Oscilación Estratégica para el Problema de Maximización de la Diversidad en Grupos

Micael Gallego[1], Manuel Laguna[2], Rafael Martí[3], Abraham Duarte[4]

Resumen-- **En este trabajo se proponen nuevos métodos para la resolución del problema de maximización de la diversidad en grupos o MDGP por sus siglas en inglés *maximally diverse grouping problem*. Este problema NP-difícil consiste en formar grupos diversos, con igual o distinto tamaño, de un conjunto dado de elementos de forma diversa. Los algoritmos propuestos en este trabajo se han diseñado para la formulación más general que permite diferentes tamaños. El MDGP tiene diversas aplicaciones como por ejemplo la creación de grupos de estudiantes diversos o la formación de grupos de revisores en el contexto académico. En este trabajo presentamos procedimientos constructivos y de mejora para el MDGP basados en la búsqueda tabú junto con la oscilación estratégica. Nuestro método permite a la búsqueda escapar de los límites de la factibilidad de la solución durante un número determinado de iteraciones. Hemos realizado un gran número de experimentos computacionales que comparan los métodos propuestos con los procedimientos del estado del arte y se ha mostrado la efectividad de la oscilación estratégica para el MDGP.**

Palabras clave—**Diversidad, metaheurísticas, oscilación estratégica.**

I. INTRODUCCIÓN

El problema de maximización de la diversidad en grupos (MDGP) consiste en la agrupación de un conjunto de M elementos en G grupos mutuamente disjuntos de forma que la diversidad de los elementos en cada grupo sea máxima. La diversidad entre los elementos de un grupo se calcula como la suma de las distancias individuales entre cada par de elementos, donde la definición de distancia depende del contexto en el que se aplique el problema. El objetivo del problema consiste en maximizar la diversidad total, es decir, la suma de la diversidad de cada uno de los grupos. Feo y Khellaf [1] demostraron que el MDGP es NP-difícil.

El MDGP ha recibido diferentes nombres a lo largo de la historia. Se denominó "el problema de la partición k" (*k-partition problem*) en Feo y colaboradores [2] y "el problema de la partición equitativa" (*equitable partition problem*) en O'Brien y Mingers [3]. Este problema aparece en una gran cantidad de aplicaciones reales, como el diseño de circuitos VLSI [1] o el almacenamiento de grandes programas en memoria paginada [4]. Una de las aplicaciones más populares del MDGP aparece en el contexto académico, con la formación de grupos de estudiantes [5] en el que es habitual crear grupos de trabajo diversos para proporcionar un ambiente diverso a los alumnos [6]. Este problema también se aplica en la formación de revisores de publicaciones científicas o proyectos de investigación [7].

Para poder formular el MDGP en términos matemáticos se considera que, dado un conjunto de elementos $\{1, 2, \dots, M\}$, entre cada par de elementos i y j se conoce una distancia d_{ij} que puede ser Euclídea o de cualquier otro tipo.

Hemos identificado en la literatura dos variantes del MDGP. La primera (MDGP1) es la más conocida y obliga a que todos los grupos tengan el mismo número S de elementos, con $S = M/G$. La segunda variante (MDGP2) permite que el tamaño S_g de cada grupo g esté dentro del intervalo $[a_g, b_g]$, donde $a_g \leq b_g$ para $g = 1, \dots, G$. Obviamente, MDGP1 es un caso particular de MDGP2 en el que $S_g = a_g = b_g$ para todo g. El procedimiento que presentamos en en este trabajo está diseñado para el caso general, MDGP2, aunque ha sido probado en ambas variantes MDGP1 y MDGP2. En el resto del artículo, MDGP se refiere al caso general MDGP2.

Ambas variantes se pueden formular como programas cuadráticos enteros con variables binarias x_{ig} que toman el valor 1 si el elemento i está en el grupo g y 0 en otro caso. La formulación para el MDGP1 es:

$$\text{Maximizar} \quad \sum_{g=1}^{G} \sum_{i=1}^{M-1} \sum_{j>i}^{M} d_{ij} x_{ig} x_{jg}$$

$$\text{sujeto a} \quad \sum_{g=1}^{G} x_{ig} = 1 \qquad i = 1,2, \dots, M$$

$$\sum_{i=1}^{M} x_{ig} = S \qquad g = 1,2, \dots, G$$

$$x_{ig} \in \{0,1\} \qquad i = 1, \dots, M; \; g = 1, \dots, G$$

[1] Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Spain. E-mail: Micael.Gallego@urjc.es
[2] Leeds School of Business, University of Colorado at Boulder, USA. E-mail: laguna@colorado.edu
[3] Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain. E-mail: Rafael.Marti@uv.es
[4] Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Spain. E-mail: Abraham.Duarte@urjc.es

La función objetivo suma la distancia de todos los pares de elementos que pertenecen al mismo grupo. El primer conjunto de restricciones fuerza la asignación de cada elemento a un único grupo. El segundo conjunto de restricciones fuerzan a que todos los grupos tengan tamaño S. En el caso general, MDGP2, el segundo grupo de restricciones se reemplaza con:

$$a_g \leq \sum_{i=1}^{M} x_{ig} \leq b_g \qquad g = 1,2,\dots,G$$

Ambos problemas (MDGP1 y MDGP2) son equivalentes en términos de la formulación matemática. No obstante, la generalización que permite que los grupos tengan tamaños diferentes tiene implicaciones a la hora de desarrollar procedimientos heurísticos porque el espacio de soluciones es más grande para el MDGP2. En particular, los métodos diseñados para la resolución del MDGP1 pueden focalizar su búsqueda en el área del espacio de soluciones en las que todos los grupos tienen el mismo tamaño.

