ALGORITMO GRASP HIBRIDO PARA RESOLVER UNA NUEVA VARIANTE DEL PROBLEMA DE LA DIVERSIDAD MAXIMA

Fernando Sandoya
Instituto de Ciencias Matemáticas, Escuela Superior Politécnica del Litoral
Campus G. Galindo, km 30.5 Vía Perimetral. Apartado 09-01-5863, Guayaquil-Ecuador
fsandoya@espol.edu.ec

Rafael Martí
Departamento de Estadística e Investigación Operativa
Facultad de Ciencias Matemáticas, Universidad de Valencia-España
Rafael.Marti@uv.es

RESUMEN

Clásicamente, el problema de la diversidad máxima consiste en seleccionar un número fijo de elementos, desde un conjunto, de tal manera que una medida de diversidad sea maximizada. En esta investigación analizamos un nuevo modelo, denominado Max-Mean, en el cuál se maximiza la diversidad promedio, con el número de elementos a seleccionar también variable de decisión. El nuevo modelo es muy útil para casos en los cuáles las distancias representan afinidades y por tanto no se restringe a que sólo tomen valores no negativos. Se desarrolla un algoritmo híbrido muy eficiente basado en GRASP, al que incorporamos las estrategias de búsqueda en vecindades variables y rencadenamiento de trayectorias. Y se prueba que este algoritmo es muy eficiente en comparación con heurísticas propuestas para otros problemas de diversidad máxima.

PALABRAS CLAVE. Problema de la diversidad máxima, Programación no lineal binaria, Metaheurística GRASP.

Área principal MH , OC, OA

ABSTRACT

Typically, the maximum diversity problem is to select a fixed number of elements, from a set, so that a diversity measure is maximized. In particular, we target the max-mean dispersion model in which the average distance between the selected elements is maximized, with the number of items to select decision variable also. The new model is very useful in cases in which the distances representing similarities and thus not restricted to take only non-negative values. We develop a very efficient hybrid algorithm based on GRASP, which incorporate local search strategies based on the Variable Neighborhood methodology, and is proven that this algorithm is very efficient compared to other heuristics proposed for other problems of maximum diversity.

KEYWORDS. Maximun diversity problem, Binary nonlinear programming, GRASP.

Main area MH , OC, OA

1. Introducción:

El proceso de seleccionar objetos, actividades, personas, proyectos, recursos, etc. es una de las actividades que frecuentemente realizamos con algún objetivo, y basados en algún criterio económico, de espacio, afectivo, político etc. En la mayoría de esos casos se trata de elegir el mejor subconjunto desde un conjunto grande de posibilidades, el mejor en algún sentido, y suele ser de interés que los elementos seleccionados no sean similares, sino que tengan características diferentes para que representen la diversidad existente en el conjunto original. Las personas suelen tomar estas decisiones intuitivamente y no analizando el correspondiente problema de decisión, pero el sentido común, generalmente, no es un buen consejero, sobretodo si estas decisiones tienen consecuencias económicas.
En la literatura de investigación de operaciones, elegir el subconjunto con la mayor diversidad configura el Problema de la Diversidad Máxima (MDP), que ha permitido tratar aplicaciones concretas como: localización de unidades logísticas mutuamente competitivas (Glover y otros, 1998), composición de paneles de jurado (Lozano y otros, 2011), ubicación de instalaciones peligrosas (Ercut y otros, 1989), formulación de políticas de inmigración (Kuo y otros, 1993), desarrollo de nuevas drogas (Meinl, 2010), entre otras. Una aplicación nueva es la conformación de equipos de trabajo, donde la diversidad jugaría un papel crucial (Martí y Sandoya, 2012).

2. El problema de la Diversidad Máxima (MDP):

Si $V = \{1, 2, \cdots, n\}$ es un conjunto y M es el subconjunto a seleccionar, de cardinalidad conocida o no, buscaremos optimizar el objetivo representado en la ecuación (1.1).

$$Max\ f_1(M) = div(M) \qquad (2.1)$$

Donde $div(M)$ representa la medición que hemos realizado de la diversidad en M. Los distintos modelos que se han planteado para el MDP difieren en como evaluarla. En particular, en este artículo, se aborda un nuevo modelo: el *Modelo de Dispersión del Máximo Promedio*, que denominamos modelo Max-Mean, en el cual $div(M)$ representa la disimiltud promedio en M, y en el cual el número de elementos seleccionados también es una variable de decisión.

Distancias, similitud y diversidad:

Para evaluar $div(M)$ de la ecuación (1.1), se requiere una relación d_{ij} que describa la distancia, o disimilitud entre cada pareja i, j. En sistemas complejos como los grupos de personas, una operación fundamental es la valoración de la similaridad entre los individuos, que en muchos casos es estimada como una distancia en algún espacio con características adecuadas, generalmente un espacio métrico, y más específicamente con la distancia euclidiana, como en (Resende y otros, 2010). Una condición importante que no se puede explotar en esas valoraciones es el signo de d_{ij}, que puede servir para caracterizar la afinidad (con negativo), y la no afinidad (con positivo). Por tanto son necesarias otras formas de caracterizar a los elementos d_{ij}.
Una medida de similitud en grupos de solucionadores de problemas establecida en (Lu Hong, 2004) en donde se estima d_{ij} de la siguiente manera: Dados dos individuos i, j con características $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$, $x_j = (x_{j1}, x_{j2}, \dots, x_{jp})$:

$$d_{ij} = \frac{\sum_{l=1}^{p} \delta(x_{il}, x_{jl})}{p} \qquad (2.2)$$

Donde:

$$\delta(x_{il}, x_{jl}) = \begin{cases} -1 & si\ x_{il} = x_{jl} \\ |x_{il} - x_{jl}| & si\ x_{il} \neq x_{jl} \end{cases}$$

