Computers & Operations Research 40 (2013) 3091–3099

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

GRASP and path relinking for the equitable dispersion problem

Rafael Martí [a,*], Fernando Sandoya [b]

[a] Departamento de Estadística e Investigación Operativa, Facultad de Ciencias Matemáticas, Universidad de Valencia, Spain
[b] Doctorado en Investigación de Operaciones, Universidad Nacional Autónoma de México, México D.F., Instituto de Ciencias Matemáticas, Escuela Superior Politécnica del Litoral, Guayaquil-Ecuador

ARTICLE INFO

Available online 21 April 2012

Keywords:
Metaheuristics
Diversity
GRASP with Path Relinking

ABSTRACT

The equitable dispersion problem consists in selecting a subset of elements from a given set in such a way that a measure of dispersion is maximized. In particular, we target the Max-Mean dispersion model in which the average distance between the selected elements is maximized. We first review previous methods and mathematical formulations for this and related dispersion problems and then propose a GRASP with a Path Relinking in which the local search is based on the Variable Neighborhood methodology. Our method is specially suited for instances in which the distances represent affinity and are not restricted to take non-negative values. The computational experience with 120 instances shows the merit of the proposed procedures compared to previous methods.

1. Introduction

The problem of maximizing diversity deals with selecting a subset of elements from a given set in such a way that the diversity among the selected elements is maximized [3]. Different models have been proposed to deal with this combinatorial optimization problem. All of them require a diversity measure, typically a distance function in the space where the objects belong. The definition of this distance between elements is customized to specific applications. As described in Glover et al. [4], these models have applications in plant breeding, social problems, ecological preservation, pollution control, product design, capital investment, workforce management, curriculum design and genetic engineering.

The most studied model is probably the *Maximum Diversity Problem* (MDP) also known as the *Max-Sum Diversity Problem* [2] in which the sum of the distances between the selected elements is maximized. Heuristics [4] and meta-heuristics [1] have been proposed for this model. The *Max-Min Diversity Problem* (MMDP), in which the minimum distance between the selected elements is maximized, has been also well documented in recent studies [11].

Prokopyev et al. [10] introduced two additional models to deal with maximizing diversity in the context of equitable models. In particular, the *Maximum Mean Dispersion Problem* (Max-Mean DP) consists of maximizing the average distance between the selected elements, while in the *Minimum Differential Dispersion Problem* (Min-Diff DP) we minimize the difference between the maximum sum and the minimum sum of the distances to the other selected elements. These authors proved that the Max-Mean DP is strongly NP-hard if the distances (diversity measure) take both positive and negative values. On the other hand, the Min-Diff problem is strongly NP-hard regardless the distance values.

Most of the previous works on diversity limit themselves to problems with non-negative distances. However, as described in Glover et al. [4], the diversity measure can be something in the nature of an affinity relationship, which expresses a relative degree of attraction between the elements as arises in settings with a behavioral component. Typical examples are architectural space planning and analysis of social networks. In such domains we do not require the "distance values" to satisfy distance norms or conditions since they only represent a measure to reflect similarity/diversity.

As Page [9] states in his book, *The difference—How the power of diversity creates better groups, firms, schools, and societies* (2007), "Diverse perspectives and tools enable collections of people to find more and better solutions and contribute to overall productivity". As a result, the problem of identifying diverse groups of people becomes a key point in large firms and institutions. We can find different measures of similarity/diversity applied to groups of people. For example, Lee et al. [8] stated that "the cosine similarity is a popular measure of similarity". Given two elements x,y (individuals) with p attributes, $x=(x_1,x_2,\ldots,x_p)$, $y=(y_1,y_2,\ldots,y_p)$, their cosine similarity cs_{xy} is computed as:

$$cs_{xy}=\frac{\sum_{k=1}^{p}x_k y_k}{\sqrt{\sum_{k=1}^{p}x_k^2}\sqrt{\sum_{k=1}^{p}y_k^2}}$$

The cosine similarity between element x and element y can be viewed as the angle between their attribute vectors. The smaller

* Corresponding author.
E-mail addresses: Rafael.Marti@uv.es (R. Martí), fsandoya@espol.edu.ec (F. Sandoya).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.04.005

the angle is, the greater the degree of similarity between the elements. It takes values in $[-1,1]$ reflecting the affinity between the individuals. Alternatively, the modified difference measure md_{xy} computes the absolute difference between the pairs of attributes and assigns a -1 when they are equal since they are opposite to diversity:

$$md_{xy}=\frac{\sum_{k=1}^{p}\delta(x_k,y_k)}{p} \quad \text{where} \quad \delta(x_k,y_k)=\begin{cases}-1 & \text{if } x_k=y_k\\ |x_k-y_k| & \text{if } x_k\neq y_k\end{cases}$$

In this paper we consider the optimization of the Max-Mean DP to target general instances in which the "distances" can take positive and negative values and do not necessarily satisfy the usual distance properties, such as the triangular inequality. We target two types of instances, Type I representing the affinity between individuals in a social network (with affinity values in $[-10,10]$), and Type II with random numbers in $[-10,-5]\cup[5,10]$, reflecting the polarization that occurs when people get together in groups, in which we can identify clusters of individuals, with a high attraction within clusters and a high repulsion between clusters, and with no room for indifference. Note that the Max-Mean Dispersion Problem is polynomially solvable if all the distances are non-negative, but, as mentioned above, it is strongly NP-hard if they can take positive and negative values [10].

In mathematical terms, given a set N of n elements and d_{ij} the affinity between any elements i and j, the Max-Mean DP consists of selecting a subset M of N in such a way that the dispersion mean $dm(M)$, in terms of the affinity values, is maximized.

$$dm(M)=\frac{\sum_{i<j;i,j\in M}d_{ij}}{|M|}$$

The $dm(M)$ value reflects an equity measure based on an average dispersion. The Max-Mean DP can be trivially formulated with binary variables as:

$$\text{Max}\ \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}d_{ij}x_ix_j}{\sum_{i=1}^{n}x_i}$$
$$s.t.\ \sum_{i=1}^{n}x_i\geq 2$$
$$x_i\in\{0,1\}\, i=1,2,\ldots,n$$

where variable x_i takes the value 1 if element i is selected and 0 otherwise. Note that the number of selected elements, $|M|$, is not set *a priori* as in the rest of diversity models, such as the MDP or the MMDP, where it is pre-specified as a problem constraint. On the contrary, in the Max-Mean model we do not impose the number of elements that have to be selected, but it is obtained as an output of the method when maximizing $dm(M)$. We therefore cannot directly apply a solving method for other diversity model to the Max-Mean DP, unless we repeatedly apply it for any possible value of $m=|M|$.

