Algoritmos Voraces Iterativos para el Problema de Agrupación de Máxima Diversidad

Jonathan D. González-Barrera(*), Francisco J. Rodríguez y Manuel Lozano(**)

Resumen— **El problema de agrupación de máxima diversidad consiste en encontrar una colección de grupos a partir de un conjunto de elementos que maximice la relación entre los elementos que componen cada grupo. Para abordar este problema, proponemos la utilización de una metaheurística voraz iterativa por dos razones fundamentales. Primero, por su flexibilidad y simpleza; y segundo, por la eficacia mostrada en otros problemas de combinatoria similares al tratado.**

Palabras clave— **Problema de agrupación de máxima diversidad, algoritmo voraz iterado, búsqueda local.**

I. Introducción

El problema de agrupación de diversidad máxima (en inglés, *maximally diverse grouping problem* - MDGP) consiste en formar grupos disjuntos de diversidad máxima (de igual o diferente tamaño) a partir de un conjunto de elementos. Dicha diversidad se calcula sumando las distancias entre los pares de elementos que pertenecen a los mismos grupos. Estas distancias dependerán del contexto de aplicación en el que se vaya a utilizar. El objetivo final del problema es maximizar la diversidad conjunta, es decir, maximizar la suma de la diversidad de todos los grupos. Feo y Khellaf [3] demostraron que este problema combinatorio es *NP-Completo*. En la literatura, el MDGP se ha presentado bajo distintos nombres: *problema de k-partición*[3][4] y *problema de particionamiento equitativo*[10].

Con el fin de formular el MDGP en términos matemáticos, asumimos que cada elemento se puede representar por un conjunto de atributos. Sea a_{ik} el valor del elemento *i-ésimo* que tiene asociado el *k-ésimo* atributo, donde $i = 1, \ldots, n$ y $k = 1, \ldots, t$ (n el número de elementos y t el número de atributos). Entonces, la distancia d_{ij} entre el *i-ésimo* elemento y el *j-ésimo* elemento viene dada por la distancia euclidiana:

$$d_{ij} = \sqrt{\sum_{k=1}^{t}(a_{ik} - a_{jk})^2}.$$

(*)Jonathan D. González-Barrera doctorando del Dept. de Estadistica, Invesgación Operativa y Ciencias de la Computación, Facultad de Matemáticas, Univ. de La Laguna 38071, San Cristobal de La Laguna, E-mail: alu0100116090@ull.edu.es

(**)F.J. Rodríguez y M. Lozano pertenecen al Dept. de Ciencias de la Computación e I.A., E.T.S de Ingenierías Informatica y de Telecomunicación, Univ. de Granada 18071, Granada, E-mail: fjrodriguez@decsai.ugr.es, lozano@decsai.ugr.es

Dicha distancia cumple la propiedad simétrica, es decir, que $d_{ij} = d_{ji}$, $\forall i, j \in N$; además, $d_{ii} = 0$, $\forall i \in N$.

Sea $X = \{x_1, \ldots x_n\}$ un conjunto de elementos ($n \in N$). El MDGP consiste en encontrar una partición de X, $G = \{G_1, \ldots, G_m\}$ ($G_1 \cup \ldots \cup G_m = X$ con $m \in N$, $G_i \cap G_j = \emptyset$ con $i \neq j$) con $2 \leq m < n$, que maximiza la diversidad intra grupos. La formulación del MDGP como problema de *programación entera cuadrática* viene dado por [5]:

$$\text{máx} \quad Z = \sum_{g=1}^{m}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_{ig}x_{jg}$$

$$s.a: \quad \sum_{g=1}^{m} x_{ig} = 1\,, \qquad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ig} \geq L_g\,, \qquad \forall g = 1, \ldots, m$$

$$\sum_{i=1}^{n} x_{ig} \leq H_g\,, \qquad \forall g = 1, \ldots, m$$

$$x_{ig} \in \{0, 1\}\,, \qquad \forall i = 1, \ldots, n \quad \forall g = 1, \ldots, m$$

donde x_{ig} es una variable binaria que toma el valor 1 si el *i-ésimo* elemento está asignado a G_g y 0 en otro caso; mientras que (L_g, H_g) es el intervalo que restringe el tamaño del *g-ésimo* grupo ($g = 1, \ldots, m$). Nótese que

$$\sum_{g=1}^{m} L_g \leq n \leq \sum_{g=1}^{m} H_g$$

y que

$$\sum_{i=1}^{n}\sum_{g=1}^{m} x_{ig} = n, \quad \forall x_{ig} \in \{0, 1\}.$$

En el caso que $L_g = H_g = n/m$ ($\forall g = 1, \ldots, m$), obtendremos el modelo propuesto en [13][14], el cual asignará el mismo número de elementos a cada grupo.

El *algoritmo voraz iterativo* [11] (en inglés, *iterated greedy*, IG) es una metaheurística desarrollada recientemente que sigue unos principios muy simples, es fácil de implementar y muestra un comportamiento bastante efectivo. De hecho, es un algoritmo esta-

do del arte para un considerable número de problemas [11][12]. El algoritmo IG, iterativamente, trata de mejorar una solución eliminando, en primer lugar, elementos de la misma para después reconstruir la solución mediante un procedimiento heurístico de construcción voraz.

En este artículo, proponemos la utilización de IG para abordar el MDGP. El estudio se estructura de la siguiente manera. En la sección II, se da una visión general de las metaheurísticas presentadas anteriormente para tratar el MDGP. En la sección III, se describe la propuesta IG. En la sección IV, se presenta el estudio empírico realizado tanto para analizar la robustez de nuestra propuesta, como para comparar sus resultados con otros enfoques de la literatura. Para finalizar, en la sección V, se discuten las conclusiones y las futuras líneas de investigación.