Esta medida toma valores negativos, en el caso de similaridad, y positivos, en el caso contrario.
Luego de caracterizar a d_{ij}, se debe especificar como estimar la diversidad de un conjunto. Pero esto dependerá de la aplicación concreta que se quiera analizar, mencionamos dos maneras:

La medida de dispersión de la Suma: Bajo esta medida la diversidad se calcula como:

$$div(M) = \sum_{i<j,i,j\in M} d_{ij} \tag{2.3}$$

La medida de la dispersión Promedio: En este caso nos interesa la diversidad media:

$$div(M) = \frac{\sum_{i<j,i,j\in M} d_{ij}}{|M|} \tag{2.4}$$

3. Modelos y Formulaciones:

Dado un conjunto $V = \{1, 2, \cdots, n\}$, y la relación de disimilitud d_{ij} entre cada pareja de elementos de V, el MDP consiste en seleccionar un subconjunto $M \subset V$, de cardinalidad $m < n$ conocida o no, que maximice la ecuación (1.1). La manera en que medimos la diversidad en esta ecuación da lugar a distintos problemas, en particular en los problemas Max-Sum y Max-Mean se maximizan las medidas de diversidad definidas en (1.3) y (1.4), respectivamente:

El Problema Max-Sum:

$$\max_{M\subset V,|M|=m} \sum_{i<j,i,j\in M} d_{ij}$$

Introduciendo las variables binarias: $x_i = \begin{cases} 1 \text{ si el elemento i es seleccionado} \\ 0 \text{ sino} \end{cases}; 1 \le i \le n$, el problema puede ser formulado como un problema de programación cuadrática binaria:

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \tag{3.1}$$

$$s.t. \sum_{i=1}^{n} x_i = m \tag{3.2}$$

$$x_i \in \{0,1\};\ 1 \le i \le n \tag{3.3}$$

La formulación (1.5)-(1.7) puede ser linealizada introduciendo nuevas variables binarias como se demuestra en (Kuo y otros, 1993).

El problema Max-Mean:

$$\max_{M\subset V,|M|\ge 2} \frac{\sum_{i<j,i,j\in M} d_{ij}}{|M|}$$

En este modelo se maximiza la diversidad promedio, obsérvese que el número de elementos a escoger también es una variable de decisión. Una formulación con variables binarias es entonces:

$$\max \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j}{\sum_{i=1}^{n} x_i} \tag{3.4}$$

$$s.t. \sum_{i=1}^{n} x_i \ge 2 \tag{3.5}$$

$$x_i \in \{0,1\},\ 1 \le i \le n \tag{3.6}$$

Tal como se lo presenta este es un problema de optimización binaria fraccional, que puede ser linealizado utilizando nuevas variables binarias, como se observa en (Martí y Sandoya, 2012).

Nótese que se puede resolver el problema Max-Mean con un método de solución para el problema Max-Sum, aplicándolo repetidamente para todos los valores factibles de $m = |M|$. Sorprendentemente, como se demuestra en (Martí y Sandoya, 2012) hallar la solución del problema Max-Mean con Cplex a través de la estrategia de resolver los $(n-1)$ problemas de tipo Max-Sum requiere menos tiempo que resolver directamente la formulación (1.8)-(3.6).

Complejidad Computacional: Es conocido que el problema Max-Sum es fuertemente NP-duro (Erkut, 1990). Recientemente también se ha demostrado la propiedad 1 (Prokopyev y otros, 2009) que el problema Max-Mean es fuertemente NP-duro si d_{ij} toman valores positivos y negativos; en (Martí y Sandoya, 2012) se demuestra la propiedad 2, que indica que si d_{ij} es una métrica, entonces $div(M)$ para cualquier $M \subset V$ es siempre menor que $div(M \cup \{k\})$ para cualquier $k \notin M$, luego, una solución con $m < n$ elementos no puede ser óptima en el problema Max-Mean, de ahí que el óptimo sería seleccionar todos los elementos.

Propiedad 1: Si los coeficientes d_{ij} no tienen restricciones en el signo, entonces el problema Max-Mean es fuertemente NP-duro

Propiedad 2: Si los coeficientes d_{ij} son no negativos, satisfacen la desigualdad triangular y son simétricos, entonces el problema Max-Mean tiene como solución $M = V$; es decir, todos los elementos del conjunto deben ser seleccionados.

4. Desarrollo del algoritmo GRASP3+PR para resolver el problema Max-Mean

Se propone una heurística muy eficiente que consiste de una fase de construcción GRASP, con búsqueda local basada vecindades variables, método que denominamos heurística GRASP3, que luego es mejorado con la metodología de Rencadenamiento de Trayectorias, con lo que obtenemos la heurística GRASP3+PR. Se trabaja con los dos tipos de ejemplos de prueba que han sido desarrollados especialmente para probar la eficiencia de los algoritmos.

