Búsqueda de Formulación Variable aplicada al problema de la Minimización de la Anchura de Corte en ordenaciones lineales

Eduardo G. Pardo[1], Juan J. Pantrigo[1], Abraham Duarte[1], and Nenad Mladenovic[2]

[1] Dpto. Ciencias de la Computación. Universidad Rey Juan Carlos. 28933, Móstoles.
{eduardo.pardo, juanjose.pantrigo, abraham.duarte}@urjc.es
[2] School of Mathematics. Brunel University. UB8 3PH, Uxbridge, London.
nenad.mladenovic@brunel.ac.uk

Resumen Muchos problemas de optimización son formulados como problemas min-max donde la función objetivo consiste en minimizar un valor máximo. En este caso, es habitual que muchas soluciones al problema tengan asociadas el mismo valor de la función objetivo. Cuando esto ocurre es difícil determinar cuál de las soluciones comparadas es más prometedora para continuar la búsqueda. En este artículo se propone una nueva variante dentro de la metodología Búsqueda de Vecindad Variable para abordar este tipo de problemas. Esta nueva variante, denominada Búsqueda de Formulación Variable, está basada en la utilización de formulaciones alternativas al problema para determinar la solución más prometedora, dentro de las funciones de agitación, búsqueda local y cambio de vecindad, en el esquema Búsqueda de Vecindad Variable Básico. La metodología propuesta es aplicada al problema de la Minimización de la Anchura de Corte en ordenaciones lineales y los resultados obtenidos son comparados con algoritmos previos en el estado del arte.

1. Introducción

Entre los problemas de optimización, existen muchos que consisten en minimizar un valor máximo o, alternativamente, maximizar un valor mínimo. Este tipo de problemas presentan un espacio de búsqueda con un horizonte plano. Es decir, muchas soluciones al problema tienen asociado el mismo valor de la función objetivo, $\mathcal{F}(x)$, para la formulación original del problema [11,12,9]. Esto hace que este tipo de problemas sean aún más difíciles de lo habitual, dado que la manera típica de búsqueda, conducida por movimientos dentro del espacio de soluciones factibles, resulta inerte en la mayoría de los casos.

El cambio de vecindad es una estrategia utilizada por la metodología Búsqueda de Vecindad Variable (VNS, acrónimo de su nombre en inglés *Variable Neighbourhood Search* [8]) para evitar caer en óptimos locales. Esta estrategia se basa en la idea de que los mínimos locales con respecto a una estructura determinada no tienen porqué serlo con respecto a otra [8,5]. Pese a que el cambio de vecindad

puede ser de utilidad para superar, parcialmente, el problema relativo al horizonte plano en el espacio de soluciones, existen problemas donde incluso esto no es suficiente para producir mejoras en la función objetivo tras un movimiento.

En este trabajo se propone una nueva variante de la metodología VNS, denominada Búsqueda de Formulación Variable (VFS, acrónimo de su nombre en inglés *Variable Formulation Search*) la cual combina el cambio de vecindad del esquema VNS con la utilización de formulaciones alternativas al problema para tratar de superar la dificultad anteriormente mencionada. Esta variante toma en consideración diferentes formulaciones de un problema cuando para una vecindad determinada, múltiples soluciones toman el mismo valor de la función objetivo. El cambio de formulación se lleva a cabo durante las etapas de búsqueda local, agitación y cambio de vecindad, dentro del esquema VNS.

Para validar la eficacia de la propuesta se ha aplicado la nueva variante al problema de la Minimización de la Anchura de Corte en ordenaciones lineales y se han comparado los resultados obtenidos con los de los métodos presentes en el estado del arte. El resto del artículo está estructurado de la siguiente manera: en la Sección 2 se describe la metodología propuesta. En la Sección 3 se presenta el problema abordado y la adaptación de VFS. Los resultados de esta validación se presentan en la Sección 4, terminando con las conclusiones en la Sección 5.

2. Búsqueda de Formulación Variable

En esta sección se presenta la Búsqueda de Formulación Variable, una nueva variante de la metodología VNS para problemas que presentan un horizonte plano del espacio de búsqueda.

