Heurísticos para el Problema de la Diversidad MaxMin

Micael Gallego[1], Mauricio Resende[2], Rafael Martí[3], Abraham Duarte[4]

Resumen— **El Problema de la Diversidad MaxMin (*MaxMin Diversity Problem*, MMDP) consiste en seleccionar un determinado número de elementos de un conjunto de forma que la diversidad entre los elementos seleccionados sea máxima. Este problema es NP-difícil y puede formularse como un problema lineal entero. Desde la década de los 80 se han desarrollado varios métodos para su resolución y se ha aplicado a diferentes campos. En este trabajo proponemos un método heurístico para la resolución aproximada de este problema. Exploraremos diferentes formas de hibridar GRASP y reencadenamiento de trayectorias, incluyendo la variente novedosa denominada GRASP con reencadenamiento de trayectorias evolutivo. Los resultados empíricos indican que las implementaciones híbridas propuestas obtienen mejores resultados que las metaheurísticas previas, tales como búsqueda tabú o recocido simulado.**

Palabras clave— **Problema de la Diversidad MaxMin, Metaheuristicas, GRASP, Reencadenamiento de Trayectorias, Reencadenamiento de Trayectorias Evolutivo**

I. Introducción

El Problema de la Diversidad Máxima consiste en seleccionar un determinado número de elementos de un conjunto de forma que la diversidad entre los elementos seleccionados sea máxima. Como se indica en [10], dependiendo de la forma de medir la diversidad de los elementos seleccionados este problema tiene diversas variantes, entre ellas destacan la variante MaxSum y la variante MaxMin. En los últimos años ambas variantes han recibido mucha atención. La primera variante, también conocida como el *Problema de la Diversidad Máxima* (*Maximum Diversity Problem*, MDP) mide la diversidad de los elementos seleccionados como la suma de las distancias entre ellos. Esta variante ha sido estudiada en diversos trabajos [8][17][2]. En la variante MaxMin, se considera que la diversidad de los elementos seleccionados es la mínima distancia entre ellos. Para esta variante, conocida como el *Problema de la Diversidad MaxMin* (*MaxMin Diversity Problem*, MMDP), se han propuesto métodos exactos [3] y heurísticos, como el recocido simulado [9], búsqueda tabú [9] y GRASP [6]. Puesto que los problemas que maximizan un mínimo presentan un espacio de soluciones plano (*flat landscape*), estos artículos concuerdan en que el MMDP presenta un reto para los métodos basados en la optimización heurística.

En términos formales, el MMDP consiste en seleccionar un subconjunto M de m elementos ($|M| = m$) de un conjunto N de n elementos ($m < n$) de forma que la menor distancia entre los elementos seleccionados sea máxima. La definición de distancia entre los elementos depende de las aplicaciones específicas. Como se menciona en [10][8], el MMDP tiene aplicaciones en fitomejoramiento, problemas sociales y preservación ecológica. En la mayoría de estas aplicaciones se asume que cada elemento puede representarse por un conjunto de atributos. Sea s_{ik} el estado o valor del k-ésimo atributo de elemento i, donde $k = 1,\ldots,K$. 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} es simplemente la distancia Euclídea entre i y j. Las distancias se usan para formular el MMDP como un problema cuadrático binario, donde por cada $i = 1,\ldots,n$, la variable x_i toma el valor 1 si el elemento i está seleccionado ó 0 en otro caso:

$$\text{(MMDP) max } z_{MM}(x) = \min_{i<j} d_{ij}x_ix_j$$

$$\text{subject to } \sum_{i=1}^{n} x_i = m$$

$$x_i = \{0,1\}, \quad i = 1,\ldots,n.$$

Erkut [3] y Ghosh [6] mostraron independientemente que el MMDP es NP-difícil.

En la Sección II, describimos el trabajo previo. En este artículo, exploramos la hibridación de las metodologías GRASP, reencadenamiento de trayectorias, y reencadenamiento de trayectorias evolutivo para encontrar soluciones óptimas o cercanas al óptimo para el MMDP. En las Secciones III y IV introducimos nuestros algoritmos. Los experimentos computacionales se describen en la Sección V y las conclusiones se exponen en la Sección VI.

II. Métodos Previos

Erkut [3] propuso un algoritmo de ramificación y acotación y un método heurístico para el MMDP. El método heurístico consiste en una fase de construcción seguida de una fase de búsqueda local. La construcción se inicia con la solución infactible en la que todos los n elementos están seleccionados. Para reducir el conjunto de elementos seleccionados a m, el procedimiento realiza $n \cdot m$ pasos. En cada paso, elimina el elemento i^* con la distancia asociada más corta. Hay que tener en cuenta que para la distancia más corta d_{ij} hay al menos dos elementos con esta distancia asociada. El método selecciona aleatoriamente uno de ellos. La construcción se puede repetir, obteniendo una solución diferente cada vez. El método de búsqueda local recorre el conjunto de elementos seleccionados buscando el mejor intercambio para reemplazar un elemento seleccionado con uno no seleccionado.

[1]Departamento de Ciencias de la Computación. Universidad Rey Juan Carlos. E-mail: micael.gallego@urjc.es

[2]Departamento de Investigación en Algoritmos y Optimización. Laboratorios de Investigación de AT&T. E-mail: mgcr@research.att.com

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

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

El método realiza movimientos siempre que se incremente el valor de la función objetivo y se detiene cuando no encuentra ningún intercambio que lo haga.

