, se busca un vértice w con una etiqueta $f(w) \in CL(v)$ adecuada para v . Se selecciona la etiqueta $f(w) \in CL(v)$ con mayor valor de $MoveValue(v, w)$. Si el movimiento compuesto de profundidad dos $move(u, v) + move(v, w)$ produce una mejora en la solución ($MoveValue(u, v) + MoveValue(v, w) \geq 0$), el movimiento se acepta y la cadena se detiene. En otro caso, la cadena continúa hasta que el movimiento compuesto produce una mejora o se llega al límite de profundidad preestablecido. Si ninguno de los movimientos ha producido una mejora, no se realiza ninguno y la exploración continúa con el siguiente vértice considerado

para el movimiento.

Se debe notar que se acepta el movimiento incluso cuando el valor es 0. Se ha probado que esta estrategia permite la exploración de más soluciones (comparada con la implementación en la que sólo se aceptan movimientos con valores positivos), obteniendo mejores resultados. Por otro lado, hemos encontrado que, cuando se aplica un movimiento, usualmente se cambian un número relativamente grande de etiquetas, y es útil aplicar la cadena de eyección desde el mismo vértice de nuevo. Por lo tanto, nos movemos al siguiente vértice considerado cuando la cadena de eyección no produce mejoras empezando en el vértice actual.

Una iteración global del método EC consiste en los siguientes pasos. En primer lugar, se ordenan los vértices en orden opuesto a su etiquetado en la construcción. A continuación se examinan los vértices en ese orden en busca de una mejora a través de un movimiento $move_{EC}$. La razón para esta ordenación es dar prioridad a los vértices más desfavorecidos en la etapa de construcción. Hemos probado además una variante en la que los vértices se ordenan de acuerdo a su contribución $L(u, f)$. Sin embargo, esta variante produce peores resultados que la primera. El método continúa iterando sólo si en la iteración anterior (entendiendo por iteración la inspección de todos los nodos) se ha realizado, al menos, un movimiento de mejora. De lo contrario, EC se detendrá.

El método EC tiene dos parámetros $width$ y $depth$ que controlan la distancia entre las etiquetas y el número de vértices, respectivamente, involucrados en el movimiento. Ambos parámetros juntos permiten aplicar EC con un tiempo de ejecución moderado; sin embargo, restringen la búsqueda a movimientos “pequeños” en el sentido de que sólo se consideran etiquetas cercanas a las de los vecinos del vértice considerado. Para diversificar la búsqueda e intentar movimientos más agresivos, aplicamos el método de “Hill Climbing” [12] en el que se seleccionan movimientos aleatoriamente al final de EC como una etapa de post-proceso para cada $iter_Hc$ iteraciones. La Sección VI reporta el estudio de estos parámetros.

IV. GRASP

Como se dijo anteriormente, el método GRASP está compuesto por una fase de construcción y otra de mejora. El elemento más importante de la etapa de construcción es que la selección en cada paso debe ser guiada por una función voraz que se adapte de acuerdo a las selecciones en pasos anteriores. En la Sección VI se comparan los tres constructivos descritos en la Sección II y se selecciona el que se ha empleado en nuestro algoritmo GRASP. La fase de mejora realiza una serie de movimientos, cadenas de eyección en nuestro método EC, buscando un óptimo local que será el resultado de una iteración completa del método.

Después de un número de iteraciones de GRASP, es posible estimar el porcentaje de mejora conseguido por la aplicación de la fase de mejora y utilizarla para incrementar la eficiencia de la búsqueda [9]. Se define el porcentaje de mejora en la iteración i del GRASP como:

$$P(i) = \frac{LA(G, f_i) - LA(G, f_i^*)}{LA(G, f_i)} \quad (10)$$

donde f_i es la solución (etiquetado) construido en la iteración i , $LA(G, f_i)$ es su valor, y f_i^* es la solución mejorada obtenida aplicando el método de mejora EC a f_i (y $LA(G, f_i^*)$ es su valor). Tras n iteraciones del GRASP, la media μ_P y la desviación típica σ_P de P se puede estimar como:

$$\hat{\mu}_P = \frac{\sum_{i=1}^n P(i)}{n}, \quad \hat{\sigma}_P = \sqrt{\frac{\sum_{i=1}^n (P(i) - \hat{\mu}_P)^2}{n-1}} \quad (11)$$

