

Búsqueda dispersa aplicada al problema de la diversidad máxima

Micael Gallego¹, Abraham Duarte², Manuel Laguna³, Rafael Martí⁴

Resumen— El problema de la diversidad máxima presenta un reto a los métodos de resolución basados en la optimización heurística. En este trabajo hemos desarrollado un algoritmo que utiliza estructuras de memoria dentro de la metodología de la búsqueda dispersa. Los experimentos y las pruebas estadísticas muestran que el algoritmo propuesto mejora a los métodos publicados hasta la fecha.

Palabras clave— Metaheurísticas, Búsqueda Tabú, Búsqueda Dispersa, Problema de la Diversidad Máxima.

1. EL PROBLEMA DE LA DIVERSIDAD MÁXIMA

El problema de la diversidad máxima (*Maximum Diversity Problem*, MDP) consiste en seleccionar un subconjunto de m elementos de un conjunto mayor de n elementos de tal forma que la suma de las distancias entre los elementos seleccionados sea máxima. La noción de distancia entre elementos es específica de las aplicaciones. Como se menciona en [4], el problema de la diversidad máxima tiene aplicaciones en problemas sociales, preservación ecológica, control de la contaminación, diseño de productos, inversión de capital, gestión de la mano de obra, diseño de currículos e ingeniería genética. En la mayoría de las aplicaciones, se asume que cada elemento puede representarse por un conjunto de atributos. Si definimos s_{ik} como el valor del atributo k -ésimo del elemento i , donde $k=1, \dots, K$. Entonces la distancia entre los elementos i y j se puede definir como:

$$d_{ij} = \sqrt{\sum_{k=1}^K (s_{ik} - s_{jk})^2}$$

En este caso, d_{ij} simplemente es la distancia euclídea entre i y j . Los valores de las distancias se usan para formular el MDP como un problema cuadrático binario:

$$\begin{aligned} &\text{Maximizar} && \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} x_i x_j \\ &\text{Sujeto a} && \sum_{i=1}^n x_i = m \\ &&& x_i = \{0, 1\} \quad 1 \leq i \leq n \end{aligned}$$

En [4] se usa esta formulación para mostrar que el problema del clique (que es NP-completo) es reducible al MDP. También se sugiere la transformación del modelo cuadrático binario en un modelo de programación lineal entera de la siguiente forma:

$$\begin{aligned} &\text{Maximizar} && \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} y_{ij} \\ &\text{Sujeto a} && \sum_{i=1}^n x_i = m \\ &&& x_i + x_j - y_{ij} \leq 1 \quad 1 \leq i < j \leq n \\ &&& -x_i + y_{ij} \leq 0 \quad 1 \leq i < j \leq n \\ &&& -x_j + y_{ij} \leq 0 \quad 1 \leq i < j \leq n \\ &&& y_{ij} \geq 0 \quad 1 \leq i < j \leq n \\ &&& x_i = \{0, 1\} \quad 1 \leq i \leq n \end{aligned}$$

A. Revisión Bibliográfica

En la literatura se pueden encontrar diferentes trabajos que abordan el MDP. Tanto desde el punto de vista de la resolución exacta como desde el punto de vista de resolución aproximada. Kuo, Glover y Dhir presentan en [4] el problema, tanto formalmente como con un ejemplo. También proponen diferentes formulaciones en programación entera. En [2], Ghosh presenta un algoritmo multiarranque para la resolución aproximada del problema. Glover y Kuo presentan en [3] cuatro nuevas heurísticas. Silva, Ochi, y Martins presentan en [6] varios algoritmos GRASP para el MDP, tres métodos constructivos y dos métodos de búsqueda local. Por último, Duarte y Martí en [1] presentan métodos constructivos basados en GRASP y métodos de búsqueda local, algunos de ellos basados en la búsqueda tabú.

Los algoritmos presentados en los últimos trabajos se han usado como base para el diseño de los algoritmos basados en la metodología de la búsqueda dispersa que se presentan en este trabajo.

¹ Universidad Rey Juan Carlos. micael.gallego@urjc.es

² Universidad Rey Juan Carlos. abraham.duarte@urjc.es

³ University of Colorado at Boulder, USA. laguna@colorado.edu

⁴ Universidad de Valencia. rafael.marti@uv.es

TABLA 1
VARIANTES DE LA BÚSQUEDA DISPERSA

Procedimiento	Generación	Combinación	Mejora
Base	Selección aleatorio de m elementos de todos los n elementos del problema	Selección aleatoria de m elementos de la unión de los elementos de las soluciones que son combinadas	Búsqueda Local (<i>Local Search</i> , LS) de tipo “ <i>best improvement</i> ”
GRASP Híbrido	GRASP_D-2 , basado en la aleatorización de la heurísticas destructiva <i>D-2</i>	Aplicación de D-2 a la unión de los elementos de las soluciones que son combinadas	Búsqueda Local Mejorada (<i>Improved Local Search</i> , I_LS) de tipo “ <i>first improvement</i> ”
Búsqueda Tabú Híbrida	Tabú_D-2 , basado en la incorporación de estructuras de memoria a <i>D-2</i>	Aplicación de Tabu_D-2 a la unión de los elementos de las soluciones que son combinadas	Búsqueda Local Búsqueda Tabú (<i>Local Search Tabu Search</i> , LS_TS) con memoria a corto plazo