2.1. VNS

La Búsqueda de Vecindad Variable es una metaheurística propuesta en [8] como un marco general para la resolución de problemas difíciles de optimización. Está basada en una idea simple: cambios sistemáticos de estructuras de vecindad durante el proceso de búsqueda. La metaheurística original ha sido ampliamente evolucionada con muchas extensiones, destacando: *Variable Neighbourhood Descent* (VND), *Reduced VNS* (RVNS), *Basic VNS* (BVNS), *Skewed VNS* (SVNS), *General VNS* (GVNS), Variable *Neighbourhood Decomposition Search* (VNSD) y *Reactive VNS*. En [4] y [5] se pueden encontrar amplias revisiones sobre las principales variantes de la metodología. De entre las anteriormente mencionadas, este trabajo centra su atención en el esquema BVNS como punto de partida.

Un pseudocódigo de BVNS se presenta en Algoritmo 1. Este procedimiento recibe tres parámetros de entrada: una solucion inicial (denotada como x), la máxima vecindad predefinida (k_{max}) y el tiempo máximo de cómputo (t_{max}). El procedimiento comienza realizando una perturbación a la solución actual a través de la función `Shake`, en el paso 5, y obteniendo así una nueva solución x'. A continuación, en el paso 6, un óptimo local x'', es alcanzado a través de la utilización de una estrategia de mejora. En el paso 7, se decide si BVNS necesita

explorar una vecindad mayor incrementando k (x'' es una solución peor que x) o no, lo que implica restablecer el valor de $k = 1$ (x'' es mejor que x). Los pasos 5 a 7 son repetidos hasta que se alcanza k_{max}. Este parámetro determina la máxima diferencia de vecindades que se probarán en la actual iteración cuando no hay mejoras en la solución. Los pasos 3 a 9 son repetidos hasta que t_{max} es alcanzado, comenzando en cada iteración desde la mejor solución encontrada.

Algoritmo 1 Pseudocódigo de BVNS

```
1: Procedimiento BVNS (x, k_max, t_max)
2: repetir
3:    k ← 1
4:    repetir
5:       x' ← Shake(x, k)
6:       x'' ← LocalSearch(x')
7:       NeighbourhoodChange(x, x'', k)
8:    hasta que k = k_max
9:    t ← CPUTime()
10: hasta que t > t_max
11: fin BVNS
```

2.2. VFS

Asúmase que además de la formulación original para un problema dado y su correspondiente función objetivo $\mathcal{F}_0(x) = \mathcal{F}(x)$ existen otras (p) formulaciones diferentes para el problema, que pueden denotarse como $\mathcal{F}_1(x)$, $\mathcal{F}_2(x)$, $\ldots, \mathcal{F}_p(x)$, $\forall\, x \in X$, donde x es una solución y X es el conjunto de soluciones factibles. Nótese que dos formulaciones son equivalentes si la solución óptima en una de ellas es equivalente a la solución óptima en la otra. Por lo tanto, dos formulaciones diferentes pero equivalentes, podrían tener el mismo valor de la función objetivo. No obstante, por simplicidad, considérense diferentes formulaciones como diferentes funciones objetivo $\mathcal{F}_i(x), i = 1, \ldots, p$.

La idea principal de VFS consiste en utilizar múltiples formulaciones de un mismo problema para comparar soluciones con el mismo valor de la función objetivo, en la formulación original. Para ello, se introduce el procedimiento `Accept`(x, x', p), presentado en el Algoritmo 2, en las tres etapas clave del esquema BVNS donde se realizan comparaciones de soluciones: `Shaking`, `LocalSearch` y `NeighbourhoodChange`. Como es evidente, si tras realizar un movimiento una solución no mejora en ninguna de las p formulaciones del problema propuestas, el movimiento es rechazado. El procedimiento `Accept` (x, x', p) recibe tres parámetros, donde x y x' representan las soluciones comparadas y p es el número de formulaciones consideradas. Así, para cualquier problema particular, es necesario diseñar diferentes formulaciones y decidir el orden en que serán empleadas dentro del procedimiento `Accept`. Las respuestas a estas preguntas son específicas del problema y en ocasiones difíciles de resolver.