Por lo tanto, en una iteración i , estas estimaciones se pueden utilizar para decidir si es “probable” que una etapa de mejora sea capaz de mejorar la construcción actual para producir una solución que mejore a la mejor hasta el momento, f_{best} . En concreto, se calcula el porcentaje mínimo de mejora $imp(i)$ que es necesario para que una construcción f_i mejore a f_{best} del siguiente modo:

$$imp(i) = \frac{LA(G, f_i) - LA(G, f_{best})}{LA(G, f_i)} \quad (12)$$

Si el valor de $imp(i)$ es cercano a la estimación de μ_P , podemos considerar que, cuando apliquemos el método de mejora EC a la solución actual f_i , probablemente obtendremos una solución f_i^* mejor que f_{best} . Por lo tanto, con el objetivo de reducir el coste computacional, solamente aplicamos EC a las soluciones prometedoras. En términos matemáticos, si $imp(i) = \hat{\mu}_P + \delta \hat{\sigma}_P$, entonces aplicamos EC a f_i ; de lo contrario, la descartamos. El valor de δ es un parámetro de búsqueda que representa un umbral del número de desviaciones típicas permitidas. En la Sección VI, se experimenta el efecto de distintos valores de δ sobre el método.

V. PATH RELINKING

Nuestra implementación del *Path Relinking* tiene dos fases. En la primera, se genera un conjunto de soluciones élite ES (es decir, un pequeño conjunto de soluciones de calidad) con el método GRASP. En lugar de conservar solamente la mejor solución cuando ejecutamos GRASP, esta fase almacena un conjunto de buenas soluciones obtenidas con el método. Utilizamos $|ES| = 10$ como se recomienda en [14] en el contexto del problema del *bandwidth minimization*, pero extendiendo el significado de “bueno” para englobar tanto a las soluciones de calidad como a las diversas. En la segunda fase, se aplica el proceso de reencadenamiento para cada par de soluciones en el ES.

El ES juega un papel similar al conjunto de referencia en *Scatter Search* [10], ya que sus elementos se combinan para producir nuevas soluciones mejores. Por lo tanto, este conjunto debe contener buenas soluciones dispersas en el espacio de estados. En nuestro procedimiento, $P = f_1^*, f_2^*, \dots, f_{|P|}^*$ donde cada f_i^* se obtiene de la aplicación del método GRASP. Entonces, inicializamos el conjunto ES con las cinco mejores soluciones de P y, a

continuación, en lugar de añadir una a una las soluciones diversas como hace *Scatter Search*, resolvemos el problema de la máxima diversidad (MDP) para obtener las 5 soluciones más diversas en $P \setminus ES$. El problema del MDP consiste en encontrar, en un conjunto de elementos dado con sus correspondientes distancias entre ellos, el subconjunto más diverso de un tamaño dado. La diversidad del subconjunto elegido está dado por la distancia entre cada par de elementos. Como el MDP es un problema duro, empleamos el método GC2 [1] porque proporciona un buen compromiso entre la calidad de las soluciones y la velocidad.

Para definir una función de distancia, es importante notar que el etiquetado inverso define una solución equivalente para el MinLA. En términos matemáticos, sean $p = (p_1, p_2, \dots, p_n)$ y $q = (q_1, q_2, \dots, q_n)$ dos etiquetados de un conjunto de n vértices en los que $q_i = n - p_i + 1$. Entonces, se dice que p y q son permutaciones inversas y soluciones equivalentes al problema del MinLA (es fácil ver que $LA(G, p) = LA(G, q)$). De modo que, dos soluciones equivalentes deben tener una distancia entre ellas de 0. Calculamos la distancia como:

$$d(p, q) = \sum_{i=1}^n \partial_i \quad (13)$$

donde $\partial_i = 1$ si $p_i \neq q_i$ y $p_i \neq n + 1 - q_i$, y $\partial_i = 0$ en otro caso. Por ejemplo, supongamos dos soluciones $p = (6, 1, 2, 3, 4, 5)$ y $q = (1, 2, 3, 4, 5, 6)$. La distancia entre ellas será: $d(p, q) = 0 + 1 + 1 + 0 + 1 + 1 = 4$.