II. Metaheurísticas para el MDGP

A continuación, describimos las metaheurísticas más relevantes presentadas para el MDGP en la literatura.

En [3], Feo y Khellaf proponen varias heurísticas basadas en la teoría de grafos para los casos de grupos de tamaño impar y de tamaño par. En [4], se propone un algoritmo para grupos de tamaño del orden de 2^n, con n el cardinal del conjunto de nodos. También proponen el método *match & contract*, mediante el que abordan los problemas de *k-partición* cuando $k = n/3$ y $k = n/4$.

En [13] y [14], Weitz y Lakshminarayanan realizan un examen exhaustivo de todas las heurísticas propuestas para el MDGP. Además, proponen el método LCW (sigas de *Lofti-Cerveny-Weitz*), una versión modificada del algoritmo LC (sigas de *Lofti-Ceveny*), presentado por Lofti [8]. LCW no es más que una búsqueda local que trabaja sobre la construcción previa de una solución.

La propuesta más reciente para resolver el MDGP, y actual estado del arte, es un algoritmo memético que combina un algoritmo genético y un procedimiento de búsqueda local (GA+LS) [5]. El GA utiliza una codificación específica desarrollada para problemas de agrupación. A cada gen se le asigna un número de grupo que indica que el elemento correspondiente pertenece a dicho grupo. Por su parte, la LS es un procedimiento iterativo en el que, en cada iteración, se examinan todas las soluciones vecinas. Tras la comprobación, se selecciona la mejor de ellas y se reemplaza la solución actual. Esta nueva solución se obtiene del intercambio entre dos elementos de diferentes grupos [1]. En lo referido a la experimentación, los autores la dividen en dos casos: cuando el tamaño de los grupos es fijo e igual, comparando GA+LS con LCW; y cuando el tamaño de los grupos es variable y fluctúa en un intervalo, comparando GA+LS con una versión del mismo pero sin búsqueda local.

III. Algoritmo Voraz Iterado para el MDGP

El algoritmo IG es una metaheurística sencilla y eficiente, de fácil implementación y, con excelente rendimiento, que se ha empleado sobre numerosos problemas de optimización combinatoria [6][7][9][11]. El algoritmo IG trata de refinar, en cada iteración, la solución actual a través de tres fases: *destrucción*, *reconstrucción* y *criterio de aceptación*. En la fase de *destrucción*, algunos elementos de la solución actual son eliminados. A continuación, en la fase de *reconstrucción*, la solución parcial resultante se completa utilizando un procedimiento heurístico de construcción voraz. Una vez obtenida la nueva solución, un *criterio de aceptación* decide si esta solución reemplaza a la solución actual o no. El proceso se repite hasta que se verifiquen las condiciones de parada. De manera opcional, a la nueva solución generada en la etapa de reconstrucción se le puede aplicar un método de mejora, utilizando una técnica de *búsqueda local*.

En esta sección proponemos una metaheurística IG, a la que llamaremos IG-MDPG, para resolver el MDGP. En la sección III-A, explicaremos la propuesta IG dentro del marco del MDGP, centrándonos en la adaptación de ésta al problema. En la sección III-B, presentamos distintos conceptos que ayudarán a entender el funcionamiento de IG-MDGP y el *algoritmo voraz* que intervendrá en las fases de *inicialización* y *reconstrucción* de nuestro algoritmo. En la sección III-C, explicamos la *fase de inicialización*. En la sección III-D y III-E, presentamos la fase de destrucción y reconstrucción para IG-MDGP. En la sección III-F, tratamos las propuestas de búsquedas locales que intervendrán en el mismo. Para finalizar, en la sección III-G, abordamos los criterios de aceptación seleccionados por su simpleza y efectividad. También enunciamos los criterios de paradas.

A. IG para el MDGP

IG-MDGP (Figura 1) comienza su ejecución considerando una solución inicial completa SS (Paso 1, fase de inicialización voraz) e itera en un bucle principal donde, en primer lugar, se obtiene una solución parcial S^{de}, eliminando un número fijo de componentes de la solución SS (Paso 3, fase de destrucción). Posteriormente, la solución parcial S^{de} se reconstruye obteniendo así la solución completa S^{rc} (Paso 4, fase de reconstrucción). Para intentar mejorar la solución completa S^{rc} se utiliza un procedimiento de *búsqueda local* que consta de dos etapas (Paso 5, fase BL-Movimientos y Paso 6, fase BL-Intercambios). Estas etapas, aplicadas de manera consecutiva, generan la solución S^{ic}. Antes de pasar a la siguiente iteración, el criterio de aceptación decide si la solución S^{ic} se convierte en la nueva solución actual (Paso 7). El proceso se repite hasta cumplir la condición de parada. La mejor solución SS encontrada durante todo el proceso de búsqueda, se devuelve

como resultado del algoritmo.

```
Procedimiento IG-MDGP
1:    SS ← Inicialización-Voráz ()
2:    repetir
3:        S^de ← Destrucción (SS)
4:        S^rc ← Reconstrucción (S^de)
5:        S^m ← BL-Movimientos (S^rc)
6:        S^ic ← BL-Intercambios (S^m)
7:        SS ← Criterio-Aceptación (SS, S^ic)
8:    hasta (alcanzar condición de parada)
```

Fig. 1. Pseudocódigo de IG para el problema MDGP

B. Algoritmo Voraz para el MDGP

En esta sección, presentamos, en primer lugar, una serie de conceptos previos que facilitan la comprensión del algoritmo IG-MDGP. Además, presentamos el algoritmo voraz que desempeña un papel primordial dentro de nuestra propuesta.