Problemas de prueba: En la literatura se reportan instancias de prueba para el MDP en sus variantes clásicas Max-Min y Max-Sum, que en general son instancias euclidianas y aleatorias con valores de d_{ij} no negativas. Como para el problema Max-Mean requerimos de valores d_{ij} sin restricción de signo, desarrollamos problemas de prueba clasificados en Tipo I y Tipo II.
TIPO I: Son matrices simétricas con números aleatorios uniformes en $[-1,1]$. Estos números podrían representar grados de afinidad o de no afinidad entre individuos en un grupo social.
TIPO II: Son matrices simétricas, cuyos coeficientes se generaron como números aleatorios uniformes en $[-1,-0.5] \cup [0.5,1]$. Al no generarse valores en $[-0.5,0.5]$ podrían representar grupos en un ambiente de polarización, con mucha afinidad o mucha no afinidad entre individuos, y poca indiferencia.
Para cada uno de los tipos, se generaron 10 problemas de prueba de este tipo para valores de $n = 20,\ 25, 30, 35, 150\ y\ 500$, para ejemplificar el rendimiento de los métodos de solución propuestos en problemas de tamaño pequeño, mediano y grande.

Fase de Construcción de GRASP3:
En la Figura 1 se muestra el resultado de resolver el problema Max-Mean de una manera indirecta para los ejemplos de prueba de tamaño pequeño $n = 30$, resolviendo en cada caso de manera exacta las formulaciones MIP de 29 problemas de tipo Max-Sum. En el eje y consta el valor óptimo de la función objetivo del problema Max-Sum dividido para el respectivo valor de m, donde m, representado en el eje x, toma todos sus 29 valores posibles: 2, 3, …, 30.

Los problemas se resolvieron con Cplex V.12. La Figura 1 muestra un patrón aproximadamente cuasi-cóncavo para los valores óptimos del problema Max-Mean en función de m. Con base a este comportamiento se diseñó un método constructivo de tipo GRASP, en el cual añadimos nuevos elementos a la solución parcial bajo construcción siempre que el valor de la función

objetivo mejore, y cuando se observa un decrecimiento de este valor, se detiene la fase de construcción. Así el método selecciona por sí mismo, basado en un método racional, el valor de m que parece más adecuado.

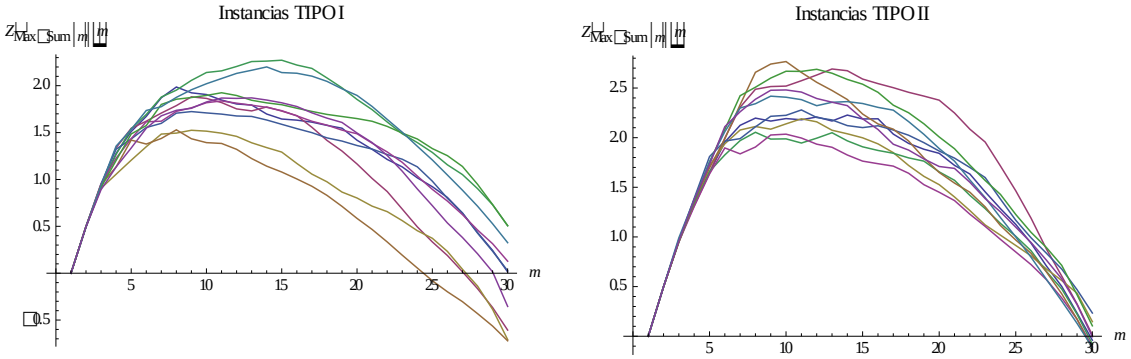

Figura 1. Evolución de los valores óptimos del problema Max-Sum divididos para el valor de m para 10 ejemplos de prueba de tipo I y de tipo II de tamaño $\boldsymbol{n}$=30

En nuestra investigación, en lugar de una típica construcción GRASP, en la cual para construir la Lista de Candidatos Restringida primero cada elemento candidato a seleccionarse es evaluado por una función voraz, y luego un elemento es seleccionado aleatoriamente, utilizamos un diseño alternativo, en concordancia con lo propuesto en estudios recientes (Resende y otros, 2004), en el cual primero se aplica la aleatorización y luego la parte voraz, obteniendo mejores resultados.

Concretamente, dada la solución parcial M_k con k elementos seleccionados, la lista de candidatos CL está formada por los $(n-k)$ elementos no seleccionados. La lista de candidatos restringida, RCL, contiene una fracción α $(0 < \alpha < 1)$ de los elementos de CL seleccionados aleatoriamente. Después, cada $i \in RCL$ es evaluado por el cambio que produce en la función objetivo; es decir:

$$eval(i) = div(M_k \cup \{i\}) - div(M_k)$$

Donde $div(\cdot)$ es la función de diversidad del promedio definida en la ecuación (1.3).

Después, el método selecciona el mejor candidato i^* en RCL si esto mejora la solución parcial actual; es decir, si $eval(i^*) > 0$, y lo añade a la solución parcial, $M_{k+1} = M_k \cup \{i^*\}$; en caso contrario, si $eval(i^*) \leq 0$, el método para. En la Figura 2 se muestra el seudocódigo respectivo.