Apliquemos el proceso de reencadenamiento a cada par de soluciones en el ES. Dado el par de soluciones (f, g) , se explora la trayectoria de f a g (donde f es la solución inicial y g es la solución guía). Esta trayectoria consiste en asignar a cada vértice de la solución inicial las etiquetas de los vértices de la solución guía, uno a uno. Dado el par de soluciones (f, g) , sea $C(f, g)$ la lista de candidatos de los vértices a examinar en la trayectoria. En cada paso, se elige un vértice v de $C(f, g)$ y se etiqueta en la solución inicial con la etiqueta $g(v)$ en la solución guía. Para ello, en la solución inicial se busca el vértice u con etiqueta $g(v)$ y se realiza el intercambio $move(u, v)$. A continuación se elimina de $C(f, g)$ el vértice v . El conjunto de candidatos $C(f, g)$ se inicializa con un vértice seleccionado aleatoriamente. En el resto de iteraciones, cada vez que se elige un vértice y se elimina de $C(f, g)$, sus adyacentes se incluyen en la lista de candidatos.

En la metodología *Path Relinking* es conveniente añadir una búsqueda local para algunas de las soluciones visitadas y producir así mejores resultados [9] [14]. Hemos aplicado una búsqueda local basada en cadenas de ejecución EC sobre algunas de las soluciones construidas en la trayectoria. Se debe notar que dos soluciones consecutivas en la trayectoria sólo se diferencian en las etiquetas de dos vértices, por lo que no es muy eficiente aplicar la búsqueda local en cada paso del proceso de reencadenamiento. Por eso, se utiliza un parámetro pr que controla la aplicación del método de mejora. En concreto, EC se aplica pr veces en el proceso de reencadenamiento. En la Sección VI se presentan los resultados obtenidos para di-

ferentes valores de este parámetro. Tras la aplicación del *Path Relinking*, se devuelve la mejor solución encontrada como resultado final del método.

VI. RESULTADOS EXPERIMENTALES

Esta sección está dedicada a la descripción de los resultados experimentales obtenidos. En primer lugar, se describe una experimentación preliminar para comparar los métodos alternativos propuestos en las secciones anteriores. Además, ajustamos los parámetros de búsqueda para establecer la “mejor” combinación de los elementos y estrategias en nuestro algoritmo final. Todos los experimentos se han realizado en un procesador Intel Xeon a 3.2 GHz y 2.0 GB de RAM y todos los algoritmos han sido codificados en C. Hemos considerado el conjunto de instancias introducidas en [12], formado por 21 instancias pequeñas y medianas ($62 \leq n \leq 10240$ y $125 \leq m \leq 30380$). Las instancias están diseñadas para ser difíciles, es decir, que no pueden ser resueltas óptimamente por algoritmos de fuerza bruta. Este conjunto incluye grafos y árboles asociados a problemas reales: VLSI, *graph-drawing* e ingeniería (dinámica de fluidos y mecánica estructural).

Hemos llevado a cabo la experimentación preliminar sobre cuatro instancias del conjunto de datos: randomA1, c1y, bintree10 y mesh33x33. En el primer experimento preliminar comparamos el constructivo previo C1 [11] con C2(α) y C3(β) descritos en la Sección II. Hemos probado cinco valores para los parámetros α y β : 0.1, 0.2, 0.3, 0.4 y 0.5. Además, hemos probado una variante C2(rand) en la que el valor de α se selecciona aleatoriamente entre uno de estos cinco valores (y, de manera similar, con β en C3(rand)). La Tabla I muestra los resultados de estos tres métodos. Construimos 100 soluciones con cada método sobre cada instancia y reportamos el promedio sobre las cuatro instancias de la mejor solución encontrada (Avg.), el porcentaje de desviación promedio respecto a la mejor solución conocida (Dev.), así como el promedio de los tiempos de ejecución (Tiempo).