Primero, resaltar que una solución para el MDGP es una partición de X. También decir que, algunos momentos, nuestro algoritmo IG-MDGP maneja *soluciones parciales* (fase de *destrucción*). Definimos dichas soluciones como

$$G^P = \{G_1^P, \ldots, G_m^P\} \quad con \quad \bigcup_{g=1}^{m} G_g^P \subset X.$$

A continuación, definimos un grupo al que llamaremos *grupo auxiliar* (C_e) como

$$C_e = X \setminus \bigcup_{g=1}^{m} G_g^P.$$

Este grupo almacenará los elementos eliminados en la fase de *destrucción*. De la misma manera, proporcionará los elementos necesarios en la fase de *reconstrucción*.

Finalmente, precisamos las diferentes medidas de contribución. La *contribución* de un elemento x_i al grupo que al pertenece G_g la definiremos como

$$D(x_i; G_g) = \sum_{x_j \in G_g} d(x_i, x_j),$$

donde $d(x_i, x_j) = d_{ij}$ y $x_i \in G_g$ para $i = 1, \ldots, n$ y $g = 1, \ldots, m$. Entonces, la *contribución* que aporta el grupo *g-ésimo* a la solución es

$$\Delta(G_g) = \frac{1}{2} \sum_{x_i \in G_g} D(x_i; G_g), \quad g = 1, \ldots m.$$

De aquí, podemos calcular el *valor de la función objetivo* de una solución z:

$$z = \sum_{g=1}^{m} \Delta(G_g).$$

Ahora, describiremos el algoritmo voraz propuesto para este problema. Esta propuesta parte de una solución parcial y añade un elemento que se caracteriza por ser el que produce una mejora más elevada en la solución parcial. En particular, el algoritmo selecciona un grupo G_g al azar entre los grupos cuyo tamaño es menor que L_g. Si todos los grupos tienen un tamaño mayor o igual que L_g, seleccionamos aleatoriamente un grupo entre los que su tamaño sea menor que H_g. A continuación, escoge el elemento x_e perteneciente a C_e cuya contribución a la diversidad de G_g sea máxima,

$$x_e = arg \left\{ \max_{x_i \in C_e} D(x_i; G_g) \right\},$$

y lo añade a la solución parcial.

C. Fase de Inicialización (paso 1)

IG-MDGP comienza con una solución inicial completa (SS) que se construye utilizando la *inicialización voraz* (Paso 1, Figura 1). En el proceso de inicialización voraz se selecciona aleatoriamente m elementos y se añaden a los m grupos (uno por grupo). A continuación, se aplica el algoritmo voraz (III-B) para completar los grupos. Así, se obtiene una solución inicial completa.

D. Fase de Destrucción (paso 3)

En la fase de *destrucción* se parte de una solución inicial (SS) y, de manera iterativa, se elimina una proporción de elementos (n_d) de forma aleatoria. Los elementos eliminados se añaden al conjunto C_e. Así, obtendremos una solución parcial actual S^{de} con

$$S^{de} = \{G_1^P, \ldots, G_m^P\}$$

La cantidad de elementos eliminados en la esta fase depende del valor del parámetro $n_d \in [0, 1]$. En concreto, se eliminan $N_d = \lfloor n_d \times n \rfloor$ elementos.

E. Fase de Reconstrucción (paso 4)

A partir de una solución parcial S^{de}, en la fase de *reconstrucción* de IG-MDGP (Paso 4, Figura 1), el algoritmo voraz construye una nueva solución completa S^{rc} con

$$S^{rc} = \{G_1', \ldots, G_m'\} :$$

$$G_g' = G_g^P \cup \{x_e^{(1)}, \ldots, x_e^{(k_g)}\} \quad \wedge \quad |G_g'| \in [L_g, H_g],$$

con $g = 1, \ldots, m$ y k_g el número de elementos seleccionados de C_e para el grupo G_g por el algoritmo voraz.

F. Fase de Búsqueda Local (pasos 5 y 6)

La utilización de una técnica de búsqueda local eficiente y rápida puede contribuir a mejorar las soluciones obtenidas tras las fases de destrucción y reconstrucción del IG. Por ello, la propuesta de IG incluye dos técnicas de búsqueda local que se ejecutan de manera consecutiva. La primera de ellas está basada en *movimientos* que fuerzan a determinados elementos a abandonar un grupo para incorporarse en otro distinto, siempre que no se violen

las limitaciones de tamaño de los mismos. La otra técnica de búsqueda local básicamente produce *intercambios* para permutar elementos entre grupos.

F.1 Búsqueda Local Basada en Movimientos

Se han analizado dos estrategias diferentes de búsqueda local basada en movimientos (Paso 5 en Figura 1):

- *Primer-Movimiento (mov-P).* Consiste en encontrar el primer movimiento que produce una mejora de la función objetivo.
- *Mejor-Movimiento (mov-M).* Lleva a cabo el movimiento que produce la mayor mejora entre todos los movimientos posibles.

Estos operadores se ejecutarán hasta que no se pueda mejorar el valor de la función objetivo.