1. Seleccionar aleatoriamente un elemento i^* en $V = \{1, 2, \dots, n\}$
2. Poner $M_1 = \{i^*\}$, $k = 1$ y $improve = 1$.

Mientras $(improve = 1)$:

3. Calcular $CL = \{1, 2, \dots, n\} \setminus M_k$
4. Construir RCL a partir de $\alpha\ |CL|$ elementos aleatoriamente seleccionados de CL
5. Calcular $eval(i) = div(M_k \cup \{i\}) - div(M_k)\ \forall i \in RCL$
6. Seleccionar el elemento i^* en RCL con el máximo valor de $eval$

Si ($eval(i^*) > 0$):

7. $M_{k+1} = M_k \cup \{i^*\}$
8. $k = k + 1$

Sino

9. $improve = 0$

Figura 2. Fase de construcción del método GRASP3

Búsqueda Local en GRASP3: Basados en la metodología de Búsqueda en Vecindades Variables, (Hansen y otros, 2003), consideramos la combinación de tres tipos de vecindad para la búsqueda local:

- N_1: Remover un elemento de la solución actual, reduciendo así el número de elementos seleccionados en uno.
- N_2: Intercambiar un elemento seleccionado, con uno no seleccionado, manteniendo así constante el número de elementos seleccionados.
- N_3: Añadir un elemento no seleccionado en la solución actual, incrementando así el número de elementos seleccionados en una unidad.

Dada una solución, M_m, la búsqueda local primero intenta obtener una solución en N_1 que la mejore. Si esto sucede, y logramos hallar M'_{m-1} con $div(M'_{m-1}) > div(M_m)$, entonces aplicamos el movimiento y consideramos M'_{m-1} como la solución actual. Caso contrario, el método explora la vecindad N_2 y busca el primer intercambio que mejora M_m. Si esto sucede, y logramos hallar M'_m con $div(M'_m) > div(M_m)$, entonces aplicamos el movimiento y consideramos M'_m como la solución actual. En cualquier caso, sin tener en cuenta si hallamos la solución mejorada en N_1 o en N_2, en la próxima iteración el método explora nuevamente N_1.
Si ni la vecindad N_1 ni la vecindad N_2 contienen una mejor solución que la solución actual, entonces finalmente recurrimos a explorar N_3. Si esta exploración es exitosa, y logramos hallar M'_{m+1} con $div(M'_{m+1}) > div(M_m)$, entonces aplicamos el movimiento y consideramos M'_{m+1} como la solución actual, y luego regresamos a explorar N_1 en la siguiente iteración. Caso contrario, estaríamos en una situación en la que ninguna de las vecindades contiene una mejor solución que la solución actual, y por tanto el método termina.
Dada una solución M_m, calculamos la contribución de cada elemento seleccionado i, así como la contribución potencial de cada elemento no seleccionado i como:

$$d_s(i, M_m) = \sum_{j \in M_m} d_{ij}$$

Entonces, cuando exploramos N_1 para remover un elemento de M_m, buscamos los elementos seleccionados en el orden dado por d_s, en N_2 vamos probando los elementos seleccionados en el mismo orden pero los elementos no seleccionados en el orden inverso. Y en N_3 los elementos no seleccionados, se exploran de la misma forma que en N_2. La Figura 3 muestra el seudocódigo.

1. Sea M_m la solución inicial, determinada en la fase de construcción
2. Calcular $d_s(i, M_m)$ para todo elemento $i \in V$.

Mientras $(improve = 1)$

3. Explorar los elementos seleccionados en el orden de d_s

   Si existe un movimiento en N_1 que mejora la solución actual,

   4. Realizar el primer movimiento de mejora (N_1)
   5. Sea M'_{m-1} la solución actual

   Sino

   6. Buscar para realizar los intercambios en el orden de d_s

      Si existe un movimiento en N_2 que mejora la solución actual,

      7. Realizar el primer movimiento de mejora (N_2)
      8. Sea M'_m la solución actual

      Sino

      9. Buscar los elementos no seleccionados en el orden reverso de d_s

         Si existe un movimiento en N_3 que mejora la solución actual,

         10. Realizar el primer movimiento de mejora (N_3)
         11. Sea M'_{m+1} la solución actual

         Sino

         12. $improve = 0$
         13. Actualizar los valores de d_s

Figura 3. Algoritmo de búsqueda local para GRASP3

GRASP3+Rencadenamiento de Trayectorias: El Rencadenamiento de Trayectorias es usado como una estrategia de intensificación de la búsqueda en regiones prometedoras del espacio de soluciones. Opera sobre un conjunto de soluciones, denominado "Conjunto Elite" (ES), que se genera con la aplicación de un método previo, en nuestro caso la heurística GRASP3.