La siguiente sección resume la literatura más relevante para el MDGP. Está seguida de una descripción de los tres métodos propuestos para la construcción de soluciones factibles, uno de los cuales utiliza mecanismos de memoria. El procedimiento de búsqueda propuesto, basado en la metodología tabú, también se describe en la Sección 3. En la Sección 4, se presenta un método de mejora basado en la técnica de la oscilación estratégica. A continuación, mostramos los experimentos computacionales realizados con instancias propuestas en la literatura y otras nuevas más grandes presentadas en este trabajo. Los análisis estadísticos muestran la eficiencia del método propuesto cuando se compara con los métodos previos.

II. Métodos Previos

El MDGP ha sido objeto de estudio en los últimos 21 años, comenzando con el algoritmo multiarranque propuesto por Arani y Lotfi [8]. Este método está formado por una construcción aleatoria seguida de una fase de mejora. En la fase de mejora se destruye la solución parcialmente y se escanean todas las posibles reconstrucciones seleccionando la mejor de todas. Este proceso se repite hasta que la solución no cambia entre reconstrucciones (es decir, cuando la solución actual no se puede mejorar más siguiendo este procedimiento).

Feo y Khellaf [1] propusieron varios heurísticos basados en la teoría de grafos para los casos especiales en los que los grupos tienen un determinado número de elementos. En concreto, diseñaron métodos para grupos con número par de elementos, con número impar y cuyo tamaño sea potencia de 2. Además, los autores demostraron que los valores obtenidos por los heurísticos están acotados por un porcentaje del valor de la solución óptima. Weitz y Jelassi [5] desarrollaron un heurístico constructivo básico denominado WJ. Está basado en la idea de evitar la asignación en el mismo grupo de elementos similares. El método comienza con la asignación aleatoria de un elemento al primer grupo. Luego selecciona el elemento con la menor distancia al elemento seleccionado previamente y le asigna al siguiente grupo. Cuando un elemento está asignado a cada uno de los grupos, el procedimiento comienza de nuevo por el primer grupo. La construcción finaliza cuando todos los elementos han sido asignados. La Figura 1 resume este procedimiento.

1. Seleccionar un elemento inicial de forma aleatoria y asignarlo al grupo 1.
2. De los elementos que no han sido asignados todavía, seleccionar aquél con menor distancia al último elemento seleccionado (si hay empate se selecciona aleatoriamente). Asignar este elemento al siguiente grupo. Si el grupo es el último, comenzar de nuevo por el grupo 1.
3. Finalizar si todos los elementos han sido asignados; si no, ir al paso 2.

Fig. 1. Método WJ

Weitz y Lakshminarayanan [9] propusieron el método LC. Este método es una versión adaptada de otro procedimiento previo publicado originalmente por Lofti y Cerveny [10] como parte de un algoritmo para la planificación de exámenes finales. Weitz y Lakshminarayanan [9] corrigieron una serie de errores que habían encontrado en el método original y publicaron LC en [6].

Estos mismos autores diseñaron un nuevo método partiendo de LC llamado LCW [6]. El método LCW se resume en la Figura 2.

Construir una solución inicial.
do {
  for ($i = 1, \dots, M$) {
    1. Encontrar el elemento j en cualquier grupo para el que un intercambio de grupo entre el elemento i y el elemento j resultaría en el mayor incremento del valor de la función objetivo.
    2. Si el incremento de la función objetivo es estrictamente positivo, realizar el intercambio.
  }
} **while** (se haya realizado algún intercambio de elementos)

Fig. 2. Método LCW

Weitz y Lakshminarayanan [6] realizaron diversos experimentos computacionales y compararon todas las heurísticas para el MDGP conocidas hasta ese momento junto con los método que proponían en ese mismo trabajo. Sus resultados experimentales les indicaron que el mejor método era LCW aplicado sobre una solución generada de forma aleatoria.

El método más reciente para la resolución del MDGP presentado en la literatura es el LSGA, un algoritmo propuesto por Fan y colaboradores [11]. LSGA es un algoritmo genético híbrido que combina un algoritmo genético y una búsqueda local. Esta hibridación se conoce habitualmente como algoritmo memético (MA) y ha sido empleada en numerosas ocasiones para la resolución de otros problemas de optimización. La parte genética de LSGA está basada en el esquema de codificación de grupos propuesto por Falkenauer [13]. La búsqueda local implementa una estrategia de tipo *mejor mejora* (*best improvement*) basada en seleccionar el mejor intercambio de elementos entre grupos de entre todos los posibles. El método LSGA es el primer método en la literatura que aborda la resolución de la versión general del MDGP (es decir, MDGP2), que permite diferentes tamaños de grupos. La estructura básica de este algoritmo memético se presenta en la Figura 3.

Fan y colaboradores [11] realizaron un extenso conjunto de experimentos computacionales en los que compararon el método LSGA, un algoritmo genético puro (sin búsqueda local) y el método LCW [6]. Sus experimentos mostraron la efectividad de LSGA para la resolución de instancias del MDGP con el mismo o diferente tamaño de los grupos. Chen y colaboradores [12] usaron este procedimiento en una aplicación práctica del MDGP conocida como "El problema de la construcción de grupos de revisores" (*Reviewer group construction problem*).