Tras cada movimiento, no es necesario volver a evaluar la nueva solución completa. Para esto, sólo es necesario actualizar el valor de la contribución de los grupos que intervienen en el movimiento de elementos. Sea G_k un grupo e y un elemento de éste. Sea G_t el grupo donde se va a mover y. Para calcular la función objetivo de la nueva solución, restamos la contribución del elemento y al grupo G_k y la sumamos a la del grupo G_t, de la siguiente forma:

$$\Delta(G'_k) = \Delta(G_k) - D(y; G_k),$$

$$\Delta(G'_t) = \Delta(G_t) + D(y; G_t).$$

Esto afectará al nuevo valor de la función objetivo (z') de la siguiente manera:

$$z' = z - [\Delta(G_k) + \Delta(G_t)] + [\Delta(G'_k) + \Delta(G'_t)] =$$

$$= z - D(y; G_k) + D(y; G_t).$$

Después de cada movimiento, el valor de las contribuciones de los elementos que forman cada grupo han variado y pueden calcularse como

$$D(x_i; G'_k) = D(x_i; G_k) - d(x_i, y),$$

$$D(x_j; G'_t) = D(x_j; G_t) + d(x_j, y),$$

con $i \neq j$, $x_i \in G'_k$ y $x_j \in G'_t$. Esta estrategia aprovecha las contribuciones anteriores para calcular las nuevas. Estos cálculos se utilizan tanto en la BL basada en *Primer-Movimiento* como en la BL basada en *Mejor-Movimiento*.

F.2 Búsqueda Local Basada en Intercambios

Hemos considerado tres procedimientos de búsqueda local basada en *intercambios* (Paso 6, Figura 1):

- *Primer-Intercambio (int-P).* Se selecciona y aplica el primer intercambio que produzca una mejora en la función objetivo. En particular, se intercambian dos elementos de dos grupos diferentes siempre y cuando se produzca una mejora en la función objetivo de la solución a refinar.
- *Mejor-Intercambio (int-M).* En este caso, se elige el intercambio que aporta una mayor ganancia en la función objetivo de todos los intercambios posibles.
- *Primer-Intercambio-Rápido (int-RP).* Proponemos una alternativa que pretende reducir el tiempo de ejecución de las técnicas anteriores. Se elige el elemento que menos aporte a su grupo entre todos los elementos, y se practica una búsqueda local basada en Primer-Intercambio. Si no se produce mejora, se selecciona el siguiente elemento que menos aporte a su grupo entre todos los elementos.

Estos algoritmos se ejecutan hasta que no sea posible encontrar mejoras en la función objetivo.

Como en la sección III-F.1, vamos a especificar los pasos a seguir para agilizar el cálculo de la función objetivo después de un intercambio. Sean G_k y G_t dos grupos con k y t fijos. Sean x e y dos elementos de G_k y G_t, respectivamente. La contribución que se produce al intercambiar los elementos x e y entre los grupos k y t se calcula de la siguiente manera:

$$\Delta(G'_k) = \Delta(G_k) - D(x; G_k) + D(y; G_k) - d(x, y),$$

$$\Delta(G'_t) = \Delta(G_t) - D(y; G_t) + D(x; G_t) - d(x, y),$$

y el valor de la función objetivo (z') se ve afectado de la siguiente forma:

$$z' = z - [\Delta(G_k) + \Delta(G_t)] + [\Delta(G'_k) + \Delta(G'_t)] =$$

$$= z - [D(x; G_k) - D(y; G_k)] + [D(x; G_t) - D(y; G_t)]$$

$$-2d(x, y).$$

Tras el intercambio, se actualizará el valor de las contribuciones de los restantes elementos a su grupo:

$$D(x_i; G'_k) = D(x_i; G_k) - d(x_i, x) + d(x_i, y),$$

$$D(x_j; G'_t) = D(x_j; G_t) + d(x_j, x) - d(x_j, y),$$

con $i \neq j$, $x_i \in G'_k$ y $x_j \in G'_t$. Estas expresiones permiten desarrollar una estrategia eficiente para la búsqueda de una solución mejorada, reduciendo el coste para calcular las contribuciones de los grupos al valor de la función objetivo.

G. Criterios de Aceptación (paso 7)

A continuación, presentamos tres *criterios de aceptación* (Paso 7, Figura 1) que hemos estudiado como alternativas para el diseño de IG-MDGP:

- *Reemplazar si es mejor* (A1) [12]. La nueva solución se acepta sólo si ésta proporciona un valor para la función objetivo mejor.
- *Reemplazar al azar* (A2). El criterio de aceptación *Reemplazar si es mejor* puede conducir a situaciones de estancamiento de la búsqueda debido a la diversificación insuficiente [11]. Una opción para que esto no suceda es aplicar el criterio de aceptación *reemplazar al azar*. Dicho criterio considera reemplazar la solución actual con la nueva solución generada con una probabilidad de 1/2.
- *Reemplazar siempre* (A3). Con este criterio de aceptación, siempre se acepta la solución generada como nueva solución actual.

IV. Estudio Experimental

En esta sección, presentamos los experimentos realizados para estudiar los distintos parámetros de IG-MDGP y para comparar sus resultados frente al estado del arte. Para comparar los resultados de las diferentes configuraciones y algoritmos, hemos utilizado tests no paramétricos debido a que no requieren ninguna condición previa para su aplicación. Concretamente, los *tests no paramétricos* utilizados en este trabajo son: el test de Friedman que comprueba si existen diferencias significativas entre los algoritmos; y el test de Holm, técnica *post-hoc* que facilitará la disertación del análisis si no se cumple la hipótesis nula del test de Friedman. El nivel de significancia considerado es del 5 %.