La estrategia del algoritmo GRASP3+PR es la siguiente: Inicialmente se aplica GRASP3, para $b = |ES|$ iteraciones, con lo que poblamos el conjunto élite, y ordenamos las soluciones en ES en base a su valor objetivo desde la mejor, representada con x^1, a la peor, representada con x^b. Una vez poblado el conjunto ES por b soluciones iniciales, en las siguientes iteraciones del algoritmo, se determina cuando una nueva solución generada, x', califica o no para entrar a ES, y cuál solución en ES debe salir, con el fin de conservar la cardinalidad de ES igual a b. Concretamente, una nueva solución x' ingresa al conjunto ES si se cumple una de las siguientes condiciones:

1. Si x' es mejor que x^1, esta solución ingresa al conjunto élite, esto indica que se privilegia la calidad como una condición esencial para pertenecer al conjunto ES.
2. Si x' es peor que x^1 pero mejor que x^b y además es suficientemente diferente que las otras soluciones del conjunto ES, x' también ingresa al conjunto ES, esto con el fin de favorecer la diversidad del conjunto de soluciones élite. Específicamente, x' ingresa al conjunto ES, si se satisface la condición (4.1):

   $$x' \text{ es mejor que } x^1, \text{ o, } x' \text{ es mejor que } x^b \text{ y } d(x', ES) \geq dth \qquad (4.1)$$

   Donde $d(x', ES)$ representa la distancia entre x' y ES, definida en la ecuación (4.2).

Para conservar la cardinalidad de ES igual a b, cuando se añade un elemento debemos remover alguno de los que estaban presentes. Y con el fin de privilegiar la calidad y la diversidad en ES, se remueve aquella solución de peor valor que está más cercana a x' en ES. Para declarar que una solución está cercana o lejana a x' en ES, se considera una medida de distancia entre esta solución y el conjunto de soluciones élite. Interpretando cada solución como $x = (x_1, x_2, \dots, x_n)$, donde x_i = 1 si el elemento i está seleccionado en esa solución y 0 en caso contrario, entonces definimos la distancia entre dos soluciones x, y como en la ecuación (2.2):

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \qquad (4.2)$$

La distancia entre una solución x' y el conjunto de soluciones élite ES, $d(x', ES)$, es la suma de las distancias entre x' y todos los elementos de ES.

$$d(x', ES) = \sum_{y \in ES} d(x', y) \qquad (4.3)$$

Se implementó el parámetro de umbral dth como un porcentaje φ del tamaño total de la población, n, representado como dth^* multiplicado por la cardinalidad del conjunto élite de soluciones, b; es decir:

$$dth = (\varphi\, n)\, b = dth^*\, b \qquad (4.4)$$

La razón de la ecuación (2.4) es que $d(x', ES)$ es la suma de las distancias desde x' hasta cada elemento del conjunto ES, que tiene cardinalidad b. Si consideramos que dos soluciones son suficientemente diferentes cuando en promedio difieran en un porcentaje φ de sus elementos, de acuerdo a la ecuación (2.2) tenemos que $d(x', y) \geq \varphi\, n$, y entonces si se usa la fórmula (4.3) para la evaluación de la distancia de x' a ES, se tiene:

$$d(x', ES) \geq \varphi\, n\, b.$$

A continuación se realiza la fase de intensificación, que se describe a continuación:
Dadas dos soluciones $x, y \in ES$, el procedimiento de Rencadenamiento de trayectorias $PR(x, y)$ inicia con la primera solución x, denominada *solución inicial*, y por medio de un mecanismo gradualmente la transformamos en la solución final y denominada *solución guía*. Para lograr esto en cada iteración consideramos dos movimientos: Remover un elemento que está en x pero que no está presente en y; o, añadir un elemento que no está presente en x pero que si está en y. El método selecciona el mejor de estos candidatos, originando la primera *solución intermedia* de la

trayectoria, representada con $x(1)$. Luego se consideran nuevamente los dos movimientos para $x(1)$: remover un elemento en $x(1)$ que no está en y, o añadir un elemento que no está en $x(1)$ pero que si está en y. El mejor de estos candidatos es la segunda solución intermedia $x(2)$. Así, recursivamente, generamos una trayectoria de soluciones intermedias $x(1),\ x(2), \dots, x(k)$ hasta alcanzar a y. Seguidamente, se aplica a la mejor solución encontrada la fase de mejora por el mecanismo de búsqueda local. En esta investigación realizamos el procedimiento PR para ir de x a y, $PR(x,y)$, y también PR para ir de y a x, $PR(y,x)$, de donde se obtiene la solución x', que será considerada como la mejor solución si su valor objetivo supera a la mejor solución de ES.

5. Heurísticas GRASP1 y GRASP2 adaptadas

En la literatura no se han propuesto métodos de solución heurísticos o exactos para el problema Max-Mean, pero si se han desarrollado algoritmos para la solución de los problemas clásicos de la diversidad máxima, en particular para el problema Max-Sum, con el cuál el problema Max-Mean tiene íntima relación. En nuestra investigación seleccionamos dos de los métodos más eficientes planteados por otros autores para la resolución del problema Max-Sum y los adaptamos a la solución del problema Max-Mean, con el fin de comprobar la eficiencia de nuestro método.