Kincaid [9] propuso dos heurísticas para el MMDP basadas en intercambios: un recocido simulado (*Simulating Annealing*, SA) y una búsqueda tabú (*Tabu Search*, TS). En una determinada iteración, el método SA genera un movimiento aleatorio (un intercambio entre un elemento seleccionado y uno no seleccionado). Si es un movimiento de mejora, se realiza automáticamente; si no, todavía puede realizarse con una determinada probabilidad. La probabilidad de aceptación de un movimiento que empeora la solución decrece conforme se realizan iteraciones. El procedimiento TS también realiza movimientos de intercambio. En cada iteración, se considera un número determinado de movimientos y el método realiza el mejor de los admitidos. En este caso, "admitido" se refiere al estado tabú del movimiento. Cuando un movimiento se realiza y un elemento seleccionado i se intercambia con un elemento no seleccionado j, el par no ordenado (i, j) se registra en una lista tabú y se etiqueta como tabú durante ciertas iteraciones. Un movimiento seleccionado es "admisible" si no está etiquetado como tabú, o si su aplicación hace que la solución sea mejor que la mejor solución conocida (criterio de aspiración). Después de un número determinado de iteraciones, la búsqueda se reinicia desde una nueva solución inicial por motivos de diversificación. Tanto SA como TS inician el algoritmo con una solución generada aleatoriamente.

Ghosh [6] propuso un método constructivo y un método de búsqueda local. Dado un conjunto N con n elementos, el método constructivo realiza m pasos como sigue. Sea M_{k-1} una solución parcial con $k-1$ elementos ($1 \leq k \leq m$). Para cada elemento $i \in N \setminus M_{k-1}$, sea $\Delta z_{MM(i)}$ la contribución de i al valor de la solución. Sea $\Delta z_L(i)$ y $\Delta z_U(i)$, respectivamente, una cota inferior y una cota superior de $\Delta z_{MM}(i)$. $\Delta z_L(i)$ se calcula como el mínimo entre el valor de la actual solución y la mínima distancia entre i y los otros elementos en N. $\Delta z_U(i)$ se calcula ordenando las distancias entre i y los elementos en $N \setminus M_{k-1}$ y calculando el menor entre los mayores $m-k$ elementos. Por tanto, $\Delta z'(i) = (1-u)\Delta z_L(i) + u\Delta z_U(i)$ es una estimación de $\Delta z_{MM}(i)$ (donde u es un número aleatorio generado con una distribución uniforme U(0,1)). El elemento i^* con el mayor valor de la estimación es seleccionado para ser incluido en la solución parcial:

$$M_k = M_{k-1} \cup \{i^*\}, \quad \Delta x'(i^*) = \max_{i \in N \setminus M_{k-1}} \{\Delta z'(i)\}.$$

Comenzando con un elemento seleccionado aleatoriamente, este proceso se repite hasta se obtiene finalmente M_m como la salida del contructivo ($|M_m| = m$). La búsqueda local es similar a la introducida por Erkut [3] y comienza después de la fase de construcción con el objeto de mejorar la solución a través de una búsqueda local. El método realiza diez iteraciones (construcción seguida de mejora) y la mejor solución se devuelve como salida.

III. GRASP

La metodología GRASP se desarrolló a finales de la década de los 80 [5]. Cada iteración GRASP consiste en la construcción de una solución y la aplicación de una búsqueda local sobre ella. La fase de construcción es iterativa, aleatoria, voraz y adaptativa. En esta sección se describe la adaptación propuesta de GRASP al MMDP.

A. Procedimientos Constructivos

De los algoritmos previos revisados en la Sección II, se pueden extraer dos procedimientos constructivos. ErkC, el método propuesto por Erkut [3], basado en eliminar elementos del conjunto de los seleccionados, y GhoC, el método propuesto por Ghosh [6], basado en una estimación de la contribución de los elementos. En esta sección, se proponen dos nuevos métodos constructivos, GRC y GRC2, que siguen la metodología GRASP.

Dado un conjunto N con n elementos, el procedimiento constructivo GRC realiza m pasos para producir una solución con m elementos. El conjunto *Sel* contiene los elementos de la solución parcial que se va construyendo. El conjunto denominado *lista de candidatos* (*Candidate List*, *CL*) contiene los elementos que no están en la solución parcial ($CL = N \setminus Sel$). En cada paso, GRC selecciona un elemento candidato $i^* \in CL$ con una distancia relativamente grande a los elementos de la solución parcial *Sel*. En concreto, primero calcula d_j como la mínima distancia entre el elemento j y los elementos seleccionados. Posteriormente, se construye una *lista de candidatos restringida* (*Restricted Candidate List*, *RCL*) con todos los elementos candidatos j ($j \in CL$) cuya distancia d_j sea igual o mayor a una fracción α ($0 \leq \alpha \leq 1$) de la máxima distancia $d^* = \max\{d_j \mid j \in CL\}$. Por último, GRC selecciona aleatoriamente un elemento de la *RCL*.

GRC implementa una construcción GRASP típica en la que, primero, cada elemento se evalúa de acuerdo a una función voraz para construir la *RCL* y, posteriormente, un elemento se selecciona aleatoriamente de esa *RCL*. También se considera GRC2, un procedimiento constructivo alternativo introducido en [15] como una construcción aleatoria y voraz. En GRC2 primero se seleccionan aleatoriamente los candidatos y luego se evalúa cada candidato de acuerdo a una función voraz que realiza la selección. GRC2 primero construye la lista de candidatos restringida *RCL2* con una fracción β ($0 \leq \beta \leq 1$) de los elementos en *CL* seleccionados aleatoriamente. Posteriormente, evalúa todos los elementos en *RCL2*, calculando d_j para todo $j \in RCL2$, y selecciona el mejor, es decir el elemento j^* tal que $d_{j^*} = \max\{d_j \mid j \in RCL2\}$.

En la Sección V de experimentos computacionales, se muestra cómo los parámetros de búsqueda α y β afectan a GRC y GRC2, respectivamente.

B. Métodos de Búsqueda Local

Erkut [3] y Ghosh [6] propusieron el método de búsqueda local BLS, basado en la estrategia *best-improvement*, en la que en cada iteración se recorre completamente la vecindad buscando el mejor intercambio (entre un elemento seleccionado y otro no seleccionado).