Los resultados de la Tabla I indican que los métodos propuestos C2 y C3 obtienen mejores soluciones que el método previo C1. Además, como se puede ver en el por-

TABLA I
COMPARATIVA DE LOS MÉTODOS CONSTRUCTIVOS.

Método	Avg.	Dev.	Tiempo
C1	272429.7	174.79 %	2.25
C2(rand)	267033.2	16.50 %	2.75
C2(0.1)	267033.2	16.50 %	2.50
C2(0.2)	268471.0	20.12 %	2.75
C2(0.3)	267161.5	19.97 %	2.75
C2(0.4)	267161.5	19.97 %	2.75
C2(0.5)	267033.2	16.50 %	2.75
C3(rand)	264475.5	13.77 %	2.75
C3(0.1)	266755.0	29.04 %	2.50
C3(0.2)	266852.7	31.71 %	2.75
C3(0.3)	266534.5	39.10 %	2.75
C3(0.4)	265768.2	36.76 %	2.75
C3(0.5)	262767.2	37.04 %	2.75

TABLA II
COMPARATIVA DE LOS MÉTODOS CONSTRUCTIVOS Y DE MEJORA.

Método	Avg.	Dev.
C4	263636.7	11.26 %
C4+EC(1,1)	264574.5	13.34 %
C4+EC(1,5)	263224.8	13.15 %
C4+EC(1,10)	262739.3	13.19 %
C4+EC(1,15)	264512.5	13.58 %
C4+EC(1,20)	262872.3	13.05 %
C4+EC(5,1)	262025.0	11.45 %
C4+EC(5,5)	261836.8	11.21 %
C4+EC(5,10)	263258.3	11.23 %
C4+EC(5,15)	260298.3	11.14 %
C4+EC(5,20)	262929.3	11.30 %
C4+EC(10,1)	261698.0	10.51 %
C4+EC(10,5)	260345.3	10.37 %
C4+EC(10,10)	263572.3	10.88 %
C4+EC(10,15)	262983.8	10.84 %
C4+EC(10,20)	261869.8	10.91 %

centaje promedio de desviación, los mejores resultados se obtienen con el método C3 con una selección aleatoria del parámetro β (C3(rand)). Sin embargo, en este experimento hemos comprobado empíricamente que, en algunas instancias, algunos métodos obtienen sistemáticamente mejores soluciones que otros (aunque, en promedio, C3 es el mejor). Por ejemplo, C1 siempre obtiene las mejores soluciones en la instancia bintree10 y C2 en las aleatorias. Por lo tanto, hemos considerado un método combinado, llamado C4, en el que seleccionamos aleatoriamente C1, C2(rand) o C3(rand) en cada construcción. C4 obtiene la menor desviación promedio (11.26 %), por lo que es el constructivo más robusto el elegido para el resto de la experimentación.

El segundo experimento previo prueba la efectividad del método de mejora descrito en la Sección III. Como base para comparar, utilizamos el método C4 construyendo durante 500 segundos sin aplicar ninguna mejora. Lo comparamos con la combinación de C4 y la mejora basada en cadenas de eyección EC con varios valores para sus dos parámetros: *width* y *depth*. Para ello, ejecutamos C4+EC(*width,depth*) durante 500 segundos para cada instancia y presentamos los mismos resultados que en la experimentación anterior.

La Tabla II muestra que la mejor combinación de parámetros de búsqueda para nuestra cadena de eyección es *width*=10 y *depth*=5, ya que C4+EC(10,5) presenta la mínima desviación promedio de 10.37 %. Se debe notar que, mientras que los trabajos previos [16] se limitan a un valor del parámetro *width* de 5, nosotros obtenemos mejores resultados con valores de 10. Por otro lado, la variante C4+EC(1,1) es C4 más una búsqueda local que, sorprendentemente, obtiene peores resultados (13.34 % de desviación promedio) que C4 únicamente (11.26 %). Esto es debido a que la búsqueda local consume mucho tiempo y, en 500 segundos, el método puede construir y mejorar menos de 50 soluciones para algunas instancias.