III. Construcción y Mejora Tabú

El principal objetivo de nuestro trabajo es desarrollar un procedimiento para el MDGP basado en la metodología tabú. En las Secciones 3 y 4 describimos los elementos del procedimiento propuesto: 1) construcción de la solución inicial, 2) búsqueda local y 3) oscilación estratégica. A continuación describimos estos elementos por separado porque se pueden combinar de diferentes formas. Un método para el MDGP se puede crear simplemente con un procedimiento de mejora y otro método puede consistir en la construcción de una solución y su posterior mejora con procedimiento de búsqueda local. Además, la oscilación estratégica está diseñada de forma que puede usarse con cualquier método constructivo y procedimiento de mejora. En vez de evaluar todas las posibles combinaciones, en la Sección de Experimentos Computacionales se evaluarán de forma secuencial las fases de construcción y mejora para elegir la mejor combinación a la que añadir la oscilación estratégica.

En este trabajo presentamos un método constructivo voraz que permite resolver ambas versiones del problema, es decir, aquellas en las que los grupos deben tener el mismo tamaño y aquellas en las que está permitido que los grupos tengan diferentes tamaños (dentro de unos límites). Hemos denominado al método GC (construcción voraz, *greedy construction*). Este se inicia seleccionando G elementos de forma aleatoria y asignando cada uno de ellos a un grupo diferente. Por tanto, al final del primer paso, cada grupo tiene asignado un elemento. Después, el procedimiento realiza $M - G$ iteraciones para asignar a grupos a los elementos no asignados. Para generar una solución factible, las iteraciones se dividen en dos fases. En la primera fase, los elementos se asignan a grupos con un número de elementos menor que el número mínimo de elementos de ese grupo (a_g). En la segunda fase, los elementos restantes se asignan a grupos cuyo número de elementos es menor que el máximo número de elementos de ese grupo (b_g).

1. Iniciar con una población de soluciones aleatorias de tamaño $populationSize$.
2. Calcular el valor de la función objetivo de cada solución en la población.
3. Repetir los siguientes pasos hasta que se haya creado una descendencia de tamaño $populationSize$.
   a. Seleccionar un par de soluciones padre de la población actual, con la probabilidad de selección directamente proporcional al valor de la solución. La selección se realiza "con reemplazo", lo que quiere decir que la misma solución puede ser seleccionada en más de una ocasión para ser un padre.
   b. Combinar los padres seleccionados para crear una nueva solución.
   c. Mutar la descendencia creada con probabilidad mp (probabilidad de mutación, *mutation probability*)
   d. Reparar la solución si no es factible.
   e. Mejorar la solución con un método de mejora.
4. Crear una nueva población con las mejores $populationSize$ soluciones de la población original y la población de descendencia.
5. Ir al paso 2.

Fig. 3. Algoritmo memético LSGA

Se define E_g como el conjunto de elementos asignados actualmente al grupo g. Las iteraciones de cada fase se inician con la identificación de los grupos que tienen: i) menos elementos que los mínimos necesarios si es la primera fase (todos los g para los que $|E_g| < a_g$); o ii) menos elementos que el máximo permitido si es la segunda fase (todos los g para los que $|E_g| < b_g$). Entonces, se selecciona aleatoriamente un elemento no asignado i y se añade al grupo (dentro del conjunto identificado) para el que la media de la distancia entre el elemento i y todos los elementos del grupo se maximiza. Es decir, i se asigna al grupo g en el que D_{ig} se maximiza:

$$D_{ig} = \frac{\sum_{j \in E_g} d_{ij}}{|E_g|}$$

La primera fase termina cuando todos los grupos contienen al menos el mínimo número de elementos (para cada grupo g, $|E_g| \geq a_g$). La segunda fase finaliza cuando todos los elementos han sido asignados. La Figura 4 resume el método GC.

1. Seleccionar aleatoriamente G elementos y asignar uno a cada grupo.
2. Repetir los siguientes pasos hasta $|E_g| = a_g$ para todos los grupos.
   a. Seleccionar aleatoriamente un elemento i no asignado a ningún grupo
   b. Asignar el elemento i al grupo g con $|E_g| < a_g$ que maximice D_{ig}.
3. Repetir los siguientes pasos hasta que todos los elementos se hayan asignado.
   a. Seleccionar aleatoriamente un elemento i no asignado a ningún grupo.
   b. Asignar el elemento i al grupo g con $|E_g| < b_g$ que maximice D_{ig}.

Fig. 4. Método GC

Además de la versión original, hemos desarrollado dos variantes del método GC. En la primera, denominada GC-FULL, hemos incluido ideas presentes en el método FULL propuesto por Mingers y O'Brien [14] y en la segunda, denominada GC-Tabu, hemos incluido elementos de la metodología tabú. En GC-FULL, en vez de la selección aleatoria de los elementos de los pasos 2.a y 3.a de la Figura 4 para realizar la asignación del elemento i al grupo g, GC-FULL busca el par (i, g) que maximice D_{ig} de entre todos los posibles. Por otro lado, hemos diseñado la variante GC-Tabu considerando que el procedimiento de construcción se utiliza dentro de un esquema multiarranque. GC-Tabu utiliza dos estructuras de memoria para guardar la información relevante asociada a las construcciones previamente generadas y aplica ese conocimiento para la construcción de nuevas soluciones. Concretamente, $freq(i,j)$ almacena el número de veces que los elementos i y j han estado asignados al mismo grupo en construcciones previas y $quality(i,j)$ almacena la calidad media (es decir, el valor de la función objetivo) de aquellas construcciones en las que los elementos i y j estaban en el mismo grupo. La distancia D_{ig} se modifica de acuerdo a la información en ambas estructuras de memoria para favorecer la asignación al mismo grupo de los pares de elementos con baja frecuencia y altos valores de calidad.