A continuación, se propone un nuevo método de búsqueda local, denominado FLS, basado en la estrategia *first-improvement* que aplica el primer movimiento que mejore la solución.

Dado un conjunto N de n elementos, y una solución Sel con m elementos, se calculan los valores $d_i = \min_{j \in Sel} d_{ij}$ y $d^* = \min_{j \in Sel} d_j$ donde d_i es la mínima distancia del elemento i a los elementos seleccionados (aquellos en Sel), y d^* es el valor de la función objetivo de la solución actual; es decir, $d^* = z_{MM}(Sel)$. Para mejorar una solución, es necesario eliminar de la solución (y por tanto reemplazar) los elementos i para los que $d_i = d^*$.

El método FLS recorre, en cada iteración, la lista de elementos en la solución ($i \in Sel$) con menor valor d_i; es decir, aquellos dónde $d_i = d^*$. Recorre la lista de elementos en orden lexicográfico, empezando con un elemento seleccionado aleatoriamente. Posteriormente, por cada elemento i con un d_i mínimo, FLS examina la lista de elementos no seleccionados ($j \in N \setminus Sel$) buscando el primer intercambio que mejore la solución. Los elementos no seleccionados también se examinan en orden lexicográfico, iniciando con un elementos seleccionado aleatoriamente. El método realiza el primer movimiento que mejore ($Sel \leftarrow (Sel \setminus \{i\}) \cup \{j\}$) y actualiza d_i para todos los elementos $i \in Sel$ así como el valor de la función objetivo d^*, finalizando la iteración actual. El algoritmo realiza iteraciones siempre que pueda realizar movimientos de mejora y finaliza cuando no es posible mejorar la solución. Como se describe posteriormente, la definición de "mejora" no está limitada a la función objetivo.

El ejemplo en la Tabla I con $n = 6$ y $m = 4$ ilustra el rendimiento del procedimiento de búsqueda local. En él se considera la solución $Sel = \{1,2,3,4\}$ con un valor de $d^* = 3$ y se realiza una iteración del método FLS.

TABLA I
EJEMPLO PARA ILUSTRAR LA BÚSQUEDA LOCAL.

	1	**2**	**3**	**4**	**5**	**6**
1	-	3	4	5	7	5
2	3	-	3	4	8	1
3	4	3	-	6	4	7
4	5	4	6	-	5	9
5	7	8	4	5	-	8
6	5	1	7	9	8	-

El valor d_i para cada elemento de la solución es $d_1 = 3$, $d_2 = 3$, $d_3 = 3$ y $d_4 = 4$. El método FLS selecciona un elemento cuyo valor de d_i sea mínimo, por ejemplo $i = 1$. A continuación se examina la lista de elementos no seleccionados buscando un movimiento de mejora. Se debe tener en cuenta que pese a eliminar el elemento 1, los elementos 2 y 3 permanecen en la solución y por tanto d^* será igual a $d_{23} = 3$, independientemente del elemento que se introduzca en la solución para reemplazar al elemento 1. Por tanto, estrictamente hablando, no se puede encontrar ningún intercambio que mejore la solución cuando se elimina el elemento 1. Pero por otro lado, en cierto sentido, la solución mejora cuando se elimina el elemento 1, ya que el número de elementos con d^* se decrementa y por tanto se puede decir que se está más cerca de obtener una mejor solución.

Esto es por lo que se considera una definición extendida de mejora para un determinado movimiento, en la que se incluye no sólo cuando el movimiento incrementa el valor de d^*, sino también cuando d^* se mantiene fijo y el número de elementos i con $d_i = d^*$ se reduce. En este ejemplo, cuando se reemplaza el elemento 1 con el elemento 5 obteniendo $Sel' = \{2,3,4,5\}$, se considera que es un movimiento de mejora porque $d^* = 3$ y d_i sólo concuerda con d^* en dos elementos (2 y 3), lo cual es mejor que la solución inicial Sel (en la que tres elementos tienen $d_i = d^* = 3$).

IV. REENCADENAMIENTO DE TRAYECTORIAS

El *reencadenamiento de trayectorias* (*Path Relinking*, PR) se sugirió como un enfoque para integrar las estrategias de intensificación y diversificación en el contexto de la búsqueda tabú [7]. Este enfoque genera nuevas soluciones explorando trayectorias que conectan soluciones de alta calidad, comenzando desde una de esas soluciones, denominada *solución inicial*, y generando una trayectoria en la vecindad hacia las otras soluciones, denominadas *soluciones guía*. Esto se consigue seleccionando movimientos que introducen atributos contenidos en las soluciones guía, e incorporándolos en una *solución intermedia* que se origina en la solución inicial.

En [11] se adaptaba PR en el contexto de GRASP como una forma de intensificación en un procedimiento denominado GRASP con PR. El reencadenamiento en este contexto consiste en encontrar una ruta entre una solución construida con GRASP y una determinada solución de élite. Por tanto, el concepto de reencadenamiento tiene una interpretación diferente en GRASP, ya que las soluciones creadas en dos iteraciones GRASP consecutivas no se enlazan por una secuencia de movimientos como ocurre en el caso de la búsqueda tabú. En esta sección se presentan diversas adaptaciones de GRASP con PR al MMDP.

Sean x e y dos soluciones del MMDP, interpretadas como los conjuntos de m elementos seleccionados Sel_x y Sel_y, respectivamente ($|Sel_x| = |Sel_y| = m$). El procedimiento de reencadenamiento de trayectorias PR(x,y) comienza con la primera solución x, y gradualmente la transforma en la segunda y intercambiando los elementos seleccionados en x por aquellos seleccionados en y. Los elementos seleccionados en ambas soluciones x e y, Sel_{xy}, se mantienen seleccionados en las soluciones intermedias generadas en la trayectoria entre ellas. Sea Sel_{x-y} el conjunto de elementos seleccionados en x y no seleccionados en y y, simétricamente, sea Sel_{y-x} el conjunto de elementos seleccionados en y y no seleccionados en x, es decir $Sel_{xy} = Sel_x \cap Sel_y$, $Sel_{x-y} = Sel_x \setminus Sel_{xy}$, $Sel_{y-x} = Sel_y \setminus Sel_{xy}$.