2. Previous heuristics

Prokopyev et al. [10] introduced several models to deal with the equitable dispersion problem. The authors proposed a GRASP for the Max-MinSum variant in which for each selected element (in M), they compute the sum of the distances to the other selected elements (also in M) and then calculate the minimum of these values. The objective of the Max-MinSum model is to maximize this minimum sum of distances.

In their GRASP algorithm, Prokopyev et al. [10] consider M_k, a partial solution with k selected elements. Each construction phase of GRASP starts by randomly selecting an element, which constitutes the initial set M_1. Then, in each iteration, the method computes a candidate list L with the elements that can be added to the partial solution under construction: $L=\{1,2,\ldots,n\}\backslash M_k$. For each element i in L, the method computes its marginal contribution, $\Delta f^k(i)$, if it is added to M_k to obtain M_{k+1}. As it is customary in GRASP constructions, a restricted candidate list RCL is computed with the best elements in L. In particular, the method orders the elements in L according to their marginal contribution and forms RCL with the first α elements (where α is an integer randomly selected in $[1,|L|]$). Then, it randomly selects (according to a uniform distribution) an element i^* in RCL and adds it to the partial solution: $M_{k+1}=M_k\cup\{i^*\}$. Each construction phase terminates when the pre-established number of selected elements m is reached ($|M_k|=m$). Fig. 1 outlines the pseudo-code of this method.

We can easily adapt the method above, originally proposed for the Max-MinSum, to the Max-Mean DP model. Specifically, given a partial solution M_k, its value for this later model, dispersion mean $dm(M_k)$, is computed as:

$$dm(M_k)=\frac{\sum_{i<j;i,j\in M_k}d_{ij}}{k}$$

Then, the dispersion mean value of $M_{k+1}=M_k\cup\{i^*\}$ can be incrementally computed as:

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

We therefore consider in the algorithm above,

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

The algorithm in Fig. 1 considers the Max-MinSum model for which the value of m is fixed. To adapt it to the Max-Mean DP model, we can simply select m at random in each construction; thus obtaining solutions for all possible values of m when the method is run for a relative large number of times. So we replace Step 3 in Fig. 1 with the following instruction:

– Randomly select an integer m in $[2,n]$.

We will call this method C1a. An alternative way to adapt the algorithm in Fig. 1 to the Max-Mean DP model, consists of enumerating all the possible values of m. In other words, we construct the first solution with $m=2$; then, the next solution with $m=3$, and so on until we construct in iteration $n-1$ a solution with all the elements selected. In the next iteration we reset the value of m and start again to construct a solution with

1. Randomly select an element $i^\star$ in $N=\{1,2,\ldots,n\}$.
2. Make $M_1=\{i^\star\}$ and $k=1$.
3. Let m be the number of elements to select from N.

While ($k<m$)

4. Compute $L=\{1,2,\ldots,n\}\setminus M_k$
5. Compute $\Delta f^k(i)\ \ \forall i\in L$
6. Order the elements in L according to their Δf^k value
7. Randomly select α in $[1,|L|]$
8. Construct RCL with the first α elements in L
9. Randomly select an element $i^\star$ in RCL
10. $M_{k+1}=M_k\cup\{i^\star\}$
11. $k=k+1$

Fig. 1. GRASP construction phase.

$m=2$, following in the same way (incrementing m) in the next iterations. We will call this method C1b.

After a solution M has been constructed, an improvement phase is performed. It basically consists of an exchanging mechanism in which a selected element (in M) is replaced with an unselected one (in $N\backslash M$). The method randomly selects both elements and exchanges them if the objective is improved; otherwise, the selection is discarded. The improvement phase terminates after 100 iterations without any improvement. When we couple C1a with this improvement method, we call the entire method GRASP1a, and similarly we call GRASP1b when we apply C1b as the constructive method.

Duarte and Martí [1] proposed different heuristics for the Max-Sum diversity problem, MDP. In particular the authors adapted the GRASP methodology to maximize the sum of the distances among the selected elements. Given a partial solution M_k, its value on this model, dispersion sum $ds(M_k)$, is computed as

$$ds(M_k)=\sum_{i<j;i,j\in M_k} d_{ij}$$

The authors introduced the distance between an element i^* and a partial solution M_k, $d(i^*,M_k)$, to incrementally compute the dispersion value of $M_{k+1}=M_k\cup\{i^*\}$, as

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

Based on these elements, they proposed a GRASP construction, called GRASP_C2, to solve the Max-Sum diversity problem. In their case, the restricted candidate list, RCL, is computed with those elements $i\in L$ with $d(i,M_k)$ larger than a threshold value. Specifically,

$$RCL=\{i\in L: d(i,M_k)\geq d_{min}(M_k)+\alpha(d_{max}(M_k)-d_{min}(M_k))\}$$

where $d_{min}(M_k)=min_{i\in L}d(i,M_k)$, $d_{max}(M_k)=max_{i\in L}d(i,M_k)$ and the value of α is set to 0.5. As in C1a, we can easily adapt this method to solve the Max-Mean DP model by selecting m at random in each construction and dividing the value of the constructed solution by m. Let C2a be the resulting method. Fig. 2 shows the associated pseudo code.

Similarly as we did in C1b, we can enumerate here all the possible values of m, generating a solution for each value. Let C2b be this adaptation of the constructive method of GRASP_C2 to solve the Max-Mean DP model.