Los procedimientos de búsqueda local mejoran las soluciones examinando la vecindad de la solución actual y aplican sobre ella el *mejor movimiento* encontrado. Estos pasos se repiten hasta que se llega a la condición de parada. La definición de mejor movimiento depende del contexto y es una decisión de diseño importante al implementar las búsquedas locales. En la búsqueda local utilizada en LSGA, Fan y colaboradores [11] consideraron que la vecindad está formada por todos los posibles intercambios de elementos asignados a diferentes grupos. La vecindad completa se calcula en cada paso y el intercambio que produce el mayor incremento en la función objetivo es considerado el *mejor movimiento* y se aplica. Es decir, el método sigue una estrategia conocida como *mejor mejora* o *best improvement* (BI). El procedimiento finaliza cuando no encuentra ningún movimiento de mejora, por tanto, la solución final es un óptimo local. Una alternativa a BI consiste en finalizar la búsqueda en la vecindad en cuanto se encuentre un intercambio que mejore el valor de la solución actual. A esta estrategia se la denomina *primera mejora* o *first improvement* (FI). Los métodos LC y LCW utilizan otros criterios para explorar la vecindad.

Todos los métodos de búsqueda local propuestos en la literatura para el MDGP están basados en el intercambio de elementos entre grupos. Este es el diseño lógico para mantener la factibilidad durante la búsqueda cuando se aborda la variante del MDGP para la que todos los grupos deben tener el mismo tamaño. No obstante, para el caso general la vecindad se podría aumentar considerando movimientos que permitan transferir un único elemento de un grupo a otro (siempre que se mantenga la factibilidad). Estos movimientos se conocen en la literatura como *inserciones*. Nosotros hemos añadido estos movimientos de inserción a los movimientos de intercambio que se consideraban en los métodos de mejora previos para crear los métodos T-BI, T-FI y T-LCW. La "T" en estos procedimientos representa "Tabú" porque además de expandir la vecindad inicial para incluir

inserciones, también hemos añadido una memoria tabú a corto plazo para permitir que la búsqueda continúe más allá del óptimo local. Concretamente, cuando la búsqueda local llega a un punto en el que no se puede realizar ningún movimiento de mejora, se aplica el movimiento que menos empeore la función objetivo. En este punto, a los elementos que se cambian de un grupo a otro grupo no se les permite volver al mismo grupo durante $tabuTenure$ iteraciones. El proceso termina cuando se han realizado $maxIter$ iteraciones consecutivas sin encontrar una mejor solución que la mejor solución que se tenía hasta ese momento.

Los procedimientos de construcción y búsqueda local se pueden combinar de una forma sencilla. El procedimiento constructivo se ejecuta para crear una solución que posteriormente es mejorada con el procedimiento de búsqueda local. La mejor solución encontrada se mantiene y el proceso se repite hasta que se alcanza el tiempo máximo de ejecución. Hay que destacar que los métodos de mejora BI, FI y LCW finalizan cuando no se existe ningún movimiento que mejore la solución actual. En cambio, en las versiones "T" de estos métodos (T-BI, T-FI y T-LCW), la finalización del método depende del parámetro $maxIter$. Por tanto, para un tiempo límite especificado, el número de construcciones depende de la elección del método de mejora y del valor de $maxIter$.

IV. Oscilación estratégica

Hasta ahora hemos asumido que todas las búsquedas locales parten de una solución factible (una en la que todos los elementos están asignados y $a_g \leq |E_g| \leq b_g$ para todos los grupos) y la factibilidad se mantiene a lo largo de la búsqueda. Un elemento de la metodología tabú que no se ha estudiado tanto como otros es la *oscilación estratégica* (del inglés *strategic oscillation*, SO). Glover y Laguna [15] la describen como:

"*La oscilación estratégica orienta los movimientos en relación a un nivel crítico, identificado por el estado de la construcción o por el intervalo de ciertos valores funcionales. Tal nivel crítico o límite de la oscilación a menudo representa un punto en el que el método pararía. Pero en vez de parar cuando este punto límite se alcanza, la oscilación estratégica modifica las reglas de selección de movimientos para permitir cruzar la región definida por el nivel crítico.*"

El límite que pretendemos cruzar en este trabajo es el definido por la factibilidad de las soluciones encontradas durante la búsqueda. Queremos limitar la búsqueda a aquellas soluciones en las que todos los elementos han sido asignados, pero queremos explorar el espacio de búsqueda que incluye soluciones en las que se han violado las restricciones de cardinalidad de los grupos. Nuestro objetivo es diseñar un mecanismo de oscilación estratégica que se pueda utilizar junto con cualquiera de los métodos constructivos y de mejora descritos anteriormentes. La oscilación entre la factibilidad y la infactibilidad se define por un parámetro entero k cuyo valor se mueve en el rango 0 y k_{max} y que se aplica de la siguiente forma:

$$a_g - k \leq |E_g| \leq b_g + k$$

Esta definición significa que mientras $k > 0$ la búsqueda puede visitar soluciones infactibles por la cardinalidad de los grupos. Para crear el patrón de oscilación, el valor de k se reinicia a 1 después de cada aplicación satisfactoria del método de mejora. En otro caso, k se incrementa por una unidad de la forma descrita en la Figura 5.

Repetir hasta que el tiempo de búsqueda finaliza.