Sea $p_0(x,y) = x$ la solución inicial el la trayectoria P(x,y) desde x hacia y. Para obtener la solución $p_1(x,y)$ en esta trayectoria, se elimina un elemento $i \in Sel_{x-y}$ de $p_0(x,y)$, y se añade un elemento de $j \in Sel_{y-x}$, obtenien-

do de esta forma $Sel_{p_1(x,y)} \leftarrow (Sel_{p_0(x,y)} \setminus \{i\}) \cup \{j\}$.

En el método propuesto, la selección de los elementos i y j se realiza de forma voraz. Para obtener $p_{k+1}(x,y)$ desde $p_k(x,y)$, se evalúan todas las posibilidades para quitar $i \in Sel_{p_k(x,y)-y}$ y añadir $j \in Sel_{x-p_k(x,y)}$ y se realiza el mejor intercambio. De esta forma, se alcanza y desde x en $r = |Sel_{x-y}| = |Sel_{y-x}|$ pasos, es decir $p_r(x,y) = y$. La salida del algoritmo PR es la mejor solución, diferente de x e y, encontrada en la trayectoria P(x,y) (entre $p_1(x,y), p_2(x,y), \ldots, p_{r-1}(x,y)$).

El algoritmo PR opera en un conjunto de soluciones, denominado *conjunto de élite* (*Elite Set*, ES), construido con la aplicación de un método previo. En el método GRASP con PR (mostrado en la Figura 1), se aplica GRASP para construir el conjunto de élite. Si sólo se considerase un criterio de calidad para crear el conjunto de élite, se podría crear simplemente con las mejores soluciones creadas con GRASP. No obstante, estudios previos [15] han demostrado empíricamente que una aplicación de PR a un par de soluciones muy similares no obtiene buenos resultados. Por tanto, para construir el *ES* se considera tanto la calidad como la diversidad.

Inicialmente el conjunto *ES* está vacío, se ejecutan $b = |ES|$ iteraciones de GRASP para poblarlo con las soluciones obtenidas. Se ordenan las soluciones de *ES* desde la mejor (x^1) a la peor (x^b). Posteriormente, en las siguientes iteraciones GRASP, se comprueba si una solución generada (construida y mejorada) x' puede incorporarse al *ES*. Especificamente, si x' es mejor que la mejor x^1, se inserta en el conjunto. Además, si es mejor que la peor x^b y es suficientemente diferente de las soluciones del conjunto de élite ($d(x', ES) \geq dth$), también se inserta en el *ES*. El parámetro *dth* es un umbral de distancia que refleja el término "suficientemente diferente" y es ajustado empíricamente (ver Sección V). Para mantener el tamaño del *ES* constante e igual a b, cuando se añade una solución a este conjunto, se elimina otra. Para conservar la calidad y la diversidad, se elimina la solución más parecida a x' de entre aquellas con peor valor. La Figura 1 muestra el pseudo-código del algoritmo GRASP con PR.

El diseño de la Figura 1 se denomina GRASP con PR estático debido a que primero se aplica GRASP para construir el conjunto de élite *ES* y posteriormente se aplica el PR para generar soluciones entre todos los pares de soluciones en *ES*. Dadas dos soluciones en *ES*, x e y, se aplica el reencadenamiento de trayectorias en ambas direcciones, es decir PR(x,y) desde x hasta y y PR(y,x) desde y hasta x. A la mejor solución generada en ambas trayectorias se le aplica la búsqueda local para mejorar su calidad. Como se muestra en la Figura 1, siempre se mantiene la mejor solución en el conjunto de élite (x^1) durante la realización de la fase GRASP y sólo se reemplaza cuando una nueva solución generada la mejora en calidad. El algoritmo finaliza cuando PR se aplica a todos los pares de *ES* y la mejor solución obtenida x^{best} se devuelve como salida.

Como se ha mencionado anteriormente, la distancia se usa para medir cómo de diversa es una solución con respecto a un conjunto de soluciones. Especificamente, para

```
begin GRASP+PR Estático
1   GlobalIter ← número de iteraciones globales
2   Aplicar GRASP (construcción y mejora) durante
    b = |ES| iteraciones para poblar
    ES ← {x^1, x^2, ..., x^b}
3   iter ← b + 1
4   while iters ≤ GlobalIter do
5       x ← construcción GRASP
6       x' ← búsqueda local GRASP comenzando en x
7       if z_MM(x') > z_MM(x^1) or
        (z_MM(x') > z_MM(x^b) and d(x', ES) ≥ dth) then
8           x^k ← solución más parecida a x' en ES
            con z_MM(x') > z_MM(x^k)
9           ES ← ES \ {x^k}
10          Insertar x' en ES de forma que ES
            esté ordenado de la mejor x^1 a la peor x^b
11      end-if
12      iters ← iters + 1
13  end-while
14  x^best ← x^1
15  for (i = 1 to b − 1 and j = i + 1 to b do
16      Aplicar PR(x^i, x^j) y PR(x^j, x^i) y
        hacer x ← mejor solución encontrada
17      x' ← búsqueda local GRASP comenzando en x
18      if z_MM(x') > z_MM(x^best) then
19          x^best ← x'
20      end-if
21  end-for
22  return x^best
end
```

Fig. 1. GRASP con PR estático.

el MMDP, se define x_i^r como el valor de la i-ésima variable de la solución de élite $r \in ES$ y x_i^t como el valor de la i-ésima variable para la solución candidata t. La distancia entre la solución candidata t y las soluciones del *ES* se calcula con la expresión $d(t, ES) = b \cdot m - \sum_{r=1}^{b} \sum_{i:x_i^t=1} x_i^r$. Esta expresión simplemente cuenta el número de veces que cada elemento seleccionado en la solución candidata t aparece en las soluciones de élite y resta este valor a la máxima distancia posible (es decir, $b \cdot m$). La distancia máxima se obtiene cuando ningún elemento que está seleccionado en la solución candidata t aparece en las soluciones de élite del *ES*.