The local search post-processing of GRASP_C2 also performs exchanges (as GRASP1). However, instead of randomly selecting the two elements for exchange, the method focuses on the selected element with the lowest contribution to the value of the current solution and tries to exchange it with an unselected one. Specifically, for each $i\in M$ we compute

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

and consider the element i^* with minimum $d(i,M)$. Then, the method scans the unselected elements in search for the first exchange of i^* to improve the value of M. Note that when an improving move is identified, it is performed without examining the remaining elements in $N\backslash M$ This improvement phase is performed as long as the solution is improved. When we couple C2a with this improvement method, we call the entire method GRASP2a, and similarly we call GRASP2b when we apply C2b as the constructive method.

1. Select an element $i^\star$ at random in $N=\{1,2,\ldots,n\}$.
2. Make $M_1=\{i^\star\}$ and $k=1$.
3. Select m at random in $[2,n]$.

While ($k<m$)

4. Compute $L=\{1,2,\ldots,n\}\setminus M_k$
5. Compute $d(i,M_k)\ \forall i\in L$, $d_{min}(M_k)$ and $d_{max}(M_k)$
6. Construct RCL
7. Randomly select an element $i^\star$ in RCL
8. $M_{k+1}=M_k\cup\{i^\star\}$
9. $k=k+1$

Fig. 2. C2a constructive method.

3. Mathematical models

Prokopyev et al. [10] proposed the following linearization of the mathematical formulation for the Max-Mean DP shown in the introduction:

$$\text{Max}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}z_{ij}$$
$$\text{Subject to}: y-z_i\leq 1-x_i, z_i\leq y, z_i\leq x_i, z_i\geq 0,\quad i=1,\ldots,n$$
$$y-z_{ij}\leq 2-x_i-x_j, z_{ij}\leq y, z_{ij}\leq x_i,$$
$$z_{ij}\leq x_j, z_{ij}\geq 0, 1\leq i<j\leq n;$$
$$\sum_{i=1}^{n} x_i\geq 2;\ \sum_{i=1}^{n} z_i=1, x_i\in\{0,1\}, 1\leq i\leq n$$

On the other hand, Kuo et al. [7] proposed the following mathematical programming formulation of the MDP

$$\text{Max}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}z_{ij}$$
$$\text{Subject to}: \sum_{i=1}^{n} x_i=m$$
$$z_{ij}\geq x_i+x_j-1, z_{ij}\leq x_i, z_{ij}\leq x_j, 1\leq i\leq j\leq n;$$
$$x_i\in\{0,1\},\quad 1\leq i\leq n$$
$$z_{ij}\in\{0,1\},\quad 1\leq i\leq j\leq n$$

It is clear that to obtain the optimum solution of the Max-Mean DP, we can solve the MDP for any possible value of m (*i.e.*, 2,3,…,n) and divide the value of each solution obtained by the corresponding value of m. The best value, across the $n-1$ Max-Sum problems solved, is the optimum of the Max-Mean DP. Fig. 3 shows the Max-Mean value (y-axis) of each Max-Sum problem solved ($m=2,\ldots,50$ on the x-axis) on an instance with $n=50$ elements, in which the distances between the elements were randomly generated in $[-10,10]$.

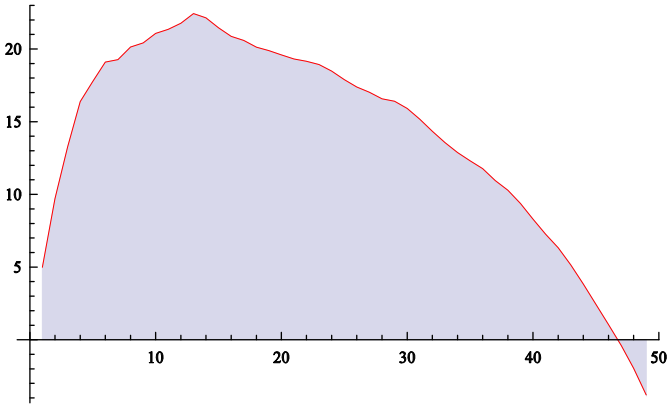

Fig. 3. Max-Mean value for Max-Sum solutions.

Fig. 3 shows that the Max-Mean value of the Max-Sum solution increases as m increases from 2 to 14. Then, it decreases in the rest of the range (from $m=15$ to 50). We therefore conclude that the optimum of the Max-Mean DP is reached in this instance when $m=14$. We have observed the same pattern (a concave function approximately) in all the examples tested with positive and negative distances randomly generated. We will consider this pattern to design an efficient GRASP algorithm in the next section.

On the other hand, when distances are non-negatives and satisfy the triangle inequality the Max-Mean is an increasing function and the best value is obtained when all the elements have been selected ($m=n$). The following proposition proves that the Max-Mean value of a given solution M, $dm(M)$, is lower than the value of the solution $M\cup\{k\}$ for any element k non-selected in M. Therefore, a solution with $m<n$ selected elements cannot provide the optimum of the Max-Mean DP because we could improve it by simply adding an extra element. Then we conclude that, under these conditions, the optimum has the n elements selected.

Proposition. *Given an instance where the distances are non-negatives, symmetric ($d_{ij}=d_{ji}$) and satisfy the triangle inequality, and given a solution M with $dm(M)$ its Max-Mean value, then $dm(M)<dm(M\cup\{k\})$ for any element k non-selected in M.*

Proof. For any $i,j\in M$ and given $k\notin M$ the triangle inequality gives $d_{ij}\le d_{ik}+d_{jk}$. Adding it for all pairs in M we obtain:

$$\sum_{\substack{i,j\in M\\ i<j}} d_{ij} \le \sum_{\substack{i,j\in M\\ i<j}} d_{ik}+d_{jk}$$

Note that in the right hand side of the expression above d_{ik} appears $(m-1)$ times, where $m=|M|$. Then,

$$\sum_{\substack{i,j\in M\\ i<j}} d_{ij} \le \sum_{\substack{i,j\in M\\ i<j}} d_{ik}+d_{jk} = (m-1)\sum_{i\in M} d_{ik} < m\sum_{i\in M} d_{ik}$$