1. Construir una solución inicial factible.
2. Poner $k = 0$ y aplicar un método de mejora. Sea s la solución obtenida como resultado.
3. Poner $k = 1$.
   while $(k \leq k_{max})$
   4. Considerar s' el resultado de aplicar el método de mejora a la solución s considerando $a_g - k \leq |E_g| \leq b_g + k$ para todos los grupos.
   5. Reparar la solución s' si es infactible y aplicar el método de mejora con $k = 0$ (es decir, con $a_g \leq |E_g| \leq b_g$).
   6. Si la solución s' es mejor que s entonces hacer $s = s'$ y poner $k = 1$; en otro caso, $k = k + 1$

Fig. 5. Oscilación estratégica

El procedimiento de reparación consiste en eliminar los elementos de los grupos g para los que $|E_g| > b_g$ y añadir los elementos a los grupos g para los que $|E_g| < a_g$. Los elementos se seleccionan de forma aleatoria y el proceso continúa hasta que la cardinalidad de los grupos es factible. Este es el procedimiento de reparación implementado en LSGA. El paso 1 de la Figura 5 puede realizarse aplicando GC, GC-FULL o GC-Tabú. Además, las soluciones pueden mejorarse (pasos 2, 4 y 5 de la Figura 5) con cualquiera de los métodos de mejora BI, FI, LCW, T-BI, T-FI y T-LCW.

V. Experimentos computacionales

Esta sección describe los experimentos computacionales que hemos realizado para

comprobar la eficacia y eficiencia de los procedimientos propuestos. Todos los métodos se han implementado en Java SE 6 en un esquema con un único hilo de ejecución. Todos los experimentos se han ejecutado en un Intel Core 2 Quad CPU Q8300 con 6 GB de RAM y SO Ubuntu 9.04 64 bits.

Se han utilizado 480 instancias en la experimentación. Este conjunto de instancias de *benchmark*, denominado MDGPLIB, está disponible en http://www.optsicom.es/mdgp. El conjunto se divide en 3 subconjuntos:

1. *RanReal:* Este conjunto consiste en 160 matrices de $M \times M$ en las que los valores de las distancias d_{ij} son números reales generados utilizando una distribución Uniforme $U(0{,}100)$. El número de elementos M, el número de grupos G y los límites de cada grupo a_g y b_g se muestran en la Tabla I. Hay 20 instancias por cada combinación de parámetros (cada fila en la Tabla I), 10 instancias con grupos del mismo tamaño (*equal group size*, EGS) y 10 instancias con grupos de diferente tamaño (*different group size*, DGS). En las instancias EGS, todos los grupos son del mismo tamaño y éste se calcula como $a_g = b_g = M/G$. En las instancias DGS, los limites de cada grupo (a_g y b_g) para cada instancia se generan de forma aleatoria en el intervalo indicado. Es decir, el valor de a_g se genera en el intervalo $[a_g^{min}, a_g^{max}]$ y el valor de b_g se genera en el intervalo $[b_g^{min}, b_g^{max}]$. Este conjunto de datos lo introdujeron Fan y colaboradores [11] con M con valores entre 10 y 240. Para el desarrollo de este trabajo se han generado instancias de mayor tamaño con $M = 480$ y $M = 960$.
2. *RanInt:* Este conjunto de datos tiene la misma estructura y tamaño que *RanReal* (como se muestra en la Tabla I), pero las distancias se generan con una distribución Uniforme entera $U(0{,}100)$.
3. *Geo:* Este conjunto sigue la misma estructura y tamaño que los dos anteriores. La diferencia radica en que los valores de d_{ij} se calculan como distancias Euclídeas entre pares de puntos con coordenadas generadas aleatoriamente en $[0{,}10]$. El número de coordenadas de cada punto se genera aleatoriamente en el rango $[2, 21]$. Glover y colaboradores [16] introdujeron este generador para el MDP.

Hemos llevado a cabo una serie de experimentos preliminares para determinar los valores más adecuados para algunos parámetros de la búsqueda. En cada experimento, se calcula, para cada instancia, el valor de la mejor solución, *BestValue*, obtenido por la ejecución de todos los métodos considerados. Posteriormente, para cada método, se calcula la desviación relativa (en porcentaje) entre el valor de la mejor solución encontrada por un método y el *BestValue*. En las tablas se muestra la media de esa desviación relativa (*Dev*) para todas las instancias consideradas en cada experimento particular. También se muestra el número de instancias (*#Best*) para las que un determinado método ha obtenido el mejor valor.

Con el objetivo de determinar cuál es el mejor método de los propuestos se ha empleado un conjunto de instancias de entrenamiento. Este conjunto está formado por 10 instancias (5 EGS y 5 DGS) de cada subconjunto de instancias con M=10 hasta $M = 240$. Por lo tanto, el conjunto de entrenamiento tiene un total de 180 instancias, 60 de tipo *RanReal*, 60 de tipo *RanInt* y 60 de tipo *Geo*. El primer experimento se realiza con el objetivo de determinar los mejores valores para los parámetros de búsqueda $tabuTenure$, $maxIter$ y k_{max} y para elegir el mejor método constructivo y de mejora.

Todos los métodos finalizan su ejecución utilizando un tiempo límite que varía en función del tamaño de la instancia. En concreto, para instancias con $M \leq 60$, el tiempo es 1 segundo; $M = 120$, 3 seg; $M = 240$, 20 seg; $M = 480$, 120 seg y $M = 960$, 600 seg.