Una implementación alternativa de GRASP con PR consiste en una actualización *dinámica* del conjunto de élite [11]. En este diseño, denominado GRASP con PR dinámico, a cada solución x' generada con GRASP se le aplica el algoritmo PR entre ella y una solución x^j seleccionada del *ES*. La selección se realiza de forma probabilística de forma proporcional al valor de cada solución. Al igual que en el diseño estático, el método de búsqueda local se aplica a la salida del PR, pero ahora, la solución resultante se evalúa para su entrada en el *ES*. Si la solución se inserta, se puede usar como una solución guía en posteriores aplicaciones del PR. La Figura 2 muestra el pseudo-código para este diseño dinámico.

begin GRASP+PR Dinámico
1 *GlobalIter* ← número de iteraciones globales
2 Aplicar GRASP (construcción y mejora) durante $b = |ES|$ iteraciones para poblar $ES \leftarrow \{x^1, x^2, \ldots, x^b\}$
3 $iter \leftarrow b + 1$
4 **while** $iters \leq GlobalIter$ **do**
5 $x \leftarrow$ construcción GRASP
6 $x' \leftarrow$ búsqueda local GRASP comenzando en x
7 Seleccionar aleatoriamente x^j de ES
8 Aplicar PR(x', x^j) y PR(x^j, x') y hacer $y \leftarrow$ mejor solución encontrada
9 $y' \leftarrow$ búsqueda local GRASP comenzando en y
10 **if** $z_{MM}(y') > z_{MM}(x^1)$ **or** ($z_{MM}(y') > z_{MM}(x^b)$ **and** $d(y', ES) \geq dth$) **then**
11 $x^k \leftarrow$ solución más parecida a y' en ES con $z_{MM}(y') > z_{MM}(x^k)$
12 Insertar y' en ES y borrar x^k
13 Ordenar ES de la mejor x^1 a la peor x^b
14 **end-if**
15 **end-while**
16 $x^{best} \leftarrow x^1$
17 **return** x^{best}
end

Fig. 2. GRASP con PR dinámico.

Resende y Werneck [15] introdujeron el *reencadenamiento de trayectorias evolutivo* como una fase de procesamiento posterior para GRASP con PR. En este diseño, denomiado GRASP con PR evolutivo o GRASP con EvPR, las soluciones en el conjunto élite (*ES*) evolucionan de forma similar a como el conjunto de referencia evoluciona en la búsqueda dispersa (*Scatter Search*, SS) [12]. La Figura 3 muestra el pseudo-código de GRASP con EvPR.

Al igual que en GRASP con PR dinámico, en GRASP con EvPR se aplican, en cada iteración, las fases de construcción y mejora de GRASP así como el método PR para obtener el conjunto de élite (ver pasos 5 a 9 en el pseudo-código mostrado en la Figura 2). Después de un número determinado de iteraciones el GRASP con PR dinámico se detiene. En cambio, en GRASP con EvPR, se aplica a cada par de soluciones en *ES* una fase de procesamiento posterior basada en PR. Las soluciones obtenidas con esta última aplicación de PR se consideran como candidatas para entrar en el *ES*, y PR se aplica de nuevo mientras nuevas soluciones entren en el *ES*. Debido a esto se dice que el *ES* evoluciona. Todo este proceso se repite *GlobalIter* iteraciones.

GRASP con EvPR y búsqueda dispersa (SS) son métodos basados en la evolución de un pequeño conjunto de soluciones (conjunto de élite en el primero y conjunto de referencia en el segundo). Por tanto, se pueden observar aspectos similares en ellos. En algunas implementaciones de SS, GRASP se usa para poblar el conjunto de referencia, pero se pueden usar otros métodos constructivos. De forma similar, PR se puede usar para combi-

begin GRASP+EvPR
1 *GlobalIter* ← número de iteraciones globales
2 Aplicar GRASP (construcción y mejora) durante $b = |ES|$ iteraciones para poblar $ES \leftarrow \{x^1, x^2, \ldots, x^b\}$
3 **for** $iter = 1, \ldots,$ *GlobalIter* **do**
4 **for** $i = 1, \ldots,$ *LocalIter* **do**
5 $x \leftarrow$ construcción GRASP
6 $x' \leftarrow$ búsqueda local GRASP comenzando en x
7 Seleccionar aleatoriamente x^j de ES
8 Aplicar PR(x', x^j) y PR(x^j, x') y hacer $y \leftarrow$ mejor solución encontrada
9 $y' \leftarrow$ búsqueda local GRASP comenzando en y
10 **if** $z_{MM}(y') > z_{MM}(x^1)$ **or** ($z_{MM}(y') > z_{MM}(x^b)$ **and** $d(y', ES) \geq dth$)
11 $x^k \leftarrow$ solución más parecida a y' en ES con $z_{MM}(y') > z_{MM}(x^k)$
12 Insertar y' en ES y borrar x^k
13 Ordenar ES de la mejor x^1 a la peor x^b
14 **end-if**
15 **end-for**
16 *NewSol* ← TRUE
17 **while** *NewSol* **do**
18 *NewSol* ← FALSE
19 Aplicar PR(x, x') y PR(x', x) por cada par (x, x') en ES no combinado previamente y hacer $y \leftarrow$ mejor solución encontrada
20 $y' \leftarrow$ búsqueda local GRASP comenzando en y
21 **if** $z_{MM}(y') > z_{MM}(x^1)$ **or** ($z_{MM}(y') > z_{MM}(x^b)$ **and** $d(y', ES) \geq dth$)
22 $x^k \leftarrow$ solución más parecida a y' en ES con $z_{MM}(y') > z_{MM}(x^k)$
23 Insertar y' en ES y borrar x^k;
24 Ordenar ES de la mejor x^1 a la peor x^b
25 *NewSol* ← TRUE
26 $x^{best} \leftarrow x^1$
27 **end-if**
28 **end-while**
29 **end-for**
30 **return** x^1
end