II. BÚSQUEDA DISPERSA

La búsqueda dispersa (*scatter search*, SS) es un procedimiento metaheurístico que explora un espacio de soluciones mediante la evolución de un conjunto de puntos de referencia. La búsqueda dispersa comienza con la aplicación de un método de generación de diversificación que resulta en una población de puntos de los cuales un conjunto es

1. Construcción de un conjunto P que consiste en $PopSize$ soluciones diversas mediante la aplicación del método de generación de diversificación.
2. Aplicación del método de mejora a todas las soluciones de P .
3. Construcción del conjunto de referencia inicial. El *RefSet* consiste en b soluciones de P , de las cuales el $q\%$ son elegidas por calidad (medida por la función objetivo) y $(100-q)\%$ son seleccionadas por su diversidad (medida por la distancia entre ellas y el resto de las soluciones que ya están en el conjunto de referencia).
4. Aplicación del método de generación de subconjuntos para crear una lista de todos los subconjuntos de soluciones de referencia que serán combinados.
5. Aplicación del método de combinación a todos los subconjuntos de soluciones de referencia generados en el paso previo.
6. Aplicación del método de mejora a todas las soluciones generadas por el método de combinación.
7. Actualización del *RefSet* si cualquier nueva solución candidata es mejor que cualquiera de las soluciones de referencia actuales.
8. Si una nueva solución ha sido incluida en el *RefSet* entonces ir al paso 4, en otro caso el procedimiento finaliza.

Fig. 1. Esquema de la búsqueda dispersa básica

seleccionado como conjunto de referencia inicial (*RefSet*). La evolución del conjunto de referencia se lleva a cabo por la aplicación de cuatro métodos adicionales: generación de subconjuntos, combinación, mejora y actualización. Una descripción detallada del método y una lista con algunas aplicaciones aparecen en el libro de Laguna y Marti [5]. En la Fig. 1 se muestra el esquema básico de la búsqueda dispersa.

Una extensión del diseño básico de búsqueda dispersa considera la reconstrucción del *RefSet*. Esto significa que en vez de finalizar cuando no se introducen nuevas soluciones al conjunto de referencia, el método de generación de diversificación es ejecutado de nuevo para generar una nueva población de soluciones (es decir, un nuevo P). El nuevo *RefSet* se construye con las mejores $q\%$ soluciones del actual *RefSet* y $(100-q)\%$ soluciones diversas de P . Después de que el *RefSet* se reconstruye el procedimiento sigue en el paso 4. La búsqueda finaliza después de un determinado tiempo de ejecución o un determinado número de reconstrucciones.

III. BÚSQUEDA DISPERSA APLICADA AL MDP

Para aplicar un procedimiento metaheurístico como la búsqueda dispersa a un problema concreto es necesario diseñar algunas partes de forma específica. En concreto, para el caso de la búsqueda dispersa, es necesario definir una distancia entre soluciones, de forma que pueda ser usada cuando se seleccionan las soluciones por diversidad. Además, se deben diseñar los métodos de generación de soluciones diversas, generación de subconjuntos, combinación de soluciones y mejora. Para determinar los métodos más adecuados se han implementado tres variantes de la búsqueda dispersa para el MDP. Las diferencias entre estas variantes se basan en los métodos usados para construir las soluciones diversas, combinar soluciones y

mejorarlas. En la Tabla 1 se indican los métodos usados en cada una de las configuraciones. El procedimiento denominado Base no utiliza ninguna información específica del problema. El procedimiento denominado GRASP Híbrido implementa varias estrategias específicas del MDP. El procedimiento Búsqueda Tabú Híbrida usa, además de información sobre el problema, estructuras de memoria que son típicas en las implementaciones de *tabu search*. En las siguientes subsecciones se define la distancia entre las soluciones y cada los métodos usados en cada una de las variantes.