Como fue descrito en la Sección III la aplicación del método de mejora finaliza con la aplicación del método *Hill-Climbing* (HC) como post-proceso. Éste se ejecuta

TABLA III
POST-PROCESO *Hill-climbing* EN EL MÉTODO DE MEJORA.

<i>iter_Hc</i>	Avg.	Dev.	Tiempo
0	261538.5	10.12 %	505.8
V /20	255043.3	10.02 %	502.0
V /15	254594.5	9.64 %	504.8
V /10	255680.8	10.47 %	508.5
V /5	253935.5	9.96 %	505.3

durante *iter_Hc* iteraciones. La Tabla III presenta los estadísticos Avg., Dev. y Tiempo de ejecución del procedimiento C4+EC+HC con *iter_Hc* = 0, *n*/20, *n*/15, *n*/10 y *n*/5.

La Tabla III muestra que la aplicación de HC en el método de mejora EC produce una mejora marginal. Concretamente, la aplicación de HC para *n*/15 iteraciones reduce la desviación promedio de 10.12 % a 9.64 % consumiendo el mismo tiempo. En adelante, llamaremos GRASP a la combinación del constructivo C4, el método de mejora EC(10,5) y el postproceso HC.

El siguiente experimento previo prueba el mecanismo de filtrado descrito en la Sección IV para ejecutar el método de mejora sólo en aquellos casos en los que el valor de la construcción lo recomiende. Durante las primeras 20 ejecuciones, el método almacena el porcentaje de mejora que produce la búsqueda local EC. Después, cuando se construye una solución, se calcula su porcentaje de mejora mínimo necesario para mejorar la mejor solución encontrada hasta en momento. Si $imp(i) = \hat{\mu}_p + \delta \hat{\sigma}_p$ aplicamos EC y, de lo contrario, la descartamos. El valor de δ es un umbral del número de desviaciones típicas permitidas. Para medir la efectividad de esta estrategia y la mejor elección para el valor de δ ejecutamos el algoritmo GRASP 100 veces y aplicamos este filtro para las iteraciones 21 a 100. La Tabla IV muestra el promedio del número de iteraciones en el que el test recomienda no aplicar el método de mejora, #Saltos, el porcentaje de desviación promedio respecto a la mejor solución conocida, Dev., y el tiempo promedio de ejecución, Tiempo.

La Tabla IV muestra que cuando δ decrece, el número de saltos aumenta, como era de esperar. Sin embargo, el porcentaje de desviación promedio respecto del mejor, Dev., no presenta cambios importantes con los distintos valores de δ , indicando la robustez del filtro (las soluciones descartadas rara vez modificarán el resultado final). El mejor valor para δ es 0.5, ya que supone un ahorro de tiempo de cómputo sin afectar en gran medida a la calidad de la solución final.

TABLA IV
FILTRO GRASP.

δ	#Saltos	Dev.	Tiempo
0.5	51.7	10.51 %	292.0
1	37.5	10.40 %	550.2
1.5	30.0	10.90 %	485.2
2	25.7	10.86 %	485.7

TABLA V
Path Relinking.

<i>pr</i>	Avg.	Dev.	Tiempo
5	254964.5	9,59 %	824.5
10	253773.5	8,51 %	1041.7
15	253516.5	8,00 %	1268.0
20	252476.5	8,04 %	1521.7

El último experimento preliminar prueba el algoritmo *Path Relinking* (PR). Utilizamos un tamaño de 10 para el conjunto de soluciones élite ES. Inicializamos ES con las soluciones obtenidas durante 100 iteraciones del GRASP. Entonces, se ejecuta PR con las soluciones en ES y se devuelve la mejor. En PR, se aplica el método de mejora (EC(10,5)+HC) a *pr* de las soluciones en la trayectoria. La Tabla V muestra los resultados obtenidos para diferentes valores del parámetro *pr*.