Algoritmo 2 Procedimiento `Accept`

```
1: Procedimiento Accept (x, x', p)
2: para i = 0 a p hacer
3:    condicion1 = F_i(x') < F_i(x)
4:    condicion2 = F_i(x') > F_i(x)
5:    si (condicion1) entonces
6:       devolver Cierto
7:    si no
8:       si (condicion2) entonces
9:          devolver Falso
10:      si no
11:         continuar /*con la siguiente formulación*/
12:      fin si
13:   fin si
14: fin para
15: devolver Falso
16: fin Accept
```

3. Problema de la Minimización de la Anchura de Corte

El problema de la Minimización de la Anchura de Corte en ordenaciones lineales (CMP, por su nombre en inglés *Cutwidth Minimization Problem*) es un problema NP-Difícil [3] de tipo min-max. El CMP consiste en encontrar una ordenación de los vértices de un grafo en una línea, tal que minimice el máximo número de aristas que cortan una línea entre vértices consecutivos. Este problema puede formularse tanto de manera combinatoria como matemática [6].

La formulación combinatoria para el CMP, en la que se centra este trabajo, se describe de la siguiente manera: dado un grafo $G = (V, E)$ con $|V| = n$, una ordenación o etiquetado $f : V \to \{1, 2, \ldots, n\}$ de G asigna los enteros $\{1, 2, \ldots, n\}$ a los vértices en V, de tal modo que cada vértice recibe una etiqueta diferente. Dada la ordenación, el *cutwidth* de un vértice v, denotado como $CW_f(v)$, es el número de aristas $(u, w) \in E$ que satisfacen $f(u) \leq f(v) < f(w)$. Por lo tanto, $CW_f(v) = |\{(u, w) \in E : f(u) \leq f(v) < f(w)\}|$. Como consecuencia de la definición previa es posible definir el *cutwidth* de G con respecto a f, denotado como $CW_f(G)$, como el máximo de entre todos los valores de *cutwidth* entre todos sus vértices. Formalmente $CW_f(G) = \text{máx}_{v \in V}\, CW_f(v)$, por lo que el *cutwidth* óptimo de G se define como el mínimo $CW_f(G)$ sobre Π_n, que representa todas las posibles ordenaciones de los vértices del grafo de tamaño n:

$$CW(G) = \min_{f \in \Pi_n} CW_f(G)$$

Ejemplo. En las Figuras 1.a y 1.b se muestran respectivamente un grafo G y una ordenación f de los vértices de G junto con el valor del *cutwidth* asociado a cada vértice. En este ejemplo, el *cutwidth* de G con respecto a f es $CW_f(G) = 6$, debido a que es el máximo *cutwidth* de cualquier vértice en la ordenación. Está representado por una línea discontinua resaltada.

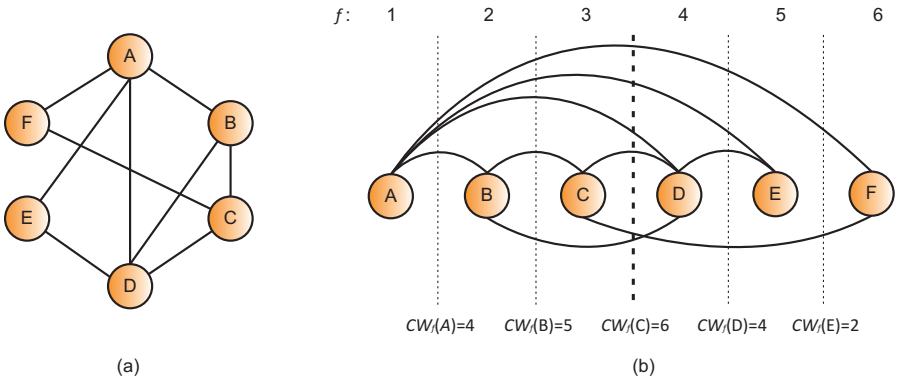

Figura 1. (a) Grafo G con 6 vértices y 9 aristas. (b) Ordenación f de los vértices del grafo en (a) con el correspondiente valor del *cutwidth* asociado.

Este problema ha sido abordado ampliamente desde diferentes perspectivas. Las primeras aproximaciones a la resolución heurística del problema se remontan a los años ochenta, cuando se propuso un algoritmo de *Simulated Annealing* (SA) para una versión generalizada del problema [2] . Tiempo después, en [1] se desarrolló un algoritmo tipo GRASP con *Path Relinking* (GPR). Las heurísticas más recientes para el problema pueden ser encontradas en [9] y [10] donde se propusieron, respectivamente, un *Scatter Search* (SS) y un VNS para el CMP. Existen también numerosos métodos exactos para tipos de grafos particulares, así como algún método genérico. Entre estos últimos, en [6] se propuso una formulación entera y en [7] un algoritmo de Ramificación y Acotación.