B. Distancia entre soluciones

La distancia se usa para medir cómo de diversa es una solución con respecto a un conjunto de soluciones. Para el MDP, se define x_i^r como el valor de la i -ésima variable de la solución de referencia $r \in RefSet$. También se define x_i^t como el valor de la i -ésima variable de la solución candidata t . La distancia entre una solución candidata t y las soluciones en el *RefSet* se define como:

$$distancia(t, RefSet) = c_{\max} - \sum_{r=1}^b \sum_{i:x_i^r=1} x_i^t$$

La fórmula simplemente cuenta el número de veces que cada elemento seleccionado en la solución candidata t aparece en cualquiera de las soluciones del *RefSet* y resta este valor de la cuenta c_{\max} más grande. La máxima distancia se da cuando ningún elemento seleccionado en la solución candidata t aparece en alguna de las soluciones de referencia. Cuando se eligen soluciones para reconstruir el conjunto de referencia, seleccionamos la solución candidata que tiene la máxima distancia a las soluciones del *RefSet*. Ya que las soluciones son incorporadas una cada vez, el cálculo de las distancias tiene que ser actualizado antes de que la siguiente solución sea seleccionada.

C. Método de generación de soluciones diversas

En esta sección se describen los métodos usados en las diferentes configuraciones para la generación de soluciones diversas:

a) Selección aleatoria

Este método consiste en seleccionar m elementos del total de n elementos del problema. La selección aleatoria no utiliza ninguna información del MDP, por ejemplo, no usa la función objetivo para guiar el proceso constructivo. La única información usada es el número m de elementos necesaria para construir una solución factible.

b) GRASP_D-2

El procedimiento GRASP_D-2 desarrollado por Duarte y Martí [1], está basado en la aleatorización de la heurística destructiva desarrollada por Glover, Kuo y Dhir [3]. D-2 comienza con una solución no factible para la que $x_i=1$ para todo i . Es decir, todos los n elementos están seleccionados inicialmente. Para reducir el conjunto de elementos seleccionados hasta m , el procedimiento realiza $m-n$ pasos. En cada paso, el procedimiento deselecta el elemento i^* (es decir, $x_i=0$), donde i^* es tal que:

$$D(i^*) = \min_{i:x_i=1} (D(i)),$$

$$y \quad D(i) = \sum_j d_{ij} x_i x_j.$$

La aleatorización de D-2 que se emplea en GRASP_D-2 consiste en seleccionar i^* de una lista de candidatos reducida (*reduced candidate list*, RCL) formada por todos aquellos elementos i tales que $D(i) \leq (1 + \alpha)D(i^*)$. El valor de α se establece a 0.5 inicialmente y se disminuyen en 0.1 después de que un tiempo de ejecución predefinido se haya consumido sin obtener una solución de mejor calidad que las anteriores. En nuestro caso, se ha establecido este valor al 20% del tiempo de CPU total de experimentación.

c) Tabu_D-2

El método Tabu_D-2 fue presentado en [1]. Se basa en la incorporación de estructuras de memoria al método D-2. En cada paso del procedimiento, el elemento i^* que se deselecta es aquél que:

$$D(i^*) = \min_{i:x_i=1} (D(i) - freqPond + qualPond)$$

donde:

$$freqPond = \beta \left(range \right) \left(\frac{f(i)}{f_{\max}} \right),$$

$$qualPond = \delta \left(range \right) \left(\frac{q(i)}{q_{\max}} \right) y$$

$$range = \max_{i:x_i=1} (D(i)) - \min_{i:x_i=1} (D(i))$$

En este cálculo modificado de la distancia, $f(i)$ indica la frecuencia en la que el elemento i ha aparecido en soluciones previas y $q(i)$ es la calidad media (medida por la función objetivo) de las soluciones previas que incluían al elemento i . Tanto f_{\max} como q_{\max} son los máximos valores de f y q de todos los elementos. Los factores de ponderación β y δ son respectivamente establecidos a 0.1 y 0.0001 en nuestros experimentos.

D. Método de generación de subconjuntos

Se ha considerado el método más sencillo para la generación de subconjuntos. Este método consiste

en generar un subconjunto por cada pareja distinta de las soluciones a combinar.

E. Métodos de combinación

En esta sección se describen los métodos utilizados para la combinación de soluciones en cada una de las variantes de la búsqueda dispersa presentadas. Para ilustrar los diferentes métodos se usará un ejemplo con $n=10$ y $m=3$ con la matriz de distancias de la Tabla 2. Supongamos que durante la ejecución del algoritmo dos soluciones de referencia, r_1 y r_2 tienen que ser combinadas. Para facilitar la notación, presentamos cada solución como el conjunto de nodos seleccionados: $r_1 = \{1, 4, 6\}$ y $r_2 = \{1, 5, 10\}$. Los valores de la función objetivo asociados con estas soluciones son 9.09 ($2.65 + 2.38 + 3.61$) y 6.38 ($2 + 2.65 + 1.73$), respectivamente.