Dividing by m we obtain,

$$\frac{1}{m}\sum_{\substack{i,j\in M\\ i<j}} d_{ij} < \sum_{i\in M} d_{ik}$$

Adding $\sum_{\substack{i,j\in M\\ i<j}} d_{ij}$ to both sides,

$$\frac{m+1}{m}\sum_{\substack{i,j\in M\\ i<j}} d_{ij} < \sum_{\substack{i,j\in M\\ i<j}} d_{ij}+\sum_{i\in M} d_{ik} = \sum_{\substack{i,j\in M\cup\{k\}\\ i<j}} d_{ij}$$

Dividing by $m+1$ we obtain,

$$dm(M)=\frac{1}{m}\sum_{\substack{i,j\in M\\ i<j}} d_{ij} < \frac{1}{m+1}\sum_{\substack{i,j\in M\cup\{k\}\\ i<j}} d_{ij} = dm(M\cup\{k\}).$$

Note that in the proposition above we cannot relax the triangle inequality condition. The following example with $n=4$ and distance matrix

$$d=\begin{pmatrix} 0 & 20 & 18 & 1\\ 20 & 0 & 20 & 2\\ 18 & 20 & 0 & 1\\ 1 & 2 & 1 & 0\end{pmatrix},$$

not verifying the triangle inequality, presents the max-mean optimum when elements 1, 2 and 3 are selected, with a value of 19.33. When all the elements are selected the Max-Mean value is 15.5.

4. A new method and adaptations of existing methods

In this section we describe our proposal to obtain high quality solutions to the Max-Mean DP. It consists of a GRASP (construction plus local search) with a Path Relinking post-processing. We also propose a new adaptation of existing methods.

4.1. GRASP construction

All the previous methods described in Section 2 *a priori* set the number of selected elements. However, from the results shown in Fig. 3, we can design a new constructive method in which we add elements to the partial solution under construction as long as the Max-Mean value improves, and when this value starts to decrease, we stop the construction. In this way, the method selects by itself the value of m, which seems adequate to this problem.

On the other hand, previous GRASP constructions for diversity problems implemented a typical GRASP construction [12] in which, first, each candidate element is evaluated by a greedy function to construct the Restricted Candidate List (RCL) and then an element is selected at random from RCL. However, more recent studies [13] have shown that an alternative design in which we first apply the randomization and then the greediness can obtain improved outcomes. In particular, in our constructive method for the Max-Mean DP, we first randomly choose candidates and then evaluate each candidate according to the greedy function, selecting the best candidate.

In mathematical terms, given a partial solution M_k with k selected elements, the candidate list CL is formed with the $n-k$ non-selected elements. The restricted candidate list, RCL contains a fraction $\alpha(0<\alpha\le 1)$ of the elements in CL selected at random. Each element $i\in RCL$ is evaluated according to the change in the objective function:

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

The method selects the best candidate $i^{\star}$ in RCL if it improves the current solution ($eval(i^*)>0$) and adds it to the partial solution, $M_{k+1}=M_k\cup\{i^*\}$; otherwise the method stops. Fig. 4 shows a pseudo-code of the method that we will call GRASP3.

4.2. Local search

The GRASP construction usually does not obtain a local optimum and it is customary in GRASP to apply a local search method to the solution constructed. As shown in Section 2, previous local search methods for diversity problems limit themselves to exchange a selected with an unselected element, keeping constant the number m of selected elements. Since we do not have this size constraint in the Max-Mean DP and we admit solutions with any value of m, we can consider an extended

1. Select an element i^* at random in $N=\{1,2,\ldots,n\}$.
2. Make $M_1=\{i^*\}$, $k=1$ and $improve=1$.

While ($improve=1$)
3. Compute $CL=\{1,2,\ldots,n\}\setminus M_k$
4. Construct RCL with $\alpha|CL|$ elements randomly selected in CL
5. Compute $eval(i)=dm(M_k\cup\{i\})-dm(M_k)\ \ \forall i\in RCL$
6. Select the element i^* in RCL with maximum $eval$ value

If ($eval(i^*)>0$)
7. $M_{k+1}=M_k\cup\{i^*\}$
8. $k=k+1$

Else
9. $improve=0$

Fig. 4. GRASP3 construction phase.

neighborhood. Specifically, based on the Variable Neighborhood Descent (VND) methodology [6], we consider the combination of three neighborhoods in our local search procedure:

- N_1: Remove an element from the current solution, thus reducing the number of selected elements by one unit.
- N_2: Exchange a selected element with an unselected one, keeping constant the number of selected elements.
- N_3: Add an unselected element to the set of selected elements, thus increasing its size by one unit.

Given a solution, M_m, the local search first tries to obtain a solution in N_1 to improve it. If it succeeds, and finds M'_{m-1} with $dm(M'_{m-1}) > dm(M_m)$, then we apply the move and consider M'_{m-1} as the current solution. Otherwise, the method resorts to N_2 and searches for the first exchange that improves M_m. If it succeeds, and finds M'_m with $dm(M'_m) > dm(M_m)$, then we apply the move and consider M'_m as the current solution. In any case, regardless that we found the improved solution in N_1 or in N_2, in the next iteration the method starts scanning N_1 to improve the current solution. If neither N_1 nor N_2 is able to contain a solution better than the current solution, we finally resort to N_3. If the method succeeds, finding M'_{m+1} with $dm(M'_{m+1}) > dm(M_m)$, then we apply the move and consider M'_{m+1} as the current solution (and come back to N_1 in the next iteration). Otherwise, since none of the neighborhoods contain a solution better that the current one, the method stops.

Given a solution M_m, we compute the contribution of each selected element i, as well as the potential contribution of each unselected element $i$as:

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

Then, when we explore N_1 to remove an element from M_m, we scan the selected elements in the order given by d, where the element with the smallest value comes first. Similarly, when we explore N_2 we explore the selected elements in the same order but the unselected ones in the "reverse order" (i.e., we first consider the unselected elements with larger potential contribution). Finally, when we explore N_3 the unselected elements, considered to be added to the current solution, are explored in the same way as in N_2, in which the element with the largest potential contribution is explored first. Fig. 5 shows a pseudo-code of the entire local search algorithm.