TABLA I
PARÁMETROS USADO PARA GENERAR LAS INSTANCIAS DE TEST DEL PROBLEMA

M	G	EGS	DGS			
		$a_g = b_g$	a_g^{min}	a_g^{max}	b_g^{min}	b_g^{max}
10	2	5	3	5	5	7
12	4	3	2	3	3	5
30	5	6	5	6	6	10
60	6	10	7	10	10	14
120	10	12	8	12	12	16
240	12	20	15	20	20	25
480	20	24	18	24	24	30
960	24	40	32	40	40	48

El objetivo del primer experimento preliminar consiste en identificar la mejor combinación de método constructivo y método de mejora. Concretamente, se ha acoplado el constructivo WJ y las tres variantes de los constructivos propuestos (GC, GC-FULL y GC-Tabu) con los métodos de mejora BI, FI y LCW y sus variantes tabú (T-BI, T-FI y T-LCW). Hemos fijado los parámetros $tabuTenure$ y $maxIter$ a $0.1M$ y $0.5M$ respectivamente basándonos en los resultados obtenidos en experimentos preliminares (no mostrados aquí por restricciones de espacio). La Tabla II resume los resultados, donde los métodos de construcción están en filas y los procedimientos de mejora están en columnas. Los resultados en negrita identifican la mejor combinación de métodos

de construcción y mejora con y sin estructuras de memoria.

Los datos de la Tabla II muestran que el mejor resultado se obtiene con el método constructivo GC acoplado con el método de mejora T-LCW. Esta combinación obtiene la menor desviación media (0,13%) y el mayor número de mejores soluciones (103) comparada con las demás combinaciones del experimento. Además, los resultados muestran que la versión tabú de los métodos de mejora es más efectiva que las versiones sencillas, independientemente del método de construcción utilizado. Por ejemplo, GC+LCW obtiene una desviación media en porcentaje de 0,84% mientras que para GC+T-LCW esa desviación es de sólo 0,13%.

El segundo experimento preliminar estudia el efecto del valor de k_{max} en el método de oscilación estratégica. En particular, basándose en los resultados de los experimentos previos, se ha construido el procedimiento de oscilación estratégica con el método constructivo GC y el procedimiento de mejora T-LCW. Por simplicidad, nos referimos a este procedimiento completo como SO.

TABLA II
COMPARATIVA DE LOS MÉTODOS MULTI-ARRANQUE PROPUESTOS Y DEL ESTADO DEL ARTE

	LCW	BI	FI	T-LCW	T-BI	T-FI
GC						
Dev (%)	**0,84**	0,90	0,89	**0,13**	0,13	0,14
#Best	**82**	81	82	**103**	100	99
GC-FULL						
Dev (%)	0,86	0,88	0,87	0,13	0,13	0,15
#Best	79	82	84	99	100	97
GC-Tabu						
Dev (%)	0,88	0,91	0,90	0,14	0,15	0,16
#Best	79	79	78	97	97	94
WJ						
Dev (%)	4,31	4,30	3,69	0,62	0,55	0,20
#Best	26	28	49	57	58	94

Hemos evaluado el comportamiento de SO con valores de $k_{max} = 1, \dots, 6$ ejecutando SO con dichos valores sobre el conjunto de instancias de entrenamiento. Los resultados muestran que SO con $k_{max} = 4$ obtiene mejores soluciones que con los otros valores del parámetro. Por tanto, usaremos ese valor para el resto de los experimentos de este trabajo.

La experimentación preliminar para ajustar los valores de los parámetros se ha llevado a cabo utilizado 180 instancias. Para evaluar el rendimiento de los métodos propuestos y comparar su comportamiento con los mejores métodos del estado del arte, hemos usado 240 instancias no usadas previamente para la calibración. En concreto, el conjunto está formado por 5 EGS y 5 DGS por cada tipo (*RanReal*, *RanInt* y *Geo*) con $M = 10$ hasta 960.

Los métodos que se comparan en los experimentos finales son:

- LSGA: Fan et al. (2011).
- LCW: Método multiarranque con construcción aleatoria y método de mejora LCW. Weitz y Lakshminarayanan [6].
- T-LCW: Método multiarranque con método constructivo GC y método de mejora T-LCW with $tabuTenure = 0.1M$ and $maxIter = 0.5M$.
- SO: Método multiarranque basado en la oscilación estratégica con método constructivo GC, método de mejora T-LCW y $k_{max} = 4$.

Los resultados de este experimento se presentan en las Tablas III y IV. La Tabla III resume los resultados del experimento completo. Los resultados muestran que los métodos T-LCW y SO mejoran a los procedimientos existentes. Es evidente también que las diferencias de rendimiento entre T-LCW y SO indican la ventaja de los mecanismos de búsqueda incluidos en la implementación de SO.

TABLA III
COMPARATIVA DE LOS MÉTODOS PROPUESTOS Y DEL ESTADO DEL ARTE

Método	*Dev*	*#Best*
LSGA	0,61%	82
LCW	1,01%	80
T-LCW	0,17%	141
SO	0,04%	192

Para proporcionar más información de la que aparece en la Tabla III, hemos calculado los valores de las métricas *Dev* y *#Best* dependiendo del tamaño de las instancias. Estos resultados aparecen en la Tabla IV y podemos observar que las diferencias de rendimiento entre los métodos previos (LSGA y LCW) y los métodos propuestos (T-LCW y SO) se incrementan con el tamaño de las instancias. Por ejemplo, en el subconjunto con instancias más grandes ($M = 480$ & 960), SO obtiene unos resultados un orden de magnitud mejores que los métodos del estado del arte en el estadístico *Dev*.