A. Marco Experimental

La elección de un marco experimental adecuado es clave para una buena experimentación. Por ello, hemos construido dos marcos experimentales (recordar que n es el número de elementos y m el número de grupos):

- *RanInt*[1]: Son 80 instancias de tamaño $n \times n$ de las cuales 40 serán utilizadas para realizar un análisis experimental de los parámetros de IG-MDGP y las 40 restantes se emplearán para comparar nuestra propuesta frente a LCW y GA+LS. Para cada combinación de n y m hay 10 instancias: 5 con igual tamaño de grupo y 5 de diferente tamaño de grupo. Para cada instancia, la distancia euclidiana d_{ij} ($i < j$) se ha generado mediante una distribución uniforme: $d_{ij} \sim U(0, 100)$ con i, j números naturales.
- Type1_22: Utilizamos las diez primeras instancias *Type1_22* presentadas en [2]. En dichas instancias, la distancia d_{ij} se genera de acuerdo a una distribución uniforme $U(0, 10)$. Hemos considerado las 5 primeras instancias para estudiar los parámetros de IG-MDGP, y las 5 restantes para comparar su comportamiento con respecto al estado del arte (en ambos casos para todos las configuraciones propuestas).

En la Tabla I, podemos encontrar 4 configuraciones correspondientes a 4 combinaciones diferentes del valor de n y de m (*RanInt*). La columna TGI (*Tamaño de Grupo Igual*) especifica el valor de L_g y H_g cuando todos los grupos tienen el mismo tamaño ($L_g = H_g = n/m$). Por su parte, en la columna TGD (*Tamaño de Grupo Diferente*) los valores de L_g y H_g se eligen de forma aleatoria para cada instancia dentro de los intervalos $[L_g^{\text{mín}}, L_g^{\text{máx}}]$ y $[H_g^{\text{mín}}, H_g^{\text{máx}}]$, respectivamente. La columna LTM (*Limite de Tiempo Máximo*) contiene los límites de tiempo en segundos permitido para cada ejecución de cada algoritmo estudiado.

Por otro lado, en la Tabla II, presentamos un resumen de los valores de los parámetros para las instancias *Type1_22*. En la columna TNE (*Tamaño No Equivalente*) muestra los intervalos (L, H) donde

[1] http://www.optsicom.es/mdgp/

TABLA I
Características de las instancias *RanInt*

		TGI	TGD				
n	m	n/m	$L_g^{\text{mín}}$	$L_g^{\text{máx}}$	$H_g^{\text{mín}}$	$H_g^{\text{máx}}$	LTM
120	10	12	8	12	12	16	3
240	12	20	15	20	20	25	20
480	20	24	18	24	24	30	120
960	24	40	32	40	40	48	600

$L_g = L$ y $H_g = H$ con L y H números naturales. Dichos intervalos se han generado de manera aleatoria.

TABLA II
Características de las instancias *Type1_22*

		TGI	TNE		
n	m	n/m	L	H	LTM
2000	2	1000	866	1134	1200
2000	10	200	173	227	1200
2000	25	80	51	109	1200
2000	50	40	26	54	1200
2000	100	20	13	27	1200
2000	200	10	6	14	1200

Se han considerado dos medidas de efectividad para comparar los resultados de los algoritmos:

- *Desviación porcentual relativa* (Dev) sobre cada instancia:

$$Dev = \frac{|E - B|}{B} \times 100\,\%,$$

donde E es el valor de la función objetivo de la mejor solución encontrada por el algoritmo sobre una instancia concreta y B es el mejor valor encontrado sobre dicha instancia por cualquier algoritmo.
- *#mejor*: número de instancias para las que el algoritmo encuentra el mejor valor de función objetivo (de nuevo, encontrado por cualquier algoritmo).

Los experimentos se han realizado en un ordenador Intel® Core™ i7 CPU930 @ 2.80GHz con 12 Gb de RAM. Nótese que sólo se ha realizado una ejecución de cada algoritmo por instancia.

B. Estudio de los Parámetros de IG-MDGP

El objetivo de esta sección es estudiar el comportamiento de IG-MDGP dependiendo del valor de los distintos parámetros asociados al algoritmo. Estos parámetros son la proporción de elementos eliminados en la fase de destrucción (n_d), el método de búsqueda local y el criterio de aceptación.

Comenzamos el estudio analizando el comportamiento de las 9 configuraciones resultantes al variar el parámetro n_d (0.25, 0.50, 0.75) y el criterio de aceptación (A1, A2 y A3). Para ello, fijamos los procedimientos de búsqueda local, utilizando *mov-P* para la etapa de búsqueda local mediante movimientos y *int-P* para la de búsqueda local mediante intercambios. En la Tabla III, podemos observar que $\{n_d$=0.25 + A3$\}$ obtiene los mejores resultados (Dev

= 0.31 % y #mejor = 30 sobre las instancias *RanInt*; y Dev = 1.17 % y #mejor = 18 para las instancias *Type1_22*).

TABLA III
RESULTADOS DE LA COMBINACIÓN DE n_d Y CRITERIO DE ACEPTACIÓN

	RanInt		Type1_22		
{n_d+CA}	Dev	#mejor	Dev	#mejor	Total
{0.25+A1}	5.61 %	0	1.95 %	18	18
{0.25+A2}	4.06 %	2	1.79 %	11	13
{0.25+A3}	0.31 %	30	1.17 %	18	48
{0.50+A1}	4.86 %	0	4.51 %	5	5
{0.50+A2}	3.81 %	0	4.25 %	5	5
{0.50+A3}	2.93 %	1	3.42 %	18	19
{0.75+A1}	4.38 %	3	4.67 %	6	9
{0.75+A2}	4.37 %	4	4.39 %	5	9
{0.75+A3}	5.73 %	0	4.02 %	15	15

Las Tablas IV y V resumen los resultados de los test estadísticos para este experimento, para las instancias *RanInt* y *Type1_22*, respectivamente. Para cada configuración del algoritmo, la columna *Ranking* se corresponde con el rango promedio. Por su parte, las tres últimas columnas se centran en un análisis *post-hoc*, aplicando el test de Holm para comparar la configuracion con mejor ranking (algoritmo de control) frente a las demás configuraciones (*uno frente a varios*).