3.1. Formulación basada en subconjuntos

Esta formulación alternativa está basada en las ideas presentadas en [9]. Dada una ordenación f de los vértices de un grafo, es posible definir subconjuntos de vértices de acuerdo con el valor de su *cutwidth* asociado. Sea S_f^i el conjunto de vértices con valor de *cutwidth* i en la ordenación f. En términos matemáticos,

$$S_f^i = \{v \in V : CW_f(v) = i\}.$$

Considerando el ejemplo representado en la Figura 1.b, se tendrían los siguientes subconjuntos: $S_f^0 = \{\texttt{F}\}$, $S_f^1 = \emptyset$, $S_f^2 = \{\texttt{E}\}$, $S_f^3 = \emptyset$, $S_f^4 = \{\texttt{A,D}\}$, $S_f^5 = \{\texttt{B}\}$ y $S_f^6 = \{\texttt{C}\}$. Sea i_{max} el índice que identifica el conjunto que contiene los vértices con el mayor *cutwidth*. En este ejemplo, el valor de i_{max} es 6. Obviamente, $i_{max} = CW_f(G)$, por lo que el CMP es equivalente a reducir i_{max}. No obstante, con esta formulación alternativa se podrían comparar dos soluciones más allá del valor de i_{max}, tomando en cuenta la cardinalidad de los subconjuntos S_f^i.

En general, dadas dos soluciones, f y f', y un índice j ($1 \leq j \leq i_{max}$) se podría afirmar que f es mejor f' si y solo si $|S_f^j| < |S_{f'}^j|$ y $|S_f^i| = |S_{f'}^i|$ para

todo i, tal que $j < i \leq i_{max}$. Juntando las anteriores condiciones en una misma expresión matemática, se propone la definición de una nueva función objetivo para el CMP, $CW_f^2(G)$, de la siguiente manera:

$$CW_f^2(G) = \sum_{i=1}^{i_{max}} n^i |S_f^i|$$

Por lo tanto, el CMP puede ser reformulado así:

$$CW^2(G) = \min_{f \in \Pi_n} CW_f^2(G)$$

donde $CW^2(G)$ es el mínimo valor de CW_f^2 sobre todos las posibles ordenaciones Π_n. Notar que una solución óptima en $CW^2(G)$ es también óptima en $CW(G)$, pero lo contrario no es necesariamente cierto.

3.2. Formulación basada en la distancia a una solución ideal

Sean f_1 y f_2 dos soluciones al CMP. Se podría definir $d(f_1, f_2)$ como la función de distancia entre estas dos soluciones. Particularmente, si una de las dos soluciones (f^*) es óptima, entonces el CMP podría ser formulado como sigue:

$$CW^3(G) = \min_{f \in \Pi_n} d(f, f^*)$$

donde $CW^3(G)$ es el mínimo valor de $d(f, f^*)$ sobre todas las posibles ordenaciones en Π_n. Desafortunadamente, considerar la solución óptima como punto de referencia no es posible dado que, de hecho, es el objetivo del problema. No obstante, en lugar de utilizar una solución óptima, se podría considerar una solución «ideal» (quizá infactible) $f\_$, como punto de referencia.

La solución «ideal» debería tener un *cutwidth* al menos tan bueno como el óptimo. Esto significa que $CW_{f\_}(G)$ es una cota inferior para el CMP. A pesar de que minimizar $d(f, f\_)$ no necesariamente implica aproximarse a la solución óptima para el CMP, puede ser de utilidad a la hora de encontrar buenas soluciones. Cualquier solución (probablemente infactible) cuya función objetivo sea una cota inferior para el CMP, puede ser utilizada como solución «ideal». Aquí se propone utilizar el procedimiento propuesto en [7] (ver Sección 3.4 en dicho artículo) para construir una solución ideal.