Con el objetivo de apoyar nuestras conclusiones sobre el rendimiento de los métodos propuestos, se han realizado tres análisis estadísticos. Primero, se ha aplicado el *test no-paramétrico de Friedman* para múltiples muestras correlacionadas con las mejores

soluciones obtenidas por cada uno de los 4 métodos. Este test calcula, para cada instancia, el valor de $rank$ de cada método en función del valor de la solución obtenida por él (donde $rank = 4$ se asigna al mejor método y $rank = 1$ al peor). Luego se calcula el valor medio de los valores de $rank$ para cada método en todas las instancias. Si las medias son suficientemente diferentes, el p-valor asociado o nivel de significancia es pequeño. El p-valor de 0,000 obtenido en este experimento indica claramente que hay diferencias estadísticamente significativas entre los 4 métodos. Los valores de $rank$ producidos por este test son 3,23 (SO), 3,02 (T-LCW), 2,08 (LSGA) y 1,67 (LCW).

TABLA IV
COMPARATIVA DE LOS MÉTODOS PROPUESTOS Y DEL ESTADO DEL ARTE AGRUPADOS POR TAMAÑO DE LA INSTANCIA

Tamaño de Instancia	Método	*Dev*	*#Best*
$M \leq 60$ (120 instancias)	LSGA	0,08%	82
	LCW	0,28%	80
	T-LCW	0,01%	107
	SO	0,01%	106
$M = 120$ & 240 (60 instancias)	LSGA	1,04%	0
	LCW	1,91%	0
	T-LCW	0,30%	19
	SO	0,08%	41
$M = 480$ & 960 (60 instancias)	LSGA	1,25%	0
	LCW	1,58%	0
	T-LCW	0,34%	15
	SO	0,06%	45

En segundo lugar, se ha empleado el *test de Wilcoxon* y el *test del Signo* para realizar comparaciones entre pares de SO y T-LCW, que consistentemente proporcionan los mejores valores en los experimentos. Los resultados del test de Wilcoxon (con un p-valor de 0,000) determinan que las soluciones obtenidas por los dos métodos representan dos poblaciones diferentes. El test del Signo (con un p-valor de 0,000) indica que las soluciones obtenidas con SO son mejores que las obtenidas con T-LCW.

VI. CONCLUSIONES

El Problema de Maximización de la Diversidad en Grupos (MDGP) es un problema de optimización difícil y una plataforma perfecta para estudiar la efectividad de diversos mecanismos de búsqueda. Es especialmente interesante el estudio que hemos realizado sobre la expansión de la vecindad de una solución, incluyendo movimientos adicionales y permitiendo a los procedimientos de búsqueda visitar soluciones infactibles. A través de una extensa experimentación, se han podido determinar los beneficios de añadir estrategias mejoradas de búsqueda a los procedimientos básicos. Hemos añadido estos mecanismos de forma secuencial, lo que nos ha permitido medir sus efectos y estudiar las combinaciones con las que se obtienen mejores resultados. Creemos que las conclusiones extraídas de nuestro estudio se pueden trasladar a otras configuraciones y esto ayudará al desarrollo de búsquedas más robustas en espacios combinatorios.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el *Ministerio de Educación y Ciencia* (Ref. TIN2009-07516).

REFERENCIAS

[1] Feo, T. and M. Khellaf (1990) "A class of bounded approximation algorithms for graph partitioning," Networks, vol. 20, pp. 181-195.
[2] Feo, T., O. Goldschmidt and M. Khellaf (1992) "One-half approximation algorithms for the k-partition problem," Operations Research, 1992, vol. 40, pp. S170-S173.
[3] O'Brien, F. A. and J. Mingers (1995) "The equitable partitioning problem: a heuristic algorithm applied to the allocation of university student accommodation," Warwick Business School, Research Paper no. 187.
[4] Kral, J. (1965) "To the problem of segmentation of a program," Information Processing Machines, vol. 2, pp. 116-127.
[5] Weitz, R. R. and M. T. Jelassi (1992) "Assigning students to groups: a multi-criteria decision support system approach," Decision Sciences, vol. 23, no. 3, pp. 746-757.
[6] Weitz, R. R. and S. Lakshminarayanan (1998) "An empirical comparison of heuristic methods for creating maximally diverse groups," Journal of the Operational Research Society, vol. 49, no. 6, pp. 635-646.
[7] Hettich S. and M. J. Pazzani (2006) "Mining for element reviewers: Lessons learned at the national science foundation," In: Proceedings of the KDD'06, ACM: New York, NY, pp. 862–871.
[8] Arani, T. and V. Lotfi (1989) "A three phased approach to final exam scheduling," IIE Transactions, vol. 21, pp. 86-96.
[9] Weitz, R. R. and S. Lakshminarayanan (1996) "On a heuristic for the final exam scheduling problem," Journal of the Operational Research Society, vol. 47, no. 4, pp. 599-600.
[10] Lotfi V. and R. Cerveny (1991) "A final exam scheduling package," Journal of the Operational Research Society, vol. 42, pp. 205-216.
[11] Fan, Z. P., Y. Chen, J. Ma and S. Zeng (2011) "A hybrid genetic algorithmic approach to the maximally diverse grouping problem", Journal of the Operational Research Society, vol. 62, pp. 92-99.
[12] Chen, Y., Z. P. Fan, J. Ma, S. Zeng (2011) "A hybrid grouping genetic algorithm for reviewer group construction problem", Expert Systems with Applications, vol. 38, pp. 2401-2411.
[13] Falkenauer, E. (1998) Genetic Algorithms for Grouping Problems, Wiley: New York.
[14] Mingers, J. and F. A. O'Brien (1995) "Creating students groups with similar characteristics: a heuristic approach," Omega, vol. 23, pp. 313-321.
[15] Glover, F. and Laguna, M. (1997) Tabu Search, Kluwer Academic Publisher: Boston.
[16] Glover, F., C. C. Kuo and K. S. Dhir (1998) "Heuristic algorithms for the maximum diversity problem," Journal of Information and Optimization Sciences, vol. 19, no. 1, pp. 109–132.