TABLA IV
RESULTADOS DEL TEST DE HOLM (*RanInt*)

IG		Test de Holm		
{n_d+CA}	*Ranking*	*p-valor*	α-*ajustado*	*¿Dif?*
{0.75+A3}	6.725	0	0.00625	si
{0.25+A1}	6.525	0	0.00714	si
{0.50+A1}	5.775	0	0.00833	si
{0.75+A1}	5.475	0	0.01	si
{0.75+A2}	5.425	0	0.0125	si
{0.25+A2}	5.175	0	0.01667	si
{0.50+A2}	4.825	0	0.025	si
{0.50+A3}	3.7	0.00015	0.05	si
{0.25+A3}	1.375			

En la Tabla IV, antes de aplicar Holm, aplicamos el test de Friedman para comprobar que existen diferencias significativas entre los algoritmos comparados. El valor del estadístico del test de Friedman es 113.066667 y su *p-valor* computado es 0, indicando así que existen diferencias entre las diversas configuraciones al considerar las instancias *RanInt*. Al existir tales diferencias, proponemos utilizar el test de Holm para realizar un análisis *a posteriori*. Si el valor de la columna *p-valor* es menor que el valor de la columna α-*ajustado* para cada algoritmo, existirán diferencias entre la combinación correspondiente a esos valores y el algoritmo control. La configuración {n_d=0.25 + A3} se corresponde con el algoritmo control ya que presenta el menor ranking promedio. El test de Holm encuentra diferencias entre la configuración {n_d=0.25 + A3} y las restantes (columna *¿Dif?*).

TABLA V
RESULTADOS DEL TEST DE HOLM (*Type1_22*)

IG		Test de Holm		
{n_d+CA}	*Ranking*	*p-valor*	α-*ajustado*	*¿Dif?*
{0.75+A1}	6.367	0	0.00625	si
{0.50+A1}	6.35	0	0.00714	si
{0.50+A2}	5.758	0	0.00833	si
{0.75+A2}	5.608	0	0.01	si
{0.25+A1}	4.725	0.00072	0.0125	si
{0.75+A3}	4.708	0.00081	0.01667	si
{0.25+A2}	4.342	0.00888	0.025	si
{0.50+A3}	4.108	0.03156	0.05	si
{0.25+A3}	3.033			

Por su parte, en la Tabla V, encontramos que la configuración {n_d=0.25 + A3} (con el *p-valor* asociado al test de Friedman de 0) difiere de manera significativa de las demás, como sucedía con las instancias *RanInt*.

Para continuar con el análisis de parámetros, vamos a centrarnos en localizar los métodos de búsqueda local con mejor comportamiento. Nos centramos en IG-MDGP con n_d=0.25 y el criterio de aceptación A3 (reemplazar siempre). Recordemos que la fase de búsqueda local se compone de dos etapas: una etapa de búsqueda local basada en movimientos (*mov-P* y *mov-M*) y una etapa de búsqueda local basada en intercambios (*int-P*, *int-M* y *int-RP*). En las Tablas VI y VII, presentamos un resumen de los resultados de las posibles combinaciones entre estas alternativas, al considerar las instancias *RanInt* y *Type1_22*, respectivamente.

TABLA VI
RESULTADOS DE LAS COMBINACIONES DE TÉCNICAS DE BÚSQUEDA LOCAL (*RanInt*)

RanInt		Dev	#mejor
int-P	*mov-P*	2.07 %	2
	mov-M	0.95 %	16
int-M	*mov-P*	4.44 %	1
	mov-M	2.39 %	3
int-RP	*mov-P*	1.29 %	15
	mov-M	0.80 %	21

TABLA VII
RESULTADOS DE LAS COMBINACIONES DE TÉCNICAS DE BÚSQUEDA LOCAL (*Type1_22*)

RanInt		Dev	#mejor
int-P	*mov-P*	2.74 %	4
	mov-M	0.96 %	15
int-M	*mov-P*	5.02 %	3
	mov-M	2.99 %	3
int-RP	*mov-P*	1.66 %	19
	mov-M	0.29 %	38

A la vista de los resultados en estas tablas, observamos que, para ambos tipos de instancias, la combinación {*mov-M+int-RP*} obtiene una menor desviación porcentual promedio y un mayor número de mejores resultados.