Fig. 3. GRASP con EvPR.

nar soluciones en SS, pero se puede usar cualquier otro método de combinación. Desde un punto de vista algorítmico, se pueden encontrar dos diferencias principales entre estos métodos. La primera es que en SS no se aplica PR a las soluciones obtenidas con GRASP (como se hace en los pasos 7 y 8 en el pseudo-código de GRASP con EvPR mostrado en la Figura 3), en cambio, sólo se aplica PR como un método de combinación entre las soluciones que están en el conjunto de referencia. La segunda diferencia es que en SS cuando ninguna de las nuevas solu-

ciones obtenidas con las combinaciones se admiten para entrar en el conjunto de referencia (conjunto élite), éste se reconstruye, eliminando algunas de sus soluciones. En GRASP con EvPR no se eliminan soluciones del *ES*, en cambio, se aplica GRASP de nuevo (iniciando desde el paso 5) y se usan las mismas reglas para la inserción en el *ES*.

V. Experimentos computacionales

Esta sección describen los experimentos computacionales que se han realizado para comprobar la eficiencia de los métodos propuestos y compararlos con los métodos previos identificados para el MMDP. Los métodos han sido implementados en Java SE 6. Todos los experimentos se han ejecutado con un ordenador Pentium 4 a 3 Ghz con 3 GB de RAM. Se han empleado dos conjuntos de instancias en nuestros experimentos:

- *Geo*: Está formado por 60 matrices cuyos valores se han calculado como distancias Euclídeas de puntos generados de forma aleatoria con coordenadas en el rango de 0 a 100. El número de dimensiones de los puntos se genera también de forma aleatoria entre 2 y 21. Este generador fue presentado en [8]. Se han generado 20 instancias con $n = 100, 250$, y 500. Para cada valor de n se ha considerado $m = 0{,}1n$, $0{,}3n$ (generando 10 instancias por cada combinación de n y m).
- *Ran*: Está formado por 60 matrices con números enteros generados aleatoriamente. Están basadas en el generador introducido en [17]. Al igual que en el conjunto *Geo*, se han generado 20 instancias con $n = 100, 250$, y 500 (y por cada valor de n se consideran $m = 0{,}1n$, $0{,}3n$). Los números enteros aleatorios se han generado entre 50 y 100 en todas las instancias excepto cuando $n = 500$ y $m = 150$ en cuyo caso han sido generadas entre 1 y 200 (para una mejor comparación entre heurísticas).

En cada experimento, se calcula para cada instancia el mejor valor de la solución, *BestValue*, obtenido por la ejecución de los métodos considerados. Posteriormente, por cada método, se calcula la desviación relativa en porcentaje entre el valor de la mejor solución obtenida con ese método y *BestValue* para esa instancia. En los resultados experimentales se indica la media de esta desviación relativa (Dev.) entre todas las instancias de cada experimento particular. Finalemente se indica, para cada método, el número de instancias (#Best) en las que el valor de la mejor solución obtenido con este método es igual a *BestValue*.

En la experimentación preliminar se considera un conjunto de 40 instancias formadas con 10 instancias del conjunto *Geo* con $n = 100$ y 10 instancias con $n = 250$ y, de forma similar, del conjunto *Ran* (la mitad con $m = 0,1n$ y la otra mitad con $m = 0,3n$). En el primer experimento preliminar, se estudia el parámetro α en el método constructivo GRC así como el parámetro β en el método constructivo GRC2. Se ejecutan GRC y GRC2 100 veces, de forma que se obtienen 100 soluciones por cada par método-instancia. La Tabla II muestra, para este conjunto de 40 instancias y cada valor de α, los valores Dev. y #Best. En esta tabla se puede observar cómo el mejor resultado se obtiene cuando el método constructivo GRC2 se ejecuta con un valor de $\beta = 0{,}90$. Por lo tanto, se usará este método en el resto de la experimentación.

TABLA II
Constructivos en instancias con $n = 100, 250$.

		Dev.	**#Best**
GRC(α)	0,75	9,23 %	0
	0,90	2,51 %	5
	0,95	1,09 %	10
GRC2(β)	0,75	0,70 %	21
	0,90	0,58 %	21
	0,95	0,66 %	18

En el segundo experimento preliminar, se ha comparado el método constructivo GRC2(0,9) con los dos métodos constructivos previos para el MMDP: ErkC [3] y GhoC [6]. También se ha considerado un constructivo aleatorio (RanC) en el que los m elementos de la solución se seleccionan de forma aleatoria como una línea base para la comparación. Se generan 100 soluciones con cada método por cada instancia y se muestran las dos estadísticas descritas anteriormente.

TABLA III
Constructivos en instancias con $n = 100, 250$.

	Dev.	**#Best**
RanC	22,85 %	0
ErkC	6,08 %	4
GhoC	1,68 %	15
GRC2	0,12 %	37

Los resultados en la Tabla III muestran claramente la superioridad del método propuesto (GRC2) en estas instancias. El método es capaz de obtener la mejor solución en 37 instancias de un total de 40. Respecto a la desviación media, los métodos ErkC, GhoC y GRC2 obtienen un valor de 6,08 %, 1,68 % y 0,12 % respectivamente, lo que muestra la superioridad del método GRC2. Como se esperaba, RanC obtiene soluciones de baja calidad.