HEURISTICA GRASP1: En (Prokopyev y otros, 2009) se propone un algoritmo GRASP para otra variante del MDP, en el que la solución parcial es M_k, con k elementos seleccionados. Cada fase de construcción de este algoritmo inicia seleccionando aleatoriamente un elemento, el cual genera el conjunto inicial M_1. Luego, en cada iteración, se determina una lista de candidatos L con los elementos que pueden añadirse a la solución en construcción: $L = \{1, 2, \dots, n\} \setminus M_k$. Para cada elemento i en L, el método calcula $\Delta f^k(i)$, que es la contribución marginal del elemento i a la solución en construcción si éste es añadido a M_k para obtener M_{k+1}. Luego se encuentra una lista restringida de candidatos RCL con los mejores elementos de L. En particular, el método ordena los elemento en L de acuerdo a su contribución marginal y forma RCL con sus α primeros elementos, donde α es un número entero aleatoriamente seleccionado en $[1, |L|]$. A continuación se selecciona $i^* \in RCL$ y se lo añade a la solución parcial, $M_{k+1} = M_k \cup \{i^*\}$. Cada fase de construcción termina cuando $|M_k| = m$. En la Figura 4 se muestra el seudocódigo del algoritmo.

1. Aleatoriamente seleccionar un elemento i^* en $V = \{1, 2, \dots, n\}$
2. Poner $M_1 = \{i^*\}$ y $k = 1$
3. Sea m el número de elementos a seleccionar desde el conjunto V

Mientras $(k < m)$:

4. Calcular $L = \{1, 2, \dots, n\} \setminus M_k$
5. Calcular $\Delta f^k(i)\ \ \forall i \in L$
6. Ordenar de mayor a menor los elementos en L de acuerdo a su valor $\Delta f^k(i)$
7. Seleccionar aleatoriamente $\alpha \in [1, |L|]$
8. Construir RCL con los primeros α elementos de L
9. Seleccionar aleatoriamente un elemento i^* en RCL
10. $M_{k+1} = M_k \cup \{i^*\}$
11. $k = k + 1$

Figura 4. Fase de construcción GRASP1

Adaptamos este método al problema Max-Mean de la siguiente manera: la diversidad promedio de un nuevo conjunto $M_{k+1} = M_k \cup \{i^*\}$ puede calcularse recursivamente como sigue:

$$\begin{aligned} div(M_{k+1}) &= \frac{\sum_{i<j, i,j \in M_{k+1}} d_{ij}}{k+1} \\ &= \frac{\sum_{i<j, i,j \in M_k} d_{ij} + \sum_{j \in M_k} d_{i^* j}}{k+1} \\ &= \frac{k\ div(M_k)}{k+1} + \frac{\sum_{j \in M_k} d_{i^* j}}{k+1} \end{aligned}$$

Por lo tanto, consideramos el cálculo de los aportes individuales de cada candidato a entrar en la solución como en la ecuación (5.1):

$$\Delta f^k(i) = div(M_{k+1}) - div(M_k) = \frac{-div(M_k)}{k+1} + \frac{\sum_{j \in M_k} d_{ij}}{k+1} \qquad (5.1)$$

Pero, como el algoritmo está diseñado para resolver el problema con $\boldsymbol{m}$ fijo, se selecciona aleatoriamente un valor para $\boldsymbol{m}$ en cada construcción; así, remplazamos el paso 3 del seudocódigo de la Figura 4 por la instrucción:

"Seleccionar aleatoriamente un número entero m en el intervalo $[2, m]$"

Después de que una solución M ha sido construida, se realiza la fase de mejora. Esta fase básicamente consiste en un mecanismo de intercambio en el cual un elemento seleccionado, de M, es remplazado por un elemento no seleccionado de $N \setminus M$. Se selecciona aleatoriamente ambos elementos y se los intercambia si el valor de la función objetivo mejora; sino, el intercambio es descartado. La fase de mejora termina luego de $maxiter$ intercambios consecutivos en los que no se haya detectado una mejora.

A todo este procedimiento que consiste de la construcción más la fase de mejora local, lo denominamos heurística GRASP1

HEURISTICA GRASP 2: La heurística GRASP_C2 (Duarte y otros, 2007), se reporta como una de las más eficientes para resolver el problema Max-Sum. Los autores introducen la distancia entre un elemento i^* y una solución parcial M_k, representada con $d_s(i^*, M_k)$, para adaptarla a nuestro nuevo problema debemos calcular recursivamente el valor de la dispersión del conjunto ampliado $M_{k+1} = M_k \cup \{i^*\}$ como sigue:

$$div_{sum}(M_{k+1}) = \sum_{i<j; i,j \in M_{k+1}} d_{ij} = \sum_{i<j; i,j \in M_k} d_{ij} + \sum_{j \in M_k} d_{i^* j}$$
$$= div_{sum}(M_k) + d_s(i^*, M_k)$$

Basados en estos elementos, pudimos adaptar esta heurística para resolver el problema Max-Sum. La lista restringida de candidatos, es calculada con los elementos $i \in L$ para los cuales $d_s(i, M_k)$ es mayor que un umbral. Más específicamente:

$$RCL = \{i \in L : d_s(i, M_k) \geq div_{sum\_min}(M_k) + \alpha(\overline{div})\} \qquad (5.2)$$

Donde: $\overline{div} = div_{sum\_max}(M_k) - div_{sum\_min}(M_k)$

$$div_{sum\_max}(M_k) = \max_{i \in L} d_s(i, M_k) \text{ ; y,}$$
$$div_{sum\_min}(M_k) = \min_{i \in L} d_s(i, M_k)$$

Para que la adaptación quede completa debemos incluir: 1. Seleccionar m aleatoriamente al inicio de cada construcción, y 2. Dividir el valor obtenido en la construcción para el valor de m generado. La Figura 5 se muestra el seudocódigo asociado.