TABLA 2
MATRIZ DE DISTANCIAS DE EJEMPLO

	2	3	4	5	6	7	8	9	10
1	2	2	2,65	2	2,83	2,4	2,6	3	2,65
2		3	3,74	1,73	1	2,6	3,1	1,4	2
3			2,65	3,46	3,16	2,4	2,6	2,6	4,12
4				3,87	3,61	3,8	2	3,4	3,74
5					2	2	3,8	2,2	1,73
6						2,8	3,3	1,7	1,73
7							3,8	2,2	3,32
8								3,1	4
9									2,83

a) Selección aleatoria

Este método consiste en seleccionar m elementos de la unión de los elementos en ambas soluciones de referencia. En nuestro ejemplo, la unión de los elementos es $U = \{1, 4, 5, 6, 10\}$. La selección aleatoria consiste en seleccionar 3 elementos de U . Por ejemplo, la nueva solución candidata puede ser $t = \{1, 4, 10\}$ con la función objetivo 9.04 ($2.65 + 2.65 + 3.74$).

b) Selección D-2

Este método consiste en la aplicación de la heurística destructiva $D-2$ a la unión de los elementos de las soluciones que son combinadas. El método comienza con la selección de todos los elementos en la unión y deseleccionada un elemento cada vez hasta que solo quedan m elementos seleccionados. El elemento i que es deseleccionado a cada paso es aquel que tiene el valor $D(i)$ mínimo. En nuestro ejemplo, la unión consiste de 5 elementos y por tanto el método realiza dos pasos únicamente. En el primer paso, los valores D son: $D(1) = d(1,4) + d(1,5) + d(1,6) + d(1,10) = 2.65 + 2 + 2.83 + 2.65 = 10.13$, $D(4) = 13.87$, $D(5) = 13.22$, $D(6) = 10.17$ y $D(10) = 9.85$.

El valor mínimo D corresponde al elemento 10 y por tanto este elemento es deseleccionado. La unión actualizada es $U = \{1, 4, 5, 6\}$. Para el siguiente paso, los valores de D correspondientes son 7.48, 10.13, 7.87 y 8.44. Por tanto, el elemento 1 es deseleccionado y la solución candidata que resulta de la aplicación de este método de combinación es $t = \{4, 5, 6\}$ con un valor de la función objetivo igual a 9.48.

c) Selección Tabu D2

Consiste en aplicar el procedimiento Tabu D-2 a la unión de los elementos de las soluciones de referencia que tienen que ser combinadas. Este método utiliza información sobre las soluciones generadas en el pasado así como información asociada con esas soluciones combinadas.

Para ilustrar el funcionamiento del procedimiento se asume que después de un número de iteraciones, la información de la Tabla 3 está disponible.

TABLA 3
INFORMACIÓN SOBRE SOLUCIONES ANTERIORES

Elemento	Frecuencia	Calidad
1	13	7.73
2	19	7.25
3	17	8.69
4	8	9.37
5	45	6.64
6	17	7.71
7	16	8.42
8	16	9.09
9	19	7.44
10	13	8.50

Usamos la Tabla 3 para calcular los valores D modificados tal y como se describe anteriormente para el método Tabu D-2. $D(1) = 10.13 - (0.02)(13) + (0.000043)(7.73) = 9.87$, $D(4) = 13.71$, $D(5) = 12.32$, $D(6) = 9.83$ y $D(10) = 9.59$.

El mínimo valor D corresponde al elemento 10 y por tanto, este elemento es deseleccionado. La unión actualizada es $U = \{1, 4, 5, 6\}$. Para el siguiente paso, los valores correspondientes son 7.30, 10, 7.24 y 8.20. Por tanto, el elemento 5 es deseleccionado y la solución candidata que resulta de la aplicación de este método de combinación es $t = \{1, 4, 6\}$ con un valor de la función objetivo igual a 9.09.

F. Métodos de mejora

En esta sección se describen los métodos utilizados para la mejora de soluciones en cada una de las variantes de la búsqueda dispersa presentadas.

a) Búsqueda Local (LS)

El método de búsqueda local (*Local Search*, LS) [2] recorre el conjunto de elementos seleccionados buscando el mejor intercambio para reemplazar un

elemento seleccionado con un elemento no seleccionado. El método realiza movimientos siempre que el valor de la función objetivo aumente. Finaliza su ejecución cuando no se encuentra ningún intercambio de elementos que mejore. Este método de mejora se clasifica en los métodos que aplican el movimiento que más mejora, es decir, el mejor movimiento (*best improvement*).

b) Búsqueda Local Mejorada (I_LS)

El método de la búsqueda local mejorada (*Improved Local Search, I_LS*) [1] selecciona el elemento i^* ($x_{i^*}=1$) que proporciona la menor contribución al valor de la función objetivo de la actual solución. A continuación, busca un elemento j ($x_j=0$) para ser intercambiado con i^* . El primer elemento j que resulte en un movimiento de mejora se selecciona y el intercambio se realiza sin examinar el resto de elementos sin seleccionar. Si no se encuentra ningún movimiento de mejora que intercambie j por i^* , se examina el elemento con la siguiente menor contribución. Este proceso continúa hasta que no se encuentre ningún movimiento de mejora.

c) Búsqueda Local Búsqueda Tabú (LS_TS)