TABLA VIII
RESULTADOS DEL TEST DE HOLM (*RanInt*)

{0.25+A3}		Test de Holm		
{BLI+BLM}	*Rank.*	*p-valor*	α-*ajust.*	*¿Dif?*
{mov-P+int-M}	5.663	0	0.01	si
{mov-P+int-P}	3.963	0.0001	0.0125	si
{mov-M+int-M}	3.863	0.0003	0.0167	si
{mov-P+int-RP}	2.75	0.3241	0.025	no
{mov-M+int-P}	2.425	0.8343	0.05	no
{mov-M+int-RP}	2.338			

También, hemos aplicado test estadísticos para analizar los datos de este experimento. Para el caso de *RanInt*, bajo un nivel de confianza del 5 %, el test de Friedman indica que existen diferencias entre las posibles combinaciones de la búsqueda local (el estadístico de Friedman, con 8 grados de libertad, es 92.471429 y *p-valor* es 0). A continuación, utilizamos el test de Holm para concretar qué configuraciones presentan diferencias con respecto al algoritmos de control (Tabla VIII). La combinación {*mov-M+int-RP*} es el algoritmo de control, pues tiene el mejor ranking. En la columna *¿Dif?*, observamos que el algoritmo de control {*mov-M+int-RP*} no presenta diferencias significativas frente a las configuraciones {*mov-M+int-P*} y {*mov-P+int-RP*}.

TABLA IX
TEST DE HOLM (*Type1_22*)

{0.25+A3}		Test de Holm		
{BLI+BLM}	*Rank.*	*p-valor*	α-*ajust.*	*¿Dif?*
{mov-P+int-M}	5.5	0	0.01	si
{mov-M+int-M}	4.35	0	0.0125	si
{mov-P+int-P}	3.892	0	0.01667	si
{mov-P+int-RP}	2.917	0.0013	0.025	si
{mov-M+int-P}	2.525	0.0381	0.05	si
{mov-M+int-RP}	1.817			

Por su parte, la Tabla IX muestra que, bajo un nivel de significación del 5 %, la combinación de los operadores *int-RP* y *mov-M* si que presentan diferencias significativas al considerar las instancias *Type1_22*.

A partir de los resultados presentados en esta sección, hemos decidido asumir como mejor configuración para IG-MDGP {(n_d=0.25)+A3+mov-M+int-RP}. Aunque en las instancias *RanInt*, *int-RP* no presenta diferencias significativas con respecto a *int-P* (Tabla VIII), para instancias más difíciles, *Type1_22*, sí existen diferencias significativas y las medidas de eficacia son mejores en ambos casos (Tablas VI y VII). Para simplificar la escritura, en secciones posteriores, cuando hagamos alusión a IG-MDGP, se dará por hecho que utilizamos esta configuración.

C. IG-MDGP frente al Estado del Arte

En [5], los autores utilizan el método de mejora LCW sólo para los casos en que el tamaño de cada grupo sea el mismo. Como el MDGP involucra grupos cuyo tamaño no tienen por qué ser fijos, hemos realizado modificaciones en LCW para adaptarlo a este caso

A la hora de implementar el algoritmo memético GA+LS [5], existen determinadas cuestiones que no quedan del todo claras y que a continuación tratarán de esclarecerse. En concreto, la selección de la nueva población presenta algunas ambigüedades. Para solucionar las mismas, hemos optado por utilizar dos variantes: *los padres e hijos con mayor valor de la función objetivo formarán la nueva población*; y *la selección de la nueva población se basará en probabilidades* (utilizando el mecanismo de la ruleta). Los resultados obtenidos muestran que la segunda variante ofrece mejores resultados, y por ello hemos optado por utilizarla para comparar la eficiencia de los algoritmos.

A continuación, presentamos el estudio comparativo de IG-MDGP con LCW y GA+LS, actual estado del arte. Para poder realizar una comparación justa, se ha establecido un tiempo límite de ejecución igual para todos los algoritmos comparados. Las Tablas X y XI muestran un resumen de la desviación porcentual relativa promediada para las instancias *RanInt* y *Type1_22*, respectivamente. El sufijo *tipo* describe si la instancia presenta iguales tamaños de los grupos (ss) o por el contrario está sujeta a distintos tamaños de grupos (ds).

TABLA X
DESVIACIÓN PROMEDIO POR INSTANCIA (*RanInt*)

Instancia-n-tipo	LCW	GA+LS	IG-MDGP
RanInt-120-ds	9.43 %	3.21 %	0 %
RanInt-120-ss	4.03 %	4.07 %	0 %
RanInt-240-ds	5.93 %	4.88 %	0 %
RanInt-240-ss	3.31 %	4.01 %	0 %
RanInt-480-ds	5.32 %	5.38 %	0 %
RanInt-480-ss	3.28 %	4.25 %	0 %
RanInt-960-ds	4.12 %	4.99 %	0 %
RanInt-960-ss	2.67 %	3.86 %	0 %
Promedio Total	4.43 %	3.85 %	0 %

La Tabla XII muestra un resumen de los resultados de los test estadísticos. Tanto para el caso de $n = 120$ y $n = 240$, como para el caso de $n = 480$ y $n = 960$, el *p-valor* computado para el test de Friedman es 0. Entonces, en ambos subgrupos de instancias, existen diferencias entre el algoritmo de control (en ambos caso IG-MDGP) y los restantes algoritmos. El test de Holm indica que las soluciones obtenidas con IG-MDGP son mejores que las alcanzadas por LCW y GA+LS.