1. Seleccionar aleatoriamente un elemento i^* en $V = \{1, 2, \ldots, n\}$
2. Poner $M_1 = \{i^*\}$ y $k = 1$
3. Seleccionar m aleatoriamente en el intervalo $[2, m]$

Mientras $(k < m)$:

4. Calcular $L = \{1, 2, \ldots, n\} \setminus M_k$
5. Calcular $d_s(i, M_k), \forall i \in L$, $div_{sum\_min}(M_k)$ y $div_{sum\_max}(M_k)$
6. Construir la lista RCL con los elementos que satisfagan la ecuación (3.1)
7. Seleccionar aleatoriamente un elemento i^* en RCL
8. $M_{k+1} = M_k \cup \{i^*\}$
9. $k = k + 1$

Figura 5. Fase de construcción del algoritmo GRASP_C2 adaptado al problema Max-Mean

Luego se realiza la fase de búsqueda local, que también realiza intercambios como en GRASP1, pero en lugar de seleccionar aleatoriamente los dos elementos para intercambiarlos, el método selecciona los elementos con la menor contribución a la función objetivo en la solución actual y trata de intercambiarlo con un elemento no seleccionado. Así, para cada $i \in M$ calculamos:

$$d_s(i, M) = \sum_{j \in M} d_{ij}$$

Y consideramos el elemento $i^* = \min_{i \in M}\{d_s(i, M)\}$. Luego, el método explora el conjunto de elementos no seleccionados en busca del primer intercambio de i^* que mejora el valor de M.
A la fase de construcción adaptada que se mostró en la Figura 5, más esta fase de búsqueda local, la denominamos heurística GRASP2.

6. Resultados numéricos

Se implementaron las heurísticas GRASP1, GRASP2, GRASP3 y GRASP3+PR en Mathematica V.8[1] y se los aplicó a los problemas de prueba de tamaño pequeño, mediano y grande, de tipo I y de tipo II. También se procesó la formulación lineal del problema Max-Mean con métodos exactos con el solver Cplex V.12 en los problemas de tamaño pequeño ($n = 30$).
En la Tabla 1 se observan los resultados para 20 problemas de prueba de tamaño pequeño de tipo I y de tipo II, GRASP1 y GRASP2 encontraron el óptimo, 4 y 5 veces respectivamente, mientras que GRASP3 encontró siempre el óptimo. El tiempo de procesamiento en todas las heurísticas es menor que el procesamiento exacto, que ocupó sobre los 100 segundos en promedio.
Para problemas de tamaño n=35 Cplex requirió en promedio 719.51 segundos, y para n=50 en 5 horas de procesamiento Cplex no pudo obtener la solución óptima, lo cual indica que es imposible resolver de manera exacta, en tiempos razonables, problemas medianos o grandes.

Tabla 1. Resultados de las heurísticas en problemas pequeños $n = 30$

		GRASP1	GRASP2	GRASP3	Exacta
Tipo I	Valor	1.842227	1.841831	1.874955	1.874955
	m	10.9	11.3	10.7	10.7
	# veces óptimo	4	5	10	10
	GAP	1.812%	1.926%	0%	0%
	Tiempo CPU (s)	7.9091	3.3088	0.4135	102.303
Tipo II	Valor	2.345667	2.35521	2.383	2.383
	m	11.4	11.4	10.8	10.8
	# veces óptimo	4	5	10	10
	GAP	1.682%	1.268%	0%	0%
	Tiempo CPU (s)	8.1357	3.1777	0.3634	182.176

Ya que no se puede resolver de manera exacta problemas de tamaño mediano o grande, aplicamos las heurísticas GRASP1, GRASP2 y GRASP3 para problemas medianos, los resultados se muestran en la Tabla 2. En todos los casos GRASP3 permitió obtener mejores soluciones, GRASP2 fue mejor que GRASP1, GRASP3 no sólo permitió obtener mejores resultados en términos de su valor objetivo, su tiempo de procesamiento también fue inferior. Este comportamiento de los algoritmos se repitió tanto en problemas tipo I como tipo II.

Por último se aplicaron los algoritmos a problemas grandes, también el algoritmo GRASP3+PR, este último fue más eficiente, pues permitió encontrar soluciones con mejor valor objetivo y en tiempos razonables. En la Tabla 3 se presentan estos resultados. El procedimiento de rencadenamiento de trayectorias permitió mejorar el valor de la función objetivo en un 1.07%, en promedio.

[1] Mathematica es un software de procesamiento numérico que es también lenguaje de programación.