El método de mejora empleado en la configuración Búsqueda Tabú Híbrida se basa en el método de búsqueda tabú (*Local Search Tabu Search, LS_TS*) [1]. Éste implementa un método de búsqueda tabú a corto plazo basado en intercambios. Una iteración de este método comienza con una selección del elemento i ($x_i=1$) con el menor valor de $D(i)$. La lista de elementos no seleccionados es revisada y se selecciona el primer movimiento de intercambio de los elementos i y j ($x_j=0$) que mejore la función objetivo. Si no se ha encontrado ningún movimiento que mejore, entonces se elige el movimiento que menos empeora. El intercambio elegido se realiza y ambos elementos que participan en el intercambio se clasifican como tabú-activos durante un número de iteraciones (conocido como permanencia tabú o *tabu tenure*). A los elementos tabú-activos no se les permite participar en intercambios. El método LS_TS finaliza después de un número consecutivo de iteraciones en los que la solución no ha sido mejorada.

Hemos modificado el método LS_TS al añadirlo a nuestro *framework* de búsqueda dispersa. La modificación consiste en el uso de un *tabu tenure* asimétrico en el que los elementos añadidos a la solución tienen un *tabu tenure* más pequeño que el asignado a aquellos elementos que han sido borrados de la solución. También, el *tabu tenure* y el máximo número de iteraciones son dependientes del número de elementos del problema. De acuerdo a la experimentación presente en [1], el *tabu tenure* de los elementos seleccionados se establece en $0.28m$ iteraciones, mientras que el *tabu tenure* para los elementos no seleccionados es de $0.028(n-m)$. El número máximo de iteraciones sin mejora se establece en $0.1(n-m)$.

IV. ELECCIÓN DEL PROCEDIMIENTO DE BÚSQUEDA DISPERSA

Para determinar la mejor configuración de nuestro algoritmo de búsqueda dispersa se han realizado varios experimentos preliminares. Estos experimentos se han realizado usando dos tipos de instancias de problemas:

- Type I: 10 soluciones con $n=500$, $m=50$ y la matriz de distancias con números reales distribuidos uniformemente entre 0 y 100.
- Type II: 10 soluciones con $n=500$, $m=50$ y la matriz de distancias con números reales distribuidos uniformemente entre 0 y 10.

En los experimentos se calcula la desviación media en porcentaje ($mDev$) de la mejor solución obtenida en cada configuración con respecto a la mejor solución obtenida en esa experimentación. También se muestra el número de soluciones ($nMej$) que han resultado ser la mejor para una instancia del problema.

A. Filtro en el método de mejora

Durante las etapas iniciales del diseño del algoritmo se ha detectado que los métodos de diversificación y combinación obtienen la misma solución más de una vez. Para evitar que se aplique el método de mejora a una misma solución más de una vez, se ha utilizado una estructura de datos de tipo mapa *hash*. Esta estructura de datos almacena todas las soluciones y su correspondiente mejora.

TABLA 4
RESUMEN DE LOS RESULTADOS DEL EXPERIMENTO PARA LA ELECCIÓN DEL PROCEDIMIENTO

Procedimiento	Método de Mejora	Desviación Media ($mDev$)	Número de Mejores ($nMej$)
Base	No	13.7%	0
	Si	0.68%	0
GRASP Híbrido	No	1.16%	0
	Si	0.18%	2
Búsqueda Tabú Híbrida	No	1.07%	0
	Si	0.00%	20

Este método es capaz de filtrar 0.5% de soluciones en el procedimiento GRASP Híbrido y un 82% de las soluciones en el procedimiento Búsqueda Tabú Híbrida. Antes de aplicar el procedimiento de mejora, se utiliza la función *hash* para buscar si esa solución existe en la estructura de datos. En ese caso, se utiliza la solución previamente mejorada que también se almacena en la estructura de datos.

B. Elección del procedimiento

El objetivo de este experimento es seleccionar la mejor variante de la búsqueda dispersa de las descritas en la Tabla 1. Establecemos un tiempo de 3 minutos de ejecución y ejecutamos cada procedimiento con y sin método de mejora. Los resultados se resumen en la Tabla 4.

Los resultados de la Tabla 4 indican la ventaja de usar un método de mejora en nuestro diseño. En términos de desviación media desde las mejoras soluciones, el método de mejora tiene el impacto más grande en el caso del diseño Base. También, el método de mejora ofrece una diferencia importante en el número de mejores soluciones encontradas en la Búsqueda Tabú Híbrida. En general, se concluye que el procedimiento Búsqueda Tabú Híbrida es la mejor variante de la búsqueda dispersa. Para el resto del trabajo esta será la variante elegida, incluyendo el método de mejora.