4.3. Path Relinking

The Path Relinking algorithm (PR) [5] operates on a set of solutions, called Elite Set (*ES*), constructed with the application of a previous method. In this paper we apply GRASP to build the Elite Set considering both quality and diversity. Initially *ES* is empty, and we apply GRASP for $b=|ES|$ iterations and populate it with the solutions obtained (ordering the solutions in *ES* from the best x^1 to the worst x^b). Then, in the following GRASP iterations, we test whether the generated (constructed and improved) solution x', qualify or not to enter *ES*. Specifically, if x' is better than x^1, it enters in the set. Moreover, if it is better than x^b and it is sufficiently different from the other solutions in the set ($d(x',ES) \geq dth$), it also enters *ES*. To keep the size of *ES* constant and equal to b, when we add a solution to this set, we remove another one. To maintain the quality and the diversity, we remove the closest solution to x' in *ES* among those worse than it in value.

Given two solutions, x and y, interpreted as binary vectors with n variables, where variable x_i (y_i) takes the value 1 if element i is selected and 0 otherwise, the distance between them can be computed as $d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$ and the distance between a solution x' and the set *ES*,$d(x',ES)$, can therefore be computed as the sum of the distances between x' and all the elements in *ES*.

Given two solutions $x,y \in ES$, the path relinking procedure $PR(x,y)$ starts with the first solution x, called the initiating solution, and gradually transforms it into the final one y called the guiding solution. At each iteration we consider to remove an elements in x not present in y, or to add an element in y not present in x. The method selects the best one among these candidates, creating the first intermediate solution, $x(1)$. Then, we consider to remove an element in $x(1)$ not present in y, or to add an element in y not present in $x(1)$. The best of these candidates is the second intermediate solution $x(2)$. In this way we generate a path of intermediate solutions until we reach y. The output of the PR algorithm is the best solution, different from x and y, found in the path. We submit this best solution to the improvement method. Fig. 6 shows a pseudo-code of the entire GRASP with Path Relinking algorithm in which we can see that we apply both $PR(x,y)$ and $PR(y,x)$ to all the pairs x,y in the elite set *ES*.

1. Let M_m be the initial solution.
2. Compute $d(i, M_m)$ for all element i.

While ($improve = 1$)
3. Explore the selected elements in the d-order to be unselected.
If(There is a move in N_1 improving the current solution)
4. Perform the first improving move (N_1)
5. Let M'_{m-1} the current solution
Else
6. Search for the exchanges in the d-order.
If(There is a move in N_2 improving the current solution)
7. Perform the first improving move (N_2)
8. Let M'_m the current solution
Else
9. Explore the unselected elements in reverse d-order.
If (There is a move in N_3 improving the current solution)
10. Perform the first improving move (N_3)
11. Let M'_{m+1} the current solution
Else
12. Improve=0
13. Update the ds-values

Fig. 5. Local search algorithm.

1. Set *GlobalIter* equal to the number of global iterations.
2. Apply the GRASP method (construction plus improvement) for $b=|ES|$ iterations to populate ES={ x^1, x^2, ..., x^b }.
3. *iter*=*b*+1.
While(*iter* ≤ *GlobalIter*)
4. Apply the construction phase of GRASP ⇒ x.
5. Apply the local search phase of GRASP to $x \Rightarrow x'$.
If ($f(x') > f(x^1)$ or ($f(x') > f(x^b)$ and $d(x',ES) \geq dth$))
6. Let x^k be the closest solution to x' in ES with $f(x') > f(x^k)$.
7. Add x' to ES and remove x^k.
8. Update the order in ES (from the best x^1 to the worst x^b).
9. Let $x^{best} = x^1$.

For(i=1 to b-1 and j=i+1 to b)
10. Apply $PR(x^i,x^j)$ and $PR(x^j,x^i)$, let x be the best solution found
11. Apply the local search phase of GRASP to $x \Rightarrow x'$.
If($f(x') > f(x^{best})$)
12. $x^{best} = x'$.
13. Return x^{best}.

Fig. 6. GRASP with Path Relinking.

4.4. New adaptations of existing methods

The principle applied in our constructive method described in Section 4.1, can also be used to adapt the previous methods for similar problems considered in Section 2. In particular, instead of setting *a priori* the number of selected elements, we can add elements to the partial solution under construction as long as the Max-Mean value improves, and when this value starts to decrease, we stop the construction. Let C1c be this adaptation to the Max-Mean DP of the constructive method in Prokopyev et al. [10] originally proposed for the Max-MinSum. Similarly, let C2c be this adaptation of the method in Duarte and Martí [1] originally proposed for the Max-Sum model.

5. Computational experiments

This section describes the computational experiments that we performed to first study the search parameters of our proposed procedure and then compare it to state-of-the-art methods for solving the maximum mean dispersion problem. Our GRASP with Path Relinking implementation follows the framework described in Section 4 (Fig. 6). For this comparison, we use the following two sets of instances where, as explained above, the distances can take both negative and positive values and do not satisfy the triangular inequality.

Type I This data set consists of 60 symmetric matrices with random numbers between -10 and 10 generated from a uniform distribution. They represent the affinity between individuals in a social network. We generate 10 instances for each size of $n=20,25,30,35,150$ and 500.

Type II This data set consists of 60 symmetric matrices with random numbers in $[-10,-5]\cup[5,10]$. These instances reflect the polarization that occurs when people get together in groups. We can identify clusters of individuals with a high attraction within clusters and a high repulsion between clusters and with no room for indifference. We generate 10 instances for each size of $n=20,25,30,35,150$ and 500.

We also consider a real example in the context of a university committee selection from a set of $n=586$ professors. In our first experiment we consider the two mathematical models described in Section 3 and the 80 small instances in our test-bed ($n=20,25,30,35$). Specifically, we compare the solutions obtained with Cplex 12 when solving the Prokopyev et al. [10] Max-Mean DP linear mixed 0-1 formulation and the Kuo et al. [7] Max-Sum model $n-1$ times (for any possible value of $m=2,3,\ldots,n$). Table 1 shows, for each method and each size, the average objective function value (Value), the average of the number of selected elements in the solution (m), and the average CPU time.