La Tabla V muestra que, cuando *pr* aumenta, el algoritmo GRASP+PR mejora marginalmente la calidad de la solución final. Sin embargo, como era de esperar, el tiempo de cómputo también aumenta, ya que la aplicación del método de mejora consume mucho tiempo. Elegimos *pr* = 15 porque supone un buen compromiso entre calidad de la solución y velocidad.

El experimento final compara GRASP+PR con el constructivo de McAllister acoplado con un *Hill-climbing* [12], C1+HC, El *simulated annealing* SAN [12] y el *two stage simulated annealing* TSSA [16]. La Tabla VI muestra los resultados obtenidos sobre 21 instancias del conjunto de Petit. Ejecutamos los métodos una vez para cada instancia (a excepción de C1+HC que, dada su simplicidad, se ejecuta 1000 veces) con sus parámetros ajustados para que la ejecución dure 1000 segundos. La Tabla VI se divide en dos partes. En la parte superior, se muestran los resultados de cada algoritmo en cada instancia particular. En la parte inferior, se presenta la media del tiempo de ejecución, Tpo. Prom, el porcentaje de desviación promedio, Dev. Prom., y el número de mejores soluciones encontradas, #Mejores, que cada método encuentra.

La Tabla VI muestra que C4+HC y SAN son claramente inferiores a los otros métodos considerados, ya que la desviación respecto de la mejor es 49.89 % y 55.17 % respectivamente. GRASP+PR y TSSA obtienen desviaciones promedio de 3.03 % y 2.97 % respectivamente y 11 y 9 mejores soluciones sobre las 21 instancias en el conjunto. GRASP+PR presenta un tiempo promedio de ejecución de 529.14 segundos en promedio, que es mejor que los 870.57 segundos de TSSA. Si ejecutamos TSSA con los parámetros de [16], algunas instancias consumirían horas (aunque la desviación promedio se reduciría a 0.05 %).

VII. CONCLUSIONES

En este trabajo se propone un procedimiento heurístico basado en GRASP, para proporcionar soluciones de alta calidad a problemas de etiquetado lineal mínimo. Este procedimiento está acoplado con un post-proceso

TABLA VI
COMPARATIVA DE LOS MEJORES MÉTODOS SOBRE LAS INSTANCIAS DE PETIT.

	GRASP+PR		C4+HC		SAN		TSSA	
	Valor	Dev.	Valor	Dev.	Valor	Dev.	Valor	Dev.
randomA1	914882	2.31	950394	6.28	894205	0.00	948868	6.11
randomA2	6572444	0.00	6708192	2.07	6596880	0.37	6625307	0.80
randomA3	14336736	0.00	14463797	0.89	14346700	0.07	14441751	0.73
randomA4	1779181	1.17	1824564	3.75	1758560	0.00	1816732	3.31
randomG4	179138	0.00	206123	15.06	299571	67.23	185912	3.78
bintree10	4267	0.00	13951	226.95	14247	233.89	4440	4.05
hc10	523776	0.00	538116	2.74	540512	3.20	523776	0.00
mesh33x33	32703	0.00	35509	8.58	38481	17.67	33464	2.33
3elt	431737	0.00	1369880	217.30	867560	100.95	509337	17.9
airfoill	322611	0.00	867560	168.92	1369880	324.62	392989	21.8
whitaker3	1307540	0.00	4857190	271.48	4857190	271.48	1313857	0.48
c1y	65084	4.23	70896	13.54	73867	18.30	62441	0.00
c2y	82665	4.38	89029	12.41	89029	12.41	79199	0.00
c3y	136103	9.66	144902	16.75	163785	31.96	124117	0.00
c4y	125720	9.19	146651	27.36	146651	27.36	115144	0.00
c5y	109279	12.71	122652	26.51	123891	27.79	96952	0.00
gd95c	506	0.00	529	4.55	506	0.00	507	0.20
gd96a	114377	18.83	107945	12.15	111144	15.47	96253	0.00
gd96b	1421	0.35	1527	7.84	1483	4.73	1416	0.00
gd96c	519	0.00	531	2.31	519	0.00	523	0.77
gd96d	2414	0.84	2399	0.21	2421	1.13	2394	0.00
Tpo. Prom.	529.14		552.90		1274.14		870.57	
Dev. Prom.	3.03 %		49.89 %		55.17 %		2.97 %	
#Mejores	11		0		4		9	