Se proponen además dos funciones de distancia distintas. La primera, inspirada en la formulación presentada en la Sección 3.1, se basa en la comparación de subconjuntos. Dada una solución f, definimos $d_1(f, f\_)$ de la siguiente manera:

$$d_1(f, f\_) = \sum_{i=1}^{i_{max}} \text{abs}\left(|S_f^i| - |S_{f\_}^i|\right)$$

No obstante, la distancia d_1 presenta el inconveniente de que solo considera la diferencia absoluta entre los subconjuntos S_f^i y $S_{f\_}^i$, pero no dónde aparecen estas diferencias. La segunda distancia propuesta, d_2, está basada en la definición de

etiquetado. Cada vértice v en el grafo tiene asignada una etiqueta $f(v)$ diferente que indica la posición i que el vértice ocupa en la ordenación (Ver Figura 1.b). Considérese $f^{-1}(i)$ como la función que, dada una posición i, determina el vértice correspondiente asociado a esa posición. Por definición, la relación entre f y f^{-1} es tal que si $f(v) = i$ entonces $f^{-1}(i) = v$. A través de la minimización de d_2 se intenta obtener soluciones próximas a la solución «ideal». Específicamente, se minimizan las diferencias entre el *cutwidth* de los vértices situados en la misma posición en $f\_$ y f. Por lo tanto, la distancia se define de la siguiente manera:

$$d_2(f, f\_) = \text{mín}\Big(\sum_{1 \le i < n} \text{abs}(CW_f(f^{-1}(i)) - CW_{f\_}(f\_^{-1}(i))),$$

$$\sum_{1 \le i < n} \text{abs}(CW_f(f^{-1}(i)) - CW_{f\_}(f\_^{-1}(n-i)))\Big)$$

La distancia d_2 considera que el etiquetado inverso es equivalente para el CMP. Sean $f = (v_1, v_2, \ldots, v_n)$ y $f' = (v'_1, v'_2, \ldots, v'_n)$ dos etiquetados de un conjunto de n vértices donde $f^{-1}(i) = f'^{-1}(n-i+1)$, $1 \le i \le n$. Entonces, se puede decir que f y f' son etiquetados inversos y, por lo tanto, equivalentes para el CMP. En particular, $CW_f(f^{-1}(i))$ es igual a $CW_{f'}(f'^{-1}(n-i))$, $\forall\, 1 \le i < n$.

3.3. Búsqueda de Formulación Variable para el CMP

El objetivo principal de este trabajo no es describir con detenimiento las fases de construcción de la solución, estrategia de agitación, búsqueda local y cambio de vecindad (elementos estándar en VNS). Por ello, se han empleado las estrategias propuestas en [10] como punto de partida para aplicar las estrategias aquí propuestas. La modificación principal ha consistido en añadir el procedimiento `Accept` (ver Algoritmo 3) a las fases de agitación, búsqueda local y cambio de vecindad a la hora de comparar soluciones. En él se utilizan la formulación original ($CW(G)$) y las formulaciones alternativas ($CW^2(G)$ y $CW^3(G)$) para comparar dos soluciones f_1 y f_2. Si $CW_{f_1}(G) < CW_{f_2}(G)$, se puede concluir que f_1 es mejor que f_2 (paso 2). En cambio, si ambas soluciones tienen el mismo valor de la fución objetivo original, no es posible asegurar cuál es mejor sin la utilización de formulaciones alternativas. En particular, si $CW_{f_1}(G) = CW_{f_2}(G)$, se puede considerar que f_1 es más prometedora que f_2 cuando $CW^2_{f_1}(G) < CW^2_{f_2}(G)$ (paso 4). Similarmente, también se puede considerar f_1 más prometedora que f_2 cuando $CW_{f_1}(G) = CW_{f_2}(G)$ y $CW^3_{f_1}(G) < CW^3_{f_2}(G)$ (paso 6). Por último, si ninguna de las anteriores condiciones es satisfecha, el procedimiento devuelve Falso, lo que significa que f' no es más prometedora que f.

4. Resultados experimentales

En esta sección se presentan los resultados experimentales llevados a cabo para validar la eficiencia de la metodología propuesta. En primer lugar se presentan los conjuntos de instancias empleados para su evaluación, a continuación

Algoritmo 3 Procedimiento Accept para el CMP

```
1: Procedimiento Accept (f, f')
2: si (CW_f'(G) < CW_f(G)) entonces
3:    devolver Cierto
4: si no, si (CW_f'(G) = CW_f(G)) y (CW^2_f'(G) < CW^2_f(G)) entonces
5:    devolver Cierto
6: si no, si (CW_f'(G) = CW_f(G)) y (CW^3_f'(G) < CW^3_f(G)) entonces
7:    devolver Cierto
8: si no
9:    devolver Falso
10: fin si
11: fin Accept
```

se muestra el impacto de la introducción de las formulaciones alternativas al problema y, por último, se compara la mejor versión del algoritmo con algoritmos previos del estado del arte. Los algoritmos fueron implementados en Java 6 SE y los experimentos realizados en un Intel Core 2 Quad CPU, con 6 GB RAM.