Results in Table 1 clearly indicate that Cplex is only able to solve small problems within moderate running times. In particular, with the Max-Mean model [10] it only solves the instances with $n=20$ and 25, and it cannot solve the instances with $n\geq 30$ in 5 h of CPU time. Surprisingly, the Max-Sum model [7] applied $n-1$ times is able to solve these later instances with moderate running times (lower than 700 s) and in all the instances tested performs better than the Max-Mean model. Since we are applying different linear integer formulations for each model (as shown in Section 3), this can be probably due to the fact that the continuous relaxation of the Max-Sum model provides better bounds than the ones provided by the Max-Mean model.

For the following preliminary experimentation with heuristics we consider the 10 instances of Type I with $n=150$ and the 10 instances of Type II with $n=150$. All the methods have been implemented in MathematicaV.7 (Wolfram Research Inc.) and conducted on a Laptop Intel Core Solo 1.4 GHz at 3 Gb of RAM.

Table 1
Max-Mean solutions with Cplex 12.

		Type I instances		Type II instances	
n		**Max-Mean**	**Max-Sum ($n-1$) times**	**Max-Mean**	**Max-Sum ($n-1$) times**
20	CPU (s)	50.33	15.97	66.71	19.16
	Value	14.43	14.43	18.98	18.98
	m	7.4	7.40	7.50	7.50
25	CPU (s)	694.60	43.51	1995.1	59.58
	Value	17.32	17.32	22.07	22.06
	m	9.8	9.80	9.6	9.60
30	CPU	> 5 h	92.74	> 5 h	182.18
	Value	–	18.75		23.83
	m	–	10.70		10.80
35	CPU	> 5 h	301.23	> 5 h	620.80
	Value	–	19.39		25.70
	m	–	12.30		12.20

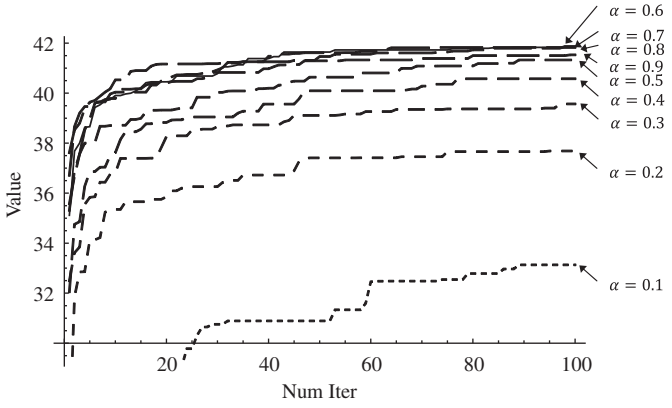

Fig. 7. Search profile of GRASP constructions.

In the second experiment we study the parameter α in our GRASP constructive method run for 100 iterations. Fig. 7 depicts the search profile of the average best values across the 10 Type I instances for nine different α values. The best solutions are consistently obtained with $\alpha=0.6$ and the worst ones with $\alpha=0.1$. Similar profiles are observed for Type II instances, being $\alpha=0.7$ the best one.

In our third experiment we compare our constructive method (C3) outlined in Fig. 4 with the parameter α set to 0.6, with the three adaptations of the constructive method by Prokopyev et al. [10], C1a, C1b and C1c, and with the three adaptations of the constructive method by Duarte and Martí [1], C2a, C2b and C2c. As described above, these methods were originally proposed for the Max-MinSum and Max-Sum models respectively, and we adapt them to the Max-Mean model (see Sections 2 and 4.4). Table 2 shows, for each method, the average objective function value (Value), the average of the number of selected elements (m), the number of best solutions found with each method (#Best), the percent deviation from the best solutions (Deviation) and the average CPU time (seconds).

Table 2 shows that our constructive procedure C3 outperforms previous constructive methods for this problem (in the different adaptations that we are proposing). Specifically, in the 20 instances tested it is able to obtain better solutions than C1a, C1b, C1c, C2a, C2b and C2c with lower CPU times. From the adaptations of previous methods, C2b emerges as the best one;

Table 2
Constructive methods.

Type		C1a	C1b	C1c	C2a	C2b	C2c	C3
I	Value	24.95	28.30	22.58	28.75	32.05	14.10	41.45
	m	54.8	56.9	14.7	68.2	62	7.1	41.8
	#Best	0	0	0	0	0	0	10
	Deviation (%)	39.8	31.7	45.5	30.7	22.6	65.9	0.0
	CPU (s)	329.9	990.5	107.6	26.9	48.2	12.6	4.7
II	Value	33.60	37.48	30.86	38.2	42.52	18.16	54.99
	m	61.6	59.8	14.6	80.7	69	7.2	40.1
	#Best	0	0	0	0	0	0	10
	Deviation (%)	38.8	31.8	43.8	30.5	22.6	66.8	0.0
	CPU (s)	230.1	595.8	71.4	25.714	48.62	12.17	5.02

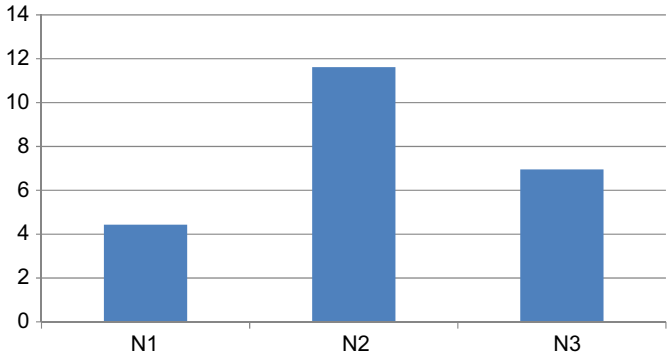

Fig. 8. Average number of improvements of each GRASP3 neighborhood.

Table 3
GRASP methods.

		GRASP1a	GRASP2a	GRASP3
Type I	Value	39.078	39.596	43.236
	m	49.4	45.7	44
	#Best	0	0	10
	Deviation (%)	9.63	8.41	0.00
	CPU time (s)	370.4	61.5	26.4
Type II	Value	49.43	52.23	56.84
	m	54.6	52.2	46.1
	#Best	0	3	10
	Deviation (%)	13.06	8.14	0.00
	CPU time (s)	275.0	80.1	22.9

Table 4
Comparison on large instances.

