GRASP and Path Relinking for the Equitable Dispersion Problem

RAFAEL MARTÍ
Departamento de Estadística e Investigación Operativa
Facultad de Ciencias Matemáticas
Universidad de Valencia, Spain
Rafael.Marti@uv.es

FERNANDO SANDOYA
Doctorado en Investigación de Operaciones
Universidad Nacional Autónoma de México, México D.F.
Instituto de Ciencias Matemáticas
Escuela Superior Politécnica del Litoral, Guayaquil-Ecuador
fsandoya@espol.edu.ec

February 25, 2011

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.

Key Words: Metaheuristics, Diversity, GRASP with Path Relinking

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 (Glover et al. 1995). 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, Kuo and Dhir (1998), 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* (MPD) also known as the *Maxi-Sum Diversity Model* (Ghosh 1996) in which the sum of the distances between the selected elements is maximized. Heuristics (Glover et al. 1998) and meta-heuristics (Duarte and Martí 2007) 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 (Resende et al. 2010).

Prokopyev et al. (2009) 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. (1998), 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 proximity-diversity.

In this paper we consider the optimization of the Max-Mean model 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 (Prokopyev et al. 2009).

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 Dispersion Problem can be trivially formulated with binary variables as:

$$\text{Max} \quad \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}\, x_i x_j}{\sum_{i=1}^{n} x_i}$$

$$s.t. \quad \sum_{i=1}^{n} x_i \geq 2$$

$$x_i \in \{0,1\} \quad i = 1, 2, \dots, n$$

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 model, unless we repeatedly apply it for any possible value of $m = |M|$.

2. Previous Heuristics

Prokopyev et al. (2009) 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. (2009) 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: $L = \{1, 2, \dots, n\} \setminus M_k$. 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 bests 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 $i^\star$ in RCL and adds it to the partial solution: $M_{k+1} = M_k \cup \{i^\star\}$. Each construction phase terminates when the preestablished number of selected elements m is reached ($|M_k| = m$). Figure 1 outlines the pseudo-code of this method.

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

Figure 1. GRASP construction phase

We can easily adapt the method above, originally proposed for the Max-MinSum, to the Max Mean 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 $M_{k+1} = M_k \cup \{i^\star\}$ can be incrementally computed as:

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

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 Figure 1 considers the Max MinSum model for which the value of m is fixed. To adapt it to the Max Mean 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 Figure 1 with the following instruction:

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

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 $N \setminus M$). 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. We will call this entire method GRASP1 and will compare our proposal with it in the computational experiments shown in Section 5.

Duarte and Martí (2007) proposed different heuristics for the Max-Sum model. 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 $i^\star$ and a partial solution M_k, $ds(i^\star, M_k)$, to incrementally compute the dispersion value of $M_{k+1} = M_k \cup \{i^\star\}$, as:

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

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

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

where $ds_{min}(M_k) = min_{i\in L}\, ds(i, M_k)$, $ds_{max}(M_k) = max_{i\in L}\, ds(i, M_k)$ and the value of α is set to 0.5. As in the algorithm above for the Max-MinSum, we can easily adapt this method to solve the Max-Mean model by selecting m at random in each construction and dividing the value of the constructed solution by m. Figure 2 shows the associated pseudo code.

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

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

Figure 2. GRASP2 construction phase