Tabla 2. Resultados en los ejemplos de prueba con $n = 150$

		GRASP1	GRASP2	GRASP3
Tipo I	Valor	3.9078	3.9596	4.3236
	m	49.4	45.7	44
	# veces mejor	0	0	10
	Desviación	9.63%	8.41%	0%
	Tiempo CPU (s)	241.6	61.5	26.4
Tipo II	Valor	4.943	5.223	5.684
	m	54.6	52.2	46.1
	# veces mejor	0	0	10
	Desviación	13.06%	8.14%	0%
	Tiempo CPU (s)	250.0	80.1	22.9

Tabla 3. Resultados en los ejemplos de prueba con $n = 500$

		GRASP1	GRASP2	GRASP3	GRASP3+PR
Tipo I	Valor	6.6796	7.0163	7.71370405	7.78896586
	m	154.4	157.6	139.4	145.2
	# veces mejor	0	0	0	10
	Desviación	14.22%	9.91%	1.07%	0.00%
	Tiempo CPU (s)	984.56882	950.821574	717.337083	669.417148
Tipo II	Valor	8.898207	9.26865	10.2957	10.43732
	m	186.1	170.3	143.2	144.4
	# veces mejor	0	0	0	10
	Desviación	14.74%	11.18%	1.35%	0.00%
	Tiempo CPU (s)	736.79645	708.2459	662.422	679.641

7. Conclusiones

El problema de maximización Max-Mean es un problema computacionalmente difícil que se plantea en el contexto de los problemas de la dispersión equitativa y de la diversidad máxima. Este problema nos ha servido como un buen caso de prueba para desarrollar las nuevas estrategias de búsqueda que proponemos para resolver este problema combinatorio. En particular hemos desarrollado un algoritmo GRASP constructivo basado en una combinación no estándar de codicia y aleatoriedad, una estrategia de búsqueda local basada en la metodología de vecindades descendientes variables, la cual incluye tres tipos diferentes de vecindades, y una fase de post procesamiento basada en rencadenamiento de trayectorias.

Al comparar la eficiencia de la heurística que proponemos con otros procedimientos previos adaptados de otros autores para problemas similares favorece nuestra propuesta.

El modelo Max-Mean se adapta plenamente a problemas de diversidad en las que los elementos son personas, ya que se puede identificar entre ellas grados de afinidad o desafinidad a través del signo de los coeficientes d_{ij}, con lo que el modelo analizado es un excelente marco de trabajo para tratar este tipo de problemas. En el contexto de resolver un problema, el valor de una persona depende de su habilidad para mejorar la decisión colectiva, pues la contribución de esta persona depende en gran medida de las perspectivas y heurísticas de las otras personas que conforman el equipo de trabajo. La diversidad en el enfoque de la solución a un problema respecto a las otras personas es un importante predictor de su valor, y a la larga puede ser más relevante que su habilidad individual para resolver el problema por su cuenta. Así, para estimar la contribución potencial de una persona en un equipo de trabajo, es más importante hacer énfasis en medir cómo esta persona piensa diferente, antes que en estimar la magnitud de su habilidad.

BIBLIOGRAFIA:

Duarte, A., y R. Martí. «Tabu Search and GRASP for the Maximum Diversity Problem.» *European Journal of Operational Research* 178 (2007): 71 - 84.

Ercut, E., y S. Neuman. «Analytic Models for locating undesirable facilities.» *European Journal of Operational Research* 40 (1989): 275-291.

Erkut, E. «The discrete p-dispersion problem.» *European Journal of Operational Research* 46 (1990): 48-60.

Glover, F., C. C. Kuo, y K. Dhir. «Heuristic Algorithms for the Maximum Diversity Problem.» *Journal of Information and Optimization Sciences* 19, nº 1 (1998): 109 - 132.

Hansen, P., y N. Mladenovic. «Variable neighborhood search.» En *Handbook in Metaheuristics*, editado por F. Glover y G. Kochenberger, 145-184. 2003.

Kuo, M., F. Glover, y K. Dhir. «Analyzing and modeling the maximum diversity problem by zero-one programming.» *Decision Sciences*, nº 24 (1993): 1171 - 1185.

Lozano, M., D. Molina, y C. García-Martínez. «Iterated greedy for the maximum diversity problem.» *European Journal of Operational Research*, 2011.

Lu Hong, Scott E. Page. «Groups of diverse problem solvers can outperform groups of high-ability problem solvers.» *PNAS* 101, nº 46 (2004): 16385-16389.

Martí, R. *Instances MMDPLIB.* 2011. http://heur.uv.es/optsicom/mmdp/ (último acceso: 24 de 05 de 2011).

Martí, R., y F. Sandoya. «GRASP and path relinking for the equitable dispersion problem.» *Computers and Operations Research* http://dx.doi.org/10.1016/j.cor.2012.04.005 (2012).

Meinl, Thorsten. *Maximum-Score Diversity Selection.* primera. Südwestdeutscher Verlag, 2010.

Page, S. *The Difference: How the Power of Diversity Creates better Groups, Firms, Schools, and Societies.* New Jersey: Princenton University Press, 2007.

Prokopyev, O., N. Kong, y D. Martínez-Torres. «The equitable dispersion problem.» *European Journal of Operational Research*, nº 197 (2009): 59 - 67.

Resende, M., R. Martí, M. Gallego, y A. Duarte. «GRASP with path relinking for the max-min diversity problem.» *Computers and Operations Research*, nº 37 (2010): 498 - 508.

Resende, M., y R. Werneck. «A hybrid heuristic for the p-median problem.» *Journal of heuristics* 10, nº 1 (2004): 59-88.

Santini, Simone, y Ramesh Jain. «Similarity Measures.» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999.