C. Configuración del Conjunto de Referencia (RefSet)

El objetivo de este experimento es comprobar la aplicación selectiva del método de mejora. Nuestro diseño de búsqueda dispersa está diseñado de tal forma que a cada solución generada por el método constructivo o por el método de combinación se le aplica el método de mejora. Desde que la ejecución del método de mejora es costosa desde el punto de vista computacional, aplicarle sobre cada solución puede impedir que se visiten nuevas soluciones durante el tiempo de ejecución. Por tanto, en este experimento se comprueba la idea de aplicar el método de mejora selectivamente a un subconjunto de las soluciones que son visitadas a lo largo de la búsqueda. Concretamente, siendo b el tamaño del *RefSet* en vez de mejorar todas las soluciones iniciales de P se seleccionan las mejores $(q\%)b$ soluciones de P y únicamente a estas soluciones se les aplica el método de mejora. Tras su aplicación, se incluyen en el *RefSet*. De forma similar, después del método de combinación, el método de mejora no se aplica a todas las soluciones. En vez de esto, las mejores $(q\%)b$ soluciones son seleccionadas y mejoradas. La aplicación selectiva del método de mejora puede resultar en una trayectoria que no visita soluciones de calidad que podrían haber sido encontradas si se hubiese aplicado el método de mejora a una solución relativamente inferior.

Debido a esto se ha diseñado un experimento preliminar con el objetivo de identificar la configuración de los parámetros (es decir, los valores de b y q) que podrían ser efectivos bajo la aplicación selectiva del método de mejora. Los resultados del experimento revelan que el mejor rendimiento medio se obtiene cuando se aplica el método de mejora a todas las soluciones con $q = 30$. Asimismo, la mejor desviación media para el procedimiento de mejora selectiva se obtiene con $q = 70$. Cuando se considera la desviación media y el número de mejores soluciones, la variante "Mejora Completa" supera a la "Mejora Selectiva".

Dada la efectividad de la variante "Mejora Completa" con $q = 30$, se ha realizado un experimento adicional donde se ha variado los valores de b desde 4 a 160. El experimento ha mostrado que el mejor resultado se encuentra cuando $b = 12$. Por tanto se utilizar los valores $b=12$ y $q=30$ en las siguientes experimentaciones.

V. EXPERIMENTOS COMPUTACIONALES

Esta sección describe los experimentos computacionales que se han realizado para comparar el procedimiento propuesto frente a los métodos del estado del arte que resuelven el problema de la diversidad máxima. El método de generación de soluciones diversas, el método de mejora y el método de combinación son los correspondientes al procedimiento Búsqueda Tabú Híbrida de la Tabla 1. El método de mejora se aplica a todas las soluciones con $b=12$ y $q=30$. Finalmente, el método de generación de subconjuntos se limita a generar subconjuntos de tamaño 2. Este procedimiento de búsqueda dispersa (etiquetado como SS en las siguientes tablas) se compara con los siguientes algoritmos:

- KLD [6] con Búsqueda Local (LS) [2]
- KLDv2 [6] con Búsqueda Local (LS) [2]
- Tabu_D-2 con con Búsqueda Local Búsqueda Tabú (LS_TS) [1]

Para esta comparación, se han usado los mismos cuatro conjuntos de datos empleados en [1]:

- SOM: Este conjunto de datos consiste en 20 matrices con números aleatorios entre 0 y 9 generados desde una distribución uniforme entera. Los tamaños de los problemas son para $n = 100, m = 10, 20, 30$ y 40 ; para $n = 200, m = 20, 40, 60$ y 80 ; para $n = 300, m = 30, 60, 90$ y 110 ; para $n = 400, m = 40, 80, 120$ y 160 ; y para $n = 500, m = 50, 100, 150$ y 200 . Estas instancias fueron generadas por Silva, Ochi y Martins en [6].
- GKD: Este conjunto de datos consiste en 20 matrices cuyos valores fueron calculados como la distancia euclídea entre puntos generados aleatoriamente con coordenadas en el rango entre 0 y 10. El número de coordenadas para cada punto se genera aleatoriamente entre 2 y

TABLA 5
COMPARACIÓN DE LA DESVIACIÓN MEDIA A LA MEJOR SOLUCIÓN Y EL NÚMERO DE MEJORES EN DIFERENTES INSTANTES DE TIEMPO

Instancias	Tiempo	KLD		KLDv2		Tabu_D-2		SS	
		mDesv	nMej	mDesv	nMej	mDesv	nMej	mDesv	nMej
SOM	30 seg.	1.056%	6	1.463%	6	0.138%	4	0.002%	17
	3 min.	0.178%	7	0.187%	9	0.095%	6	0.000%	20
GKD	30 seg.	0.000%	20	0.000%	19	0.000%	20	0.000%	20
	3 min.	0.000%	20	0.000%	20	0.000%	20	0.000%	20
Type II	30 seg.	0.857%	1	1.083%	0	0.245%	0	0.010%	15
	3 min.	0.525%	1	0.607%	0	0.203%	0	0.000%	20
Type I	30 seg.	9.807%	0	100%	0	0.453%	0	0.453%	0
	3 min.	9.807%	0	9.828%	0	0.331%	0	0.331%	0
	30 min	1.018%	0	0.923%	0	0.233%	0	0.000%	20

21. Glover, Kuo y Dhir [3] desarrollaron este generador de datos y construyeron instancias con $n = 30$. Para esta experimentación se han generado instancias con $n = 500$ y $m = 50$.