Path Relinking, para mejorar la calidad de los resultados finales. Se han realizado experimentos para estudiar la contribución de los distintos elementos del método y también para comparar el método propuesto con otros en la literatura. Los experimentos preliminares muestran el potencial de los mecanismos propuestos, como el uso de cadenas de eyección en GRASP, el filtrado de soluciones de baja calidad y la selección de soluciones diversas para *Path Relinking*, que esperamos puedan ser útiles en otros problemas de optimización. GRASP+PR es altamente efectivo, superando incluso a los métodos en la literatura con tiempos de ejecución moderados (1000 segundos en instancias intermedias). Si los tiempos de ejecución no se limitan, el método de Rodríguez-Tello [16] obtiene los mejores resultados en promedio.

AGRADECIMIENTOS

Esta investigación ha sido subvencionada parcialmente por el proyecto Ref. TIN2006-02696 del MEC. Agradecemos a los profesores Rodríguez-Tello y Hao que hayan compartido sus resultados con nosotros.

REFERENCIAS

[1] Duarte, A., R. Martí, *Tabu Search and GRASP for the Maximum Diversity Problem*, European Journal of Operational Research, 178, pp. 71-84, 2007.

[2] Garey, M.R., D.S. Johnson, *Computers and Intractability. A guide to the theory of completeness*, W. H. Freeman and Company, New York, 1979.

[3] Glover, F., *Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems*, Discrete Applied Mathematics 65(1-3): 223-253, 1996.

[4] Glover, F., M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.

[5] Harper, L. H., *Optimal assignment of numbers to vertices*, SIAM J on Applied Mathematics, 12(1):131-135, 1964.

[6] Juvan, M., B. Mohar, *Optimal linear labelings and eigenvalues of graphs*, Discrete Applied Mathematics, 36(2):153-168, 1992.

[7] Kirkpatrick, S., C. Gelatt, M. Vecchi, *Optimization by simulated annealing*, Science, 220:671-680, 1983.

[8] Koren, Y., D. Harel, *A multi-scale algorithm for the linear arrangement problem*, LNCS, 2573:293-306, 2002.

[9] Laguna, M., R. Martí, *GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization*, INFORMS Journal on Computing, vol. 11, no. 1, pp. 44-52, 1999.

[10] Laguna, M., R. Martí, *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers, Boston, 2003.

[11] McAllister, A. J., *A new heuristic algorithm for the linear arrangement problem*, Technical Report TR-99-126a, Faculty of Computer Science, University of New Brunswick, 1999.

[12] Petit, J., *Combining Spectral Sequencing and Parallel Simulated Annealing for the MinLA Problem*, Parallel Processing Letters 13 (1), 71-91, 2003.

[13] Petit, J., *Experiments on the minimum linear arrangement problem*, ACM J of Experimental Algorithmics 8, 2003.

[14] Piñana, E., I. Plana, V. Campos and R. Martí, *GRASP and Path Relinking for the Matrix Bandwidth Minimization*, European J of Operational Research 153, 200-210, 2004.

[15] Resende, M.G.C. and C.C. Ribeiro, *Greedy Randomized Adaptive Search Procedures*, Handbook of Metaheuristic, Kluwer Academic Publishers, 219-249, 2003.

[16] Rodríguez-Tello, E., J. Hao, J. Torres-Jimenez, *An Effective Two-Stage Simulated Annealing Algorithm for the Minimum Linear Arrangement Problem*, Computers & Operations Research, in press, 2007.

[17] Safro, I., D. Ron, and A. Brandt, *Graph minimum linear arrangement by multilevel weighted edge contractions*, J of Algorithms, 60(1):24-41, 2006.