TABLA XI
DESVIACIÓN PROMEDIO POR INSTANCIA (*Type1_22*)

Instancia-n-L_g-tipo	LCW	GA+LS	IG-MDGP
Type1_22-2000-1000-ss	0.09 %	0.23 %	0 %
Type1_22-2000-200-ss	0.94 %	1.76 %	0 %
Type1_22-2000-80-ss	6.26 %	3.94 %	0 %
Type1_22-2000-40-ss	19.06 %	5.91 %	0 %
Type1_22-2000-20-ss	28.26 %	6.13 %	0 %
Type1_22-2000-10-ss	34.41 %	6.57 %	0 %
Type1_22-2000-866-ds	1.84 %	0.26 %	0 %
Type1_22-2000-173-ds	2.68 %	2.41 %	0 %
Type1_22-2000-51-ds	15.23 %	7.25 %	0 %
Type1_22-2000-26-ds	25.24 %	8.51 %	0 %
Type1_22-2000-13-ds	32.59 %	8.64 %	0 %
Type1_22-2000-6-ds	42.75 %	7.93 %	0 %
Promedio Total	17.45 %	4.96 %	0 %

TABLA XII
IG-MDGP VERSUS LCW Y GA+LS (*RanInt*)

n=120, n=240		Test de Holm		
Metaheur.	*Ranking*	*p-valor*	*α-ajustado*	*¿Dif?*
LCW	2.65	0	0.025	si
GA+LS	2.35	0.00002	0.05	si
IG-MDGP	1			
n=480, n=960		**Test de Holm**		
Metaheur.	*Ranking*	*p-valor*	*α-ajustado*	*¿Dif?*
GA+LS	2.85	0	0.025	si
LCW	2.15	0.000276	0.05	si
IG-MDGP	1			

Para el caso de las instancias *Type1_22* (Tabla XIII), encontramos diferencias significativas entre los tres métodos al utilizar el test de Friedman (el valor del estadístico es 102.033 con un *p-valor* computado por el test de 0). Por su parte, el test de Holm indica que el método IG-MDGP difiere tanto de LCW como de GA+LS, al igual que con las instancias *RanInt* (Tabla XII).

TABLA XIII
IG-MDGP VERSUS LCW Y GA+LS (*Type1_22*)

n=2000		Test de Holm		
Metaheur.	*Ranking*	*p-valor*	*α-ajust.*	*¿Dif?*
LCW	2.8167	0	0.025	si
GA+LS	2.1833	0	0.05	si
IG-MDGP	1			

Podemos destacar del análisis experimental un hecho importante: tanto para las instancias *RanInt* como para las instancias *Type1_22* utilizadas, IG-MDGP genera mejores resultados que GA+LS en la totalidad de los casos (*Dev* del 0 %). Así, podemos concluir que nuestra propuesta IG-MDGP es muy competitiva frente al estado del arte actual.

V. CONCLUSIONES

En este trabajo, hemos presentado un modelo de IG para abordar el MDGP. En primer lugar, hemos realizado un estudio experimental para comparar el rendimiento de IG-MDGP utilizando diferentes valores para sus parámetros asociados: proporción de elementos eliminados de la solución actual durante la etapa de destrucción, tipo de búsqueda local y tipo de criterio de aceptación. Como resultado de este estudio, hemos obtenido la configuración que presenta un mejor rendimiento bajo el conjunto de instancias considerado. Además, dicho algoritmo se ha mostrado muy competitivo frente al estado del arte, obteniendo, en el 100 % de las instancias, soluciones de mejor calidad.

La propuesta de IG presentada en este trabajo representa una contribución interesante, merecedora de más estudios en diferentes líneas. Entre estas, podemos destacar la aplicación a problemas análogos a MDGP y también a problemas reales, en ambientes sanitarios, tecnológicos, jurídicos, deportivos, de enseñanza, etc.

REFERENCIAS

[1] K.R. Baker, S.G. Powell, *Methods for assigning students to groups: a study of alternative objective functions*, Journal of the Operational Research Society, 53(4), 397-404, 2002.

[2] A. Duarte, R. Martí, *Tabu search and GRASP for the maximum diversity problem*, European Journal of Operational Research, 178(1), 71–84, 2007.

[3] T.A. Feo, M. Khellaf, *A class of bounded approximation algorithms for graph partitioning*, Networks, 20(2), 181-195, 1990.

[4] T.A. Feo, O. Goldschmidt, M. Khellaf, *One-half approximation algorithms for the k-partition problem*, Operations research, 1992.

[5] Z.P. Fan, Y. Chen, J. Ma, S. Zeng, *A hybrid genetic algorithmic approach to the maximally diverse grouping problem*, Journal of the Operational Research Society, 62(7), 92-99, 2010.

[6] L.W. Jacobs, M.J. Brusco, *A local search heuristic for large set covering problems*, Naval Research Logistics Quarterly, 42, 1129-1140, 1995.

[7] Q. Kang, H. Heb, H. Song, *Task assignment in heterogeneous computing systems using an effective iterated greedy algorithm*, The Journal of Systems and Software, 84, 985-992, 2011.

[8] V. Lofti, R. Cerveny, *A final exam scheduling package*, Journal of the Operational Research Society, 42, 205-216, 1991.

[9] M. Lozano, D. Molina, C. García-Martínez, *Iterated greedy for the maximum diversity problem*, European Journal of Operational Research, 214(1), 31-38, 2011.

[10] F.A. O'Brien, J. Mingers, *A heuristic algorithm for the equitable partitioning problem*, Omega, 25(2), 215-223, 1997.

[11] R. Ruiz, T. Stutzle, *A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem* European Journal of Operational Research, 177(3), 2033-2049, 2007.

[12] K.-C. Ying, H.-M. Cheng, *Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic* Expert Systems with Applications, 37(4), 2848-2852, 2010.

[13] R.R. Weitz, S. Lakshminarayanan, *An empirical comparison of heuristic and graph theoretic methods for creating maximally diverse groups, VLSI design, and exam scheduling*, Omega, 25(4), 473-482, 1997.

[14] R.R. Weitz, S. Lakshminarayanan, *An empirical comparison of heuristic for creating maximally diverse groups*, Journal of Operation Research Society, 49, 635-646, 1998.