- Type I: Se han generado 20 instancias de Type I, como las usadas en los experimentos preliminares, con $n = 2000$ y $m = 200$.
- Type II: Se han generado 20 instancias de Type II, como las usadas en los experimentos preliminares, con $n = 500$ y $m = 50$.

En estos experimentos, se ha observado la calidad de la solución obtenida por cada método después de 30 segundos y después de 3 minutos de ejecución. También se ha incluido una ejecución de 30 minutos para los problemas Type I. Todos los experimentos se han ejecutado en un PC con procesador Intel Pentium 4 a 3 Ghz con 3 GB de RAM. Se han implementado todos los procedimientos en Java y se han ejecutado en el entorno de ejecución de Java (*Java Runtime Environment, JRE*) versión 1.5.

La Tabla 5 muestra un resumen de los

resultados. En ella se observa el mérito del procedimiento propuesto. Nuestra implementación de la búsqueda dispersa consistentemente produce las mejores soluciones con desviaciones que en algunos casos son ordenes de magnitud más pequeñas que los métodos con los que se compara. Las instancias del conjunto GKD no permiten diferenciar el rendimiento de los métodos que estamos comparando. Estas instancias o son muy sencillas de resolver y todos los métodos son capaces de encontrar la solución óptima en un periodo muy corto de tiempo o las instancias son difíciles y todos los métodos son atraídos por un óptimo local. Lo más probable es que las instancias sean muy sencillas. La Fig. 2 muestra la evolución típica para los métodos que comparamos. Esta ejecución corresponde al conjunto de datos SOM con un límite de 3 minutos.

Se ha aplicado una prueba estadística a los datos usados para generar la Tabla 5. Los resultados de la ejecución en los 30 segundos no se usaron debido a que KLDv2 no es capaz de obtener soluciones para

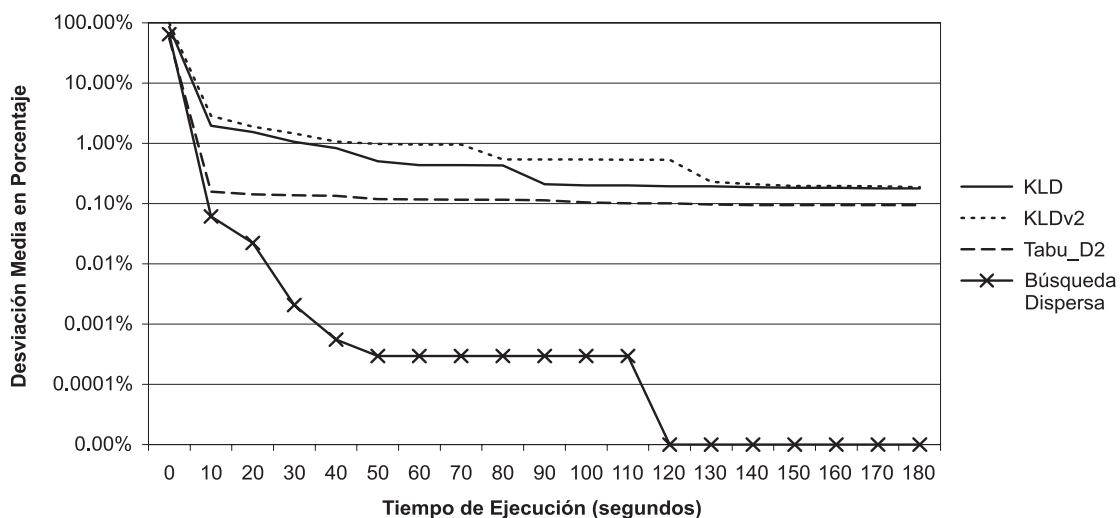


Fig. 2. Evolución de la desviación media en la ejecución de los algoritmos durante 3 minutos con el conjunto de datos SOM