Para evaluar los algoritmos propuestos se han utilizado dos conjuntos de instancias ("*Grid*" y "*Harwell-Boeing*"). Estas instancias fueron propuestas en [9], donde pueden encontrarse descripciones detalladas de las mismas. Además, todas las instancias se encuentran disponibles en `http://www.optsicom.es/cutwidth`.

El primer experimento está destinado a la comparación del uso de las diferentes formulaciones propuestas para el problema, en combinación, en lugar de utilizar solo una de ellas. Para realizarlo se ha utilizado un subconjunto representativo de instancias de los conjuntos "*Grid*" y "*Harwell-Boeing*". Concretamente se han tomado 16 instancias (el 10 % de los conjuntos).

	BVNS	**VFS$_1$**	**VFS$_2$**	**VFS$_3$**
Promedio F.O.	137.31	93.56	91.56	90.75
Desviación (%)	192.44	60.40	49.23	48.22
Tiempo CPU (s)	30.17	30.47	30.50	30.96

Tabla 1. Impacto de la utilización de formulaciones alternativas.

En la Tabla 1, se presentan los resultados obtenidos con cuatro algoritmos diferentes tras ser ejecutados durante 30 segundos sobre cada una de las 16 instancias mencionadas. La columna BVNS representa el algoritmo basado en la metodología BVNS que utiliza únicamente la formulación original para el CMP en el proceso de búsqueda. VFS$_1$ es la primera versión de la variante metodológica propuesta en este trabajo y está basada en la combinación del BVNS mencionado junto con la formulación CW^2 presentada en la Sección 3.1. VFS$_2$ es equivalente a VFS$_1$ pero utilizando esta vez CW^3 (Ver Sección 3.2) en lugar de CW^2. Finalmente, la cuarta columna (VFS$_3$) combina la formulación original con las dos anteriores empleando el procedimiento descrito en la Sección

3.3. Todos los algoritmos fueron configurados con $k_{max} = 0{,}1n$ (siendo n el número de vértices del grafo) y comenzaron desde la misma solución inicial.

Los resultados presentados en la Tabla 1 claramente confirman que la utilización de más de una formulación para determinar con qué solución continuar la búsqueda es más eficaz que emplear solo la formulación original. En este sentido, la combinación de varias formulaciones (columna VFS_3) fue ligeramente mejor que la utilización de cada formulación por separado, por lo que es la configuración elegida para la experimentación final.

Finalmente, se compara la mejor versión de VFS con los principales algoritmos presentes en el estado del arte sobre los conjuntos de instancias completos. Específicamente, los procedimientos comparados son: Simulated Annealing (SA), [2]; Greedy Randomized Adaptive Search Procedure con Path Relinking (GPR), [1]; Scatter Search (SS), [9] y VFS. El algoritmo de VFS fue configurado de la siguiente manera: se construyeron 1000 soluciones con el procedimiento constructivo descrito en [10], de entre las que se escogió la mejor como punto de partida. Se empleó la búsqueda local basada en inserciones propuesta en [10]. El parámetro k_{max} se estableció a $0{,}2n$ de manera experimental y, el máximo tiempo de cómputo para cada instancia se fijó en $0{,}4n$, siendo n el número de vértices de cada instancia. Finalmente, se utilizó la variante VFS_3 donde las dos formulaciones alternativas propuestas fueron utilizadas para determinar qué solución es la más prometedora para continuar la búsqueda.

En la Tabla 2 se presenta la comparación con los algoritmos anteriormente mencionados sobre el conjunto de instancias *Grid* y *HB*. En ambos casos es posible observar como la variante de VFS elegida mejora a los algoritmos previos del estado del arte. En particular, en el conjunto *Grid*, VFS obtuvo un 3.2 % de desviación, mientras que el siguiente mejor algoritmo (SS) obtuvo un 7.8 % de desviación, más del doble en más del doble del tiempo de cómputo. De manera similar, en el conjunto *HB*, VFS es nuevamente el mejor método, obteniendo un 1.8 % frente al método más cercano (SS) con un 3.4 %. Nuevamente, en más del doble del tiempo de cómputo. Para complementar esta información, se ha aplicado el test de Friedman a las mejores soluciones obtenidas por cada método. Este experimento fue llevado a cabo de manera independiente para los conjuntos *Grid* y *HB*. En ambos casos, el p-valor resultante de 0.000 indica que existen diferencias significativas entre los métodos comparados.