En el tercer experimento preliminar, se comparan los constructivos con métodos de búsqueda local para el MMDP. Concretamente, el experimento se centra en los métodos con constructivo y mejora ErkC+BLS [3] y GhoC+BLS [6]. Se considera el método de búsqueda local propuesto en la Subsección III-B, FLS, en combinación con el método constructivo GRC2. Se denomina GRASP1 al método constructivo GRC2(0,9) en combinación con la búsqueda local FLS. Se construyen y mejoran 100 soluciones para cada instancia con estos cuatro métodos y se presentan estadísticas de las mejores soluciones obtenidas en la Tabla IV.

La Tabla IV muestra que los enfoques propuestos basados en la metodología GRASP son capaces de mejorar

TABLA IV
BÚSQUEDA LOCAL EN INSTANCIAS CON $n = 100, 250$.

	Dev.	**#Best**
ErkC+BLS	2,40 %	8
GhoC+BLS	0,82 %	22
GRASP1	0,24 %	29

los métodos previos que se basan en la construcción y mejora. Especificamente, GRASP1 presenta una desviación media de la mejor solución obtenida en este experimento de 0,24 %, mientras que ErkC+BLS y GhoC+BLS obtienen 2,4 % y 0,82 % respectivamente.

En el cuarto experimento preliminar se comparan el diseño estático y el diseño dinámico de GRASP con PR. El método de GRASP con PR depende del parámetro *dth* que especifica la distancia mínima para que una solución entre en el conjunto de élite (*ES*). La Tabla V presenta, para el conjunto de 40 instancias consideradas en los experimentos preliminares y cuatro valores diferentes de *dth*, la desviación media desde la mejor solución obtenida (Dev.), el número de mejores soluciones (#Best) y el tiempo medio de CPU (Time) en segundos.

TABLA V
PR EN INSTANCIAS CON $n = 100, 250$.

	dth	**Dev.**	**#Best**	**Time**
GRASP con PR	4	0,65 %	20	11,0s
estático	8	0,63 %	21	11,3s
	10	0,54 %	24	10,7s
	12	0,76 %	20	11,0s
GRASP con PR	4	0,21 %	30	18,1s
dinámico	8	0,32 %	29	18,6s
	10	0,49 %	22	18,3s
	12	0,47 %	22	18,2s

La Tabla V muestra claramente que GRASP con PR dinámico obtiene mejores soluciones que GRASP con PR estático, aunque consume más tiempo de ejecución (18 segundos en media comparado con los 11 segundos del diseño estático). Además, esta tabla también muestra que el mejor valor de *dth* en la versión dinámica es 4, puesto que el método obtiene una desviación media de 0,21 % y 30 mejores soluciones, lo que mejora las otras configuraciones que se muestran. Por lo tanto, para los siguientes experimentos se fija el valor de *dth* a 4 y se usa el diseño dinámico.

En el experimento final, se comparan los mejores métodos propuestos en este trabajo con los métodos del estado del arte para el MMDP. Concretamente, se consideran los seis algoritmos siguientes (todos ellos se ejecutan con 100 iteraciones globales excepto GRASP+EvPR):

- *GhoC+BLS*:Método multi-arranque [6].
- *SA*: Recocido Simulado [9].
- *TS*: Búsqueda Tabú [9].
- *GRASP1*: Método constructivo GRC2(0,9) acoplado con la búsqueda local FLS.
- *GRASP+PR Din*: GRASP con reencadenamiento de trayectorias dinámico con *dth*=4.
- *GRASP+EvPR*: GRASP con reencadenamiento de trayectorias evolutivo con *dth*=4, *GlobalIter*=5 y *LocalIter*=20.

La Tabla VI presenta, para cada método en cada conjunto de instancias, la desviación media relativa en porcentaje (Dev.) entre los valores de la mejor solución obtenida con cada método y el mejor conocido, el número de instancias (#Best) en las que el valor de la mejor solución obtenida con cada método es igual al mejor conocido, y el tiempo de CPU medio en segundos (Time).

TABLA VI
COMPARACIÓN DE LOS MEJORES MÉTODOS

		Dev.	**#Best**	**Time**
Geo	GhoC+BLS	0,75 %	10	2,45s
$n = 100$	SA	0,00 %	19	20,96s
	TS	0,00 %	20	33,64s
	GRASP1	0,76 %	10	0,68s
	GRASP+PR Din	0,11 %	16	1,68s
	GRASP+EvPR	0,09 %	17	3,76s
Geo	GhoC+BLS	1,00 %	0	30,50s
$n = 250$	SA	0,68 %	6	220,57s
	TS	1,75 %	2	439,68s
	GRASP1	1,11 %	1	5,58s
	GRASP+PR Din	0,19 %	7	33,44s
	GRASP+EvPR	0,16 %	14	65,57s
Geo	GhoC+BLS	2,36 %	0	282,37s
$n = 500$	SA	3,48 %	0	1449,85s
	TS	9,27 %	0	3633,36s
	GRASP1	2,39 %	0	34,99s
	GRASP+PR Din	0,25 %	7	788,31s
	GRASP+EvPR	0,04 %	16	1465,44s
Ran	GhoC+BLS	1,71 %	4	1,37s
$n = 100$	SA	2,89 %	9	10,82s
	TS	3,28 %	10	33,11s
	GRASP1	1,37 %	7	0,84s
	GRASP+PR Din	0,61 %	14	2,96s
	GRASP+EvPR	0,49 %	15	7,36s
Ran	GhoC+BLS	2,01 %	3	15,98s
$n = 250$	SA	3,73 %	0	115,10s
	TS	7,49 %	0	430,07s
	GRASP1	1,34 %	5	19,22s
	GRASP+PR Din	0,81 %	11	101,57s
	GRASP+EvPR	0,26 %	17	271,05s
Ran	GhoC+BLS	2,95 %	13	93,05s
$n = 500$	SA	41,62 %	0	868,00s
	TS	41,62 %	0	3606,49s
	GRASP1	1,70 %	14	99,02s
	GRASP+PR Din	0,18 %	18	2172,38s
	GRASP+EvPR	0,27 %	17	6349,20s