		GRASP1a	GRASP2a	GRASP3	GRASP3+PR
Type I	Value	66.796	70.163	77.137	77.977
	m	154.4	157.6	139.4	145.4
	#Best	0	0	0	10
	Deviation (%)	14.31	10.01	1.07	0.00
	CPU time (s)	1414.5	950.9	717.3	688.1
Type II	Value	88.98	92.68	102.96	104.37
	m	186.1	170.4	143.2	144.4
	#Best	0	0	0	10
	Deviation (%)	14.74	11.18	1.53	0.00
	CPU time (s)	804.8	708.3	662.9	679.64

although it must be noted that C2c needs shorter running times than its competing adaptations. On the other hand, from the previous methods, C1a and C2a present the best trade-off between solution quality and CPU time, we therefore will use them in the following experiments.

In our fourth experiment we compare our entire GRASP algorithm described in Section 4, constructive + local search, called GRASP3, with the previous methods GRASP1a [10] and GRASP2a [1] in which we apply the best constructive methods identified above (C1a and C2a respectively). Table 3 reports the associated results on the medium sized instances ($n=150$).

Results in Table 3 confirm the superiority of our proposal with respect to previous methods. Specifically, GRASP3 obtains an average percent deviation with respect to the best known solution of 0.00% and 0.00% on Type I and II instances respectively, while GRASP1a presents 9.63 % and 13.06 % and GRASP2a 8.41% and 8.14% respectively.

To complement this information, we apply a Friedman test for paired samples to the data used to generate Table 3. The resulting *p*-value of 0.000 obtained in this experiment clearly indicates that there are statistically significant differences among the three methods tested (we are using the typical significance level of 0.05 as the threshold between rejecting or not the null hypothesis). A typical post-test analysis consists of ranking the methods under comparison according to the average rank values computed with this test. According to this, the best method is the GRASP3 (with a rank value of 2.93), followed by the GRASP2a (1.93) and finally the GRASP1a (with 1.15 rank value).

Our local search in GRASP3 is formed with three different neighborhoods in a VND method: N_1 (remove an element from the solution), N_2 (exchange a selected element with an unselected one) and N_3 (add an unselected element to the solution). Thus, an interesting study is to measure the contribution of each neighborhood to the quality of the final solution. Fig. 8 depicts a bar chart with the average number of times, in the 20 instances used in our preliminary experimentation, that each neighborhood is able to improve the current solution. We can see that, although N_2 improves the solutions in a larger number of cases, N_1 and N_3 are also able to improve them and therefore contribute to obtain the final solution.

In the final experiment we target the 20 largest instances in our data set ($n=500$). Table 4 shows the average results on each type of instances of GRASP1a [10], GRASP2a [1] and our two methods, GRASP3 and GRASP3 with Path Relinking described in Section 4. We set the number of global iterations in order that all the methods run for similar CPU times. The size of the elite set in the Path Relinking, *b*, is set to 20 and the distance threshold *dth* is set to 25*b*.

Results in Table 4 are in line with the results obtained in the previous experiments. They confirm that GRASP3 consistently obtains better results than GRASP1a and GRASP2a. As shown in the last column of Table 4, Path Relinking is able to improve the results of GRASP3 in all the instances. The Friedman test applied to the results summarized in Table 4 exhibits a *p*-value of 0.000 indicating that there are statistically significant differences among the four methods tested. According to the ranking of this test, the best method is the GRASP3+PR (with a rank value of 3.70), followed by the GRASP3 (3.30), GRASP2 (2.00) and finally the GRASP1 (with 1.00 rank value).

Considering that GRASP3 and GRASP3+PR obtain similar rank values, we compared both with two well-known nonparametric tests for pairwise comparisons: the Wilcoxon test and the Sign test. The former one answers the question: Do the two samples (solutions obtained with GRASP3 and GRASP3+PR in our case) represent two different populations? The resulting *p*-value of 0.028 indicates that the values compared come from different methods. On the other hand, the Sign test computes the number of instances on which an algorithm supersedes another one. The resulting p-value of 0.021 indicates that GRASP3+PR is the clear winner.

Figs. 9 and 10 show the search profile of the four methods on Type I and Type II instances respectively for an 800 s run.

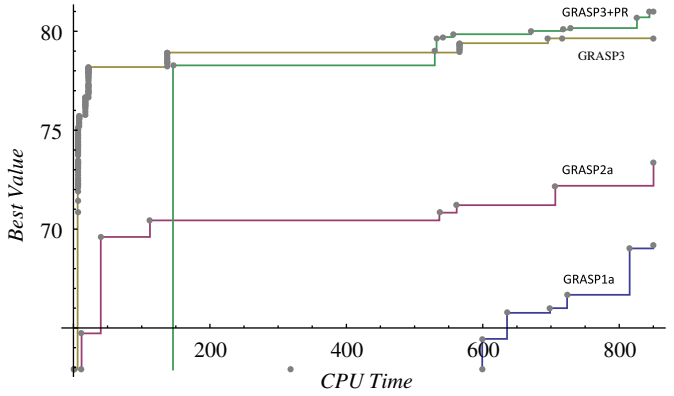

Fig. 9. Search profile on large Type I instances.

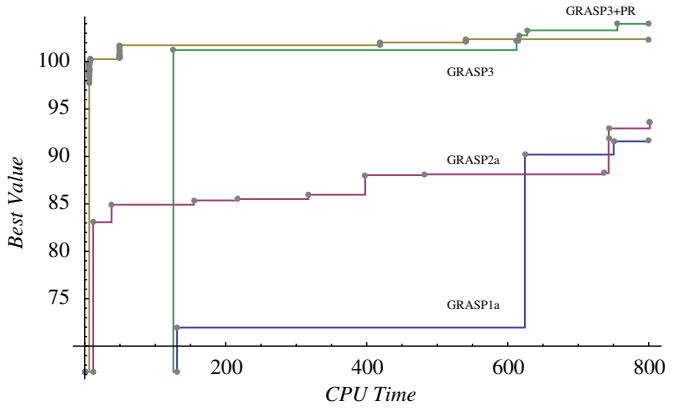

Fig. 10. Search profile on large Type II instances.