Análisis Empírico del *No Free Lunch* en Problemas Binarios Reales

Carlos García-Martínez, Francisco J. Rodriguez, Manuel Lozano

Resumen— **Los teoremas del *No Free Lunch* para optimización (*no hay comida gratis*, en español) establecen que el rendimiento de cualquier par de algoritmos es equivalente cuando se considera el rendimiento medio sobre el conjunto completo de posibles problemas. Sin embargo, estudios más recientes sugieren que los algoritmos pueden ofrecer rendimientos diferentes cuando se consideran únicamente el conjunto de problemas reales, es decir, aquellos con interés práctico. En este trabajo, hacemos la suposición de que los problemas binarios que se han tratado en la literatura científica representan, al menos mínimamente, los verdaderos problemas binarios de interés en el mundo real. A partir de ahí, analizamos el rendimiento medio de varios algoritmos en subconjuntos de este banco de pruebas. En particular, consideramos problemas de optimización combinatoria estáticos sin restricciones de un solo objetivo y un solo jugador y cuyas soluciones pueden codificarse directamente como vectores de variables binarias. Nuestros resultados muestran que existe evidencia empírica para rechazar los teoremas del No Free Lunch en el conjunto de problemas binarios del mundo real.**

Palabras clave— **No free lunch, optimización binaria, estudio empírico, problemas reales**

I. Introducción

En 1997, Wolpert y Macready presentaron los teoremas para optimización del No Free Lunch (NFL) [1], los cuales establecen, en pocas palabras, que todos los algoritmos tienen un rendimiento equivalente cuando se consideran todos los posibles problemas, o más concretamente, conjuntos de problemas cerrados bajo permutaciones (c.b.p) [2] (Es posible reducir aún más el conjunto de problemas cuando se considera un conjunto concreto de algoritmos [3]). Es importante destacar que en este contexto se entiende por algoritmo cualquier procedimiento que realiza un muestreo sin reemplazamiento de las soluciones candidatas al problema. Este resultado impactó profundamente a investigadores que pretendían diseñar procedimientos eficientes de propósito general para problemas de optimización, porque se demostraba que no serían mejores que la búsqueda aleatoria sin reemplazamiento. Recientemente, Marshall y Hinton han demostrado que cuando se consideran procedimientos que pueden visitar soluciones más de una vez los teoremas del NFL no pueden aplicarse, sin embargo, aún existe una elevada probabilidad de que el rendimiento de cualquier algoritmo no sea superior al de la búsqueda aleatoria con o sin reemplazamiento [4].

En 2003, Igel y Toussaint mostraron que los teoremas del NFL posiblemente no tienen validez cuando se consideran conjuntos de problemas de interés práctico [5]. En particular, demostraron que los conjuntos de problemas a los que se les pueden asociar restricciones basadas en relaciones de similitud no triviales entre soluciones no son c.b.p. Según los autores, este tipo de restricciones son muy comunes en problemas de interés práctico, lo cual permite concebir procedimientos de propósito general para la optimización de problemas reales. En 2010, Jiang y Chen aportaron resultados empíricos en favor de las afirmaciones de Igel y Toussaint [6] (aunque el trabajo no se citaba), considerando una clase específica de funciones matemáticamente definidas que incorporaban restricciones basadas en la similitud de las soluciones. Sin embargo, la asunción de que los problemas reales incorporaban las mencionadas restricciones se mantenía aún sin demostrar.

En este trabajo, pretendemos aportar evidencias empíricas a favor de las afirmaciones de Igel y Toussaint en el conjunto específico de problemas con interés en el mundo real. Para ello, hacemos la siguiente suposición: los problemas tratados en la literatura científica representan, superando al menos un umbral mínimo, a los problemas del mundo real. En caso contrario, debe entenderse que los estudios científicos correspondientes habrían sido inútiles. A partir de ahí, seleccionamos un conjunto extenso, potencialmente infinito, de instancias de problemas binarios representativo de los que se han tratado en la literatura, y para el cual no se conoce si el NFL se cumple o no (entiéndase que el conjunto completo de posibles problemas binarios sí es c.b.p). Finalmente, realizamos un estudio empírico comparando varios algoritmos en el banco de problemas considerado con la intención de determinar si existen o no diferencias de rendimiento significativas entre ellos.

El resto del trabajo se estructura de la siguiente forma. En la Sección II, describimos el marco experimental bajo el cual desarrollamos nuestro estudio. En la Sección III, presentamos los resultados del estudio. En la Sección IV interpretamos las implicaciones de los resultados obtenidos. Finalmente, la Sección V ofrece las conclusiones del trabajo.

II. Marco Experimental

En esta sección, definimos el marco experimental en el que se realizan los experimentos. En la Sec-

Dpto. Informática y Análisis Numérico. Universidad de Córdoba. E-mail: cgarcia@uco.es .

Dpto. Ciencias de la Computación e I.A (CITIC-UGR). Universidad de Granada. E-mail: fjrodriguez@decsai.ugr.es .

Dpto. Ciencias de la Computación e I.A (CITIC-UGR). Universidad de Granada. E-mail: lozano@decsai.ugr.es .