La Tabla VI muestra el mérito de los procedimientos propuestos. Los algoritmos GRASP con PR dinámico y

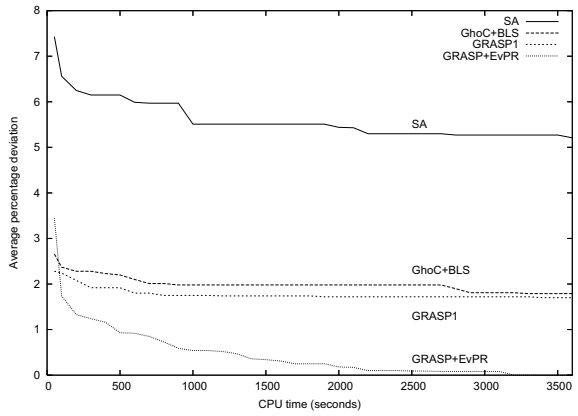

Fig. 4. Perfil de búsqueda en una instancia con $n = 500$.

GRASP con EvPR consistentemente producen las mejores soluciones con desviaciones medias más pequeñas que los otros métodos (y con un número mayor de mejores soluciones encontradas que los otros). GRASP con EvPR presenta una mejora marginal cuando se compara con GRASP con PR dinámico, pero requiere mayores tiempos de ejecución (especialmente para los ejemplos grandes). Por otro lado, el algoritmo GRASP1 es capaz de obtener relativamente buenas soluciones empleando poco tiempo computacional, con un rendimiento muy similar al método GhoC+BLS. Los métodos SA y TS tienen buen rendimiento en instancias pequeñas de *Geo* ($n = 100$) pero son claramente inferiores a los otros métodos presentados en la comparación cuando el objetivo son instancias de gran tamaño ($n = 500$). La clasificación de los métodos con respecto al número de mejores soluciones encontradas en las 120 instancias *Geo* y *Ran* es: GRASP+EvPR(96), GRASP+PR Din(73), GRASP1(37), SA(34), TS(32), and GhoC+BLS(30).

La Figura 4 muestra el perfil de búsqueda típico para los métodos que se han comparado. Esta ejecución corresponde a las instancias más grandes de tipo *Geo* ($n = 500$, $m = 150$) con un tiempo límite de 3.600 segundos para cada instancia y método. Esta figura muestra que GRASP+EvPR mejora a los otros métodos en el horizonte temporal grande (3.600 segundos en este experimento). Además, es interesante destacar que GRASP+EvPR obtiene soluciones de alta calidad (mejores que las de los métodos en la comparativa) desde las primeras iteraciones (100 segundos). Por otro lado, SA presenta un rendimiento bajo cuando se compara con los otros tres métodos de este experimento.

El método GRASP1 obtiene las mejores soluciones en los primeros 50 segundos. No obstante, GRASP1 por si mismo (sin un proceso posterior con PR) no es capaz de mejorar estas soluciones iniciales y presenta un perfil plano durante la búsqueda.

VI. Conclusiones

El objetivo de este estudio ha sido ampliar nuestro conocimiento sobre el diseño de los procedimientos de reencadenamiento de trayectorias (PR) para la optimización combinatoria. Se ha estudiado la generación de soluciones con GRASP y su combinación con PR. También se han los diseños GRASP con PR estático, GRASP con PR dinámico y GRASP con PR evolutivo. Se han realizado varios experimentos con instancias previamente presentadas. Los experimentos muestran que GRASP con PR dinámico y GRASP con PR evolutivo son los mejores métodos para las instancias del MMDP probadas en este artículo. Además, los resultados indican que las heurísticas híbridas propuestas presentan mejores resultados que las metaheurísticas previas, como búsqueda tabú y recocido simulado.

Agradecimientos

Esta investigación se ha financiado parcialmente por el *Ministerio de Educación y Ciencia* de España (TIN2006-02696).

Referencias

[1] R. Chandrasekaran and A. Daughety. Location on tree networks: p-centre and n-dispersion problems. *Mathematics of Operations Research*, 6:50-57, 1981.

[2] A. Duarte and R. Martí Tabu Search and GRASP for the maximum diversity problem. *European Journal of Operational Research*, 178:71-84, 2007.

[3] E. Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46:48-60, 1990.

[4] H. Faria Jr., S. Binato, M.G.C. Resende, and D.J. Falcao. Transmission network design by a greedy randomized adaptive path relinking approach. *IEEE Transactions on Power Systems*, 20:43-49, 2005.

[5] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67-71, 1989.

[6] J.B. Ghosh. Computational aspects of the maximum diversity problem. *Operations Research Letters*, 19:175-181, 1996.

[7] F. Glover. Tabu search and adaptive memory programing - Advances, applications and challenges. In R.S. Barr, R.V. Helgason, and J.L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1-75. Kluwer Academic Publishers, 1996.

[8] F. Glover, C.C. Kuo, and K.S. Dhir. Heuristic algorithms for the maximum diversity problem. *Journal of Information and Optimization Sciences*, 19:109-132, 1998.

[9] R.K. Kincaid. Good solutions to discrete noxious location problems via metaheuristics. *Annals of Operations Research*, 40:265-281, 1992.

[10] C.C. Kuo, F. Glover, and K.S. Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24:1171-1185, 1993.

[11] M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44-52, 1999.

[12] M. Laguna and R. Martí. *Scatter search: Methodology and implementations in C*. Kluwer Academic Publishers, 2003.

[13] M. Lundi and A. Mess. Convergence of an annealing algorithm. *Mathematical Programming*, 34:111-124, 1986.

[14] M. Prais and C.C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164-176, 2000.

[15] M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10:59-88, 2004.

[16] C.C. Ribeiro, E. Uchoa, and R.F. Werneck. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing*, 14:228-246, 2002.

[17] G.C. Silva, L.S. Ochi, and S.L. Martins. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. *Lecture Notes in Computer Science*, 3059:498-512, 2004.