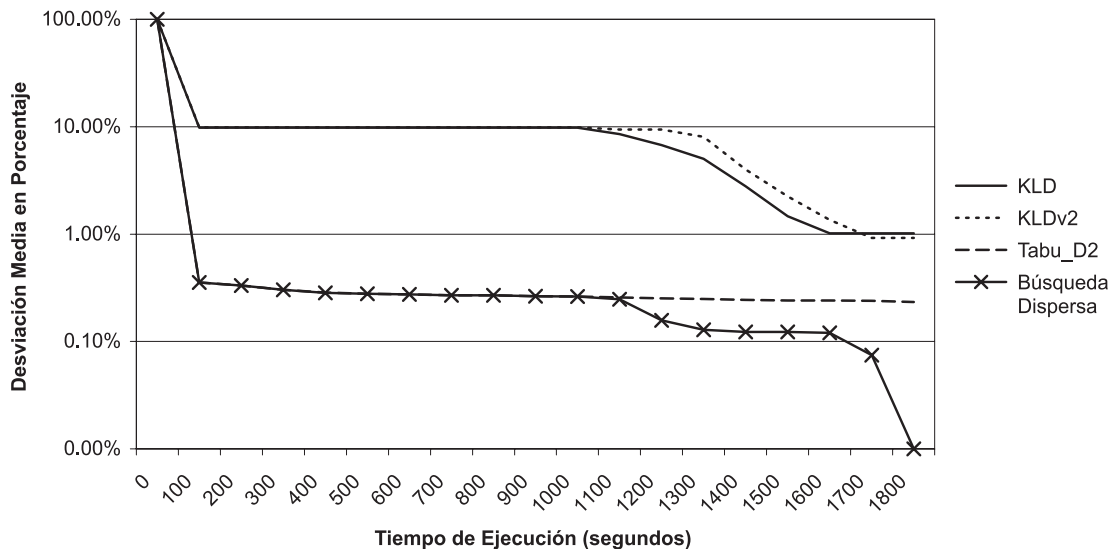


Fig. 3. Evolución de la desviación media en la ejecución de los algoritmos durante 30 minutos con el conjunto de datos Type I

las instancias Type I en ese tiempo (ver la desviación del 100% en la Tabla 5). Se ha aplicado el test de Friedman a las parejas de ejemplos de las mejores soluciones obtenidas por cada método. El nivel de significación resultante de 0.000 indica claramente que hay diferencias estadísticamente significativas entre los cuatro métodos comparados. Concretamente, los valores producidos por este test son 1.98, 1.86, 2.82 y 3.36 para KLD, KLDv2, Tabu_D-2 y SS, respectivamente. Esto indica que entre los procedimientos que se han probado, la búsqueda dispersa es la mejor obteniendo las soluciones de mejor calidad, seguido de Tabu-D-2, KLD y finalmente KLDv2.

Es interesante remarcar que para las instancias grandes Type I, no hay una diferencia relevante en el rendimiento del algoritmo Tabu_D2 y la búsqueda dispersa cuando el procedimiento termina a los 3 minutos. Esto se debe a que el tiempo que emplea la búsqueda dispersa generando la población inicial de soluciones es mayor de 3 minutos. Por tanto, en ese tiempo no se alcanza la fase en las que las soluciones de referencia son combinadas para generar otras soluciones. Las diferencias entre estos dos métodos solo se detectan después de 20 minutos de búsqueda, como se muestra en la Fig. 3.

VI. CONCLUSIONES

En este trabajo se describe el desarrollo e implementación de un procedimiento de búsqueda dispersa para resolver el problema de la diversidad máxima. Se ha llegado al diseño final realizando una serie de experimentos preliminares. Este diseño final se ha comparado con los métodos más eficientes publicados hasta la fecha y el resultado de los experimentos muestra el mérito del procedimiento propuesto. Hasta donde se conoce, el trabajo presentado es el primero en comprobar el comportamiento de varios procedimientos híbridos

dentro de la metodología de la búsqueda dispersa. Se considera que la mejora de rendimiento que se ha obtenido por el uso de mecanismos simples de memoria (algunos basados en lo reciente y otros basados en la frecuencia) en el procedimiento de la búsqueda dispersa puede constituir la base para futuras implementaciones.

AGRADECIMIENTOS

Este trabajo está financiado parcialmente por el proyecto TIN2006-02696 del Ministerio de Educación y Ciencia Español.

REFERENCIAS

- [1] A. Duarte and R. Martí. Tabu Search and GRASP for the Maximum Diversity Problem. to appear in the European Journal of Operational Research.
- [2] J. B. Ghosh. Computational Aspects of the Maximum Diversity Problem. *Operations Research Letters*, 19:175–181, 1996.
- [3] F. Glover, C. C. Kuo, and K. S. Dhir. Heuristic Algorithms for the Maximum Diversity Problem. *Journal of Information and Optimization Sciences*, 19(1):109–132, 1998.
- [4] C.C. Kuo, F. Glover, and K. S. Dhir. Analyzing and Modeling the Maximum Diversity Problem by Zero-One programming. *Decision Sciences*, 24(6):1171–1185, 1993.
- [5] M. Laguna and R. Martí. *Scatter Search methodology and implementations in C*. Kluwer Academic Publishers, 2003.
- [6] G. C. Silva, L. S. Ochi, and S. L. Martins. Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem. In *Experimental and Efficient Algorithms*, volume 3059 of *Lecture Notes in Computer Science*, pages 498–512. Springer Berlin / Heidelberg, 2004.