	Grid (81)				**HB (87)**			
	GPR [1]	**SA** [2]	**SS** [9]	**VFS**	**GPR** [1]	**SA** [2]	**SS** [9]	**VFS**
Promedio F.O.	38.4	16.1	13.0	12.2	364.8	346.2	315.2	314.4
Desviación (%)	201.8	25.4	7.8	3.2	95.1	53.3	3.4	1.8
#Mejores	2	37	44	59	2	8	47	61
Tiempo CPU (s)	235.2	216.1	210.1	90.3	557.5	435.4	430.6	128.1

Tabla 2. Comparación con los algoritmos del estado del arte (Instancias Grid y HB).

5. Conclusiones

En este trabajo se ha propuesto una nueva variante de la metodología VNS. La nueva variante, denominada Búsqueda de Formulación Variable, utiliza las características principales de VNS y añade nuevos criterios para la comparación de soluciones. El propósito de esta metodología es determinar qué solución es más prometedora para continuar la búsqueda, más allá del valor de la función objetivo original. Este hecho es de especial relevancia en problemas de tipo min-max donde existen muchas soluciones con el mismo valor de la función objetivo. Se ha ilustrado la efectividad de la propuesta mediante su aplicación al problema de la Minimización de la Anchura de Corte en ordenaciones lineales. Los resultados obtenidos han sido comparados favorablemente en términos de calidad y tiempo de cómputo con los métodos previos más detacados en el estado del arte.

Agradecimientos Esta investigación ha sido parcialmente financiada por los proyectos de investigación con referencias: TIN2009-07516, TIN2011-28151 y TIN2012-35632.

Bibliografía

1. D. V. Andrade and M. G. C. Resende. GRASP with Path-Relinking for Network Migration Scheduling. In *Proceedings of International Network Optimization Conference (INOC)*, 2007.
2. J. Cohoon and S. Sahni. Heuristics for backplane ordering. *Journal of VLSI and Computer Systems*, 2:37–61, 1987.
3. F. Gavril. Some NP-complete problems on graphs. In *Proceedings of the Eleventh Conference on Information Sciences and Systems*, pages 91–95, Baltimore, 1977.
4. P. Hansen, N. Mladenovic, J. Brimberg, and J. A. Moreno Pérez. *Handbook of metaheuristics*, chapter Variable Neighbourhood Search, pages 61–86. Number 146. Kluwer, 2nd edition, 2010.
5. P. Hansen, N. Mladenovic, and J. A. Moreno Pérez. Variable neighbourhood search: algorithms and applications. *Annals of Operations Research*, 175:367–407, 2008.
6. J. Luttamaguzi, M. Pelsmajer, Z. Shen, and B. Yang. Integer programming solutions for several optimization problems in graph theory. Technical report, Center for Discrete Mathematics and Theoretical Computer Science, DIMACS, 2005.
7. R Martí, J. J. Pantrigo, A. Duarte, and E. G. Pardo. Branch and bound for the Cutwidth Minimization Problem. *Computers & Op. Res.*, 40(1):137–149, 2013.
8. N. Mladenović and P. Hansen. Variable Neighborhood Search. *Computers & Operations Research*, 24(11):1097–1100, 1997.
9. J. J. Pantrigo, R. Martí, A. Duarte, and E. G. Pardo. Scatter search for the cutwidth minimization problem. *Annals of Operations Research*, 199:285–304, 2012. DOI:10.1007/s10479-011-0907-2.
10. E. G. Pardo, N. Mladenovic, J. J. Pantrigo, and A. Duarte. A variable neighbourhood search approach to the cutwidth minimization problem. *Electronic Notes in Discrete Mathematics*, 39(0):67 – 74, 2012.
11. E. Piñana, I. Plana, V. Campos, and R. Martí. GRASP and Path Relinking for the Matrix Bandwidth Minimization. *European Journal of Operational Research*, 153(1):200 – 210, 2004.
12. M. G. C. Resende, R. Martí, M. Gallego, and A. Duarte. Grasp and path relinking for the max-min diversity problem. *Computers & Op. Res.*, 37:498–508, 2010.