Figs. 9 and 10 clearly show that from the very beginning GRASP3 obtains better results than GRASP1a and GRASP2a. On the other hand, in the GRASP3+PR execution, Path Relinking is applied after 100 GRASP3 iterations, which occurs approximately after 700 s. Then it starts to apply PR to pairs of solutions in the elite set, obtaining new better solutions. This diagram and the results in Table 4 show that Path Relinking constitutes a good post-processing for GRASP solutions.

Finally, we compare the four methods solving a real instance. In particular we apply them to obtain a diverse assembly of professors from a set of $n=586$ in the ESPOL University at Guayaquil (Ecuador). For each professor, we record 7 attributes (tenure position, gender, academic degree, research level, background, salary level, and department), and the similarity measure between each pair of them is computed with the modified difference measure md_{xy} described in the introduction. The solution obtained with our GRASP3+PR method in 127.1 seconds has 90 professors and a similarity value of 1.11. This result compares favorably with the similarity values of 0.65, 0.94 and 1.09 obtained with GRASP1a, GRASP2a and GRASP3 in 307.9, 209.5 and 116.6 seconds respectively.

To set a benchmark for future comparisons, we include Table 5 with the optimum value of the small instances ($n=35$) and Table 6 with our best value on the large instances ($n=500$).

6. Conclusions

The Max-Mean DP maximization is a computationally difficult optimization problem that arises in the context of equitable dispersion problems. It has served us well as test case for a few new search strategies that we are proposing. In particular, we tested a GRASP constructive algorithm based on a non-standard combination of greediness and randomization, a local search strategy based on the variable neighborhood descent methodology, which includes three different neighborhoods, and a path relinking post-processing. This later method is based on a measure to control the diversity in the search process.

We performed extensive computational experiments to first study the effect of changes in critical search elements and then to compare the efficiency of our proposal with previous solution procedures. The comparison with two previous methods also based on GRASP favors our proposal.

Table 5
Optimum values for test instances with $n=35$ nodes.

Type I			Type II		
Instance	**Optimum**	**CPU (s)**	**Instance**	**Optimum**	**CPU (s)**
MDPI1_35	19.1833	414.8	MDPII1_35	25.968	537.0
MDPI2_35	17.168	287.4	MDPII2_35	26.135	460.6
MDPI3_35	17.0746	215.7	MDPII3_35	24.159	800.3
MDPI4_35	23.35	266.9	MDPII4_35	24.415	754.3
MDPI5_35	19.0177	246.0	MDPII5_35	23.857	770.3
MDPI6_35	19.445	564.7	MDPII6_35	24.673	801.1
MDPI7_35	19.4971	199.5	MDPII7_35	29.394	279.3
MDPI8_35	21.2307	233.7	MDPII8_35	25.297	819.9
MDPI9_35	20.98	274.0	MDPII9_35	27.435	413.2
MDPI10_35	16.9378	309.1	MDPII10_35	25.712	571.5

Table 6
Best values for test instances with $n=500$ nodes.

Type I			Type II		
Instance	**Best value**	**CPU (s)**	**Instance**	**Best value**	**CPU (s)**
MDPI1_500	78.605	715.8	MDPII1_500	108.152545	765.8
MDPI2_500	76.8734667	682.0	MDPII2_500	103.287851	655.8
MDPI3_500	75.6914063	667.6	MDPII3_500	106.301714	709.7
MDPI4_500	81.8058434	647.1	MDPII4_500	104.618442	725.4
MDPI5_500	78.5695714	683.1	MDPII5_500	103.608188	706.9
MDPI6_500	79.6426282	732.0	MDPII6_500	104.813987	713.1
MDPI7_500	75.4989726	607.2	MDPII7_500	104.503378	625.6
MDPI8_500	76.9836424	666.4	MDPII8_500	100.021407	609.3
MDPI9_500	75.7209449	634.7	MDPII9_500	104.927769	635.7
MDPI10_500	80.3789051	849.2	MDPII10_500	103.497014	648.6

Acknowledgments

This research has been partially supported by the Ministerio de Ciencia e Innovación of Spain (Grant Ref. TIN2009-07516).

References

[1] Duarte A, Martí R. Tabu search and grasp for the maximum diversity problem. European Journal of Operational Research 2007;178:71–84.
[2] Ghosh J. Computational aspects of the maximum diversity problem. Operations Research Letters 1996;19:175–181.
[3] Glover F, Kuo CC, Dhir KS. A discrete optimization model for preserving biological diversity. Applied Mathematical Modeling 1995;19:696–701.
[4] Glover F, Kuo CC, Dhir KS. Heuristic algorithms for the maximum diversity problem. Journal of Information and Optimization Sciences 1998;19(1):109–132.
[5] Glover F, Laguna M. Tabu search. Boston: Kluwer Academic Publishers; 1997.
[6] Hansen, P and Mladenovic N. Variable neighborhood search. In: Glover F, Kochenberger G, editors. Handbook of metaheuristics. 2003. p. 145–84.

[7] Kuo M, Glover F, Dhir. K. Analyzing and modeling the maximum diversity problem by zero-one programming. Decision Sciences 1993;24:1171–1185.
[8] Lee, TQ, Park, Y and Park, YT. A similarity measure for collaborative filtering with implicit feedback. In: Huang D-S, Heutte L, Loog M, editors. ICIC 2007, LNAI 4682. 2007. p. 385–97.
[9] Page SE. The difference—How the power of diversity creates better groups, firms, schools, and societies. Princeton University Press; 2007.
[10] Prokopyev OA, Kong N, Martinez-Torres DL. The equitable dispersion problem. European Journal of Operational Research 2009;197:59–67.
[11] Resende MGC, Martí R, Gallego M, Duarte A. GRASP with path relinking for the max-min diversity problem. Computers and Operations Research 2010;37:498–508.
[12] Resende MGC, Ribeiro CC. Greedy randomized adaptive search procedures. In: Metaheuristics F, Glover, Kochenberger G, editors. State of-the-art Handbook. Boston: Kluwer Academic Publishers; 2001. p. 219–250.
[13] Resende M, Werneck R. A hybrid heuristic for the *p*-median problem. Journal of Heuristics 2004;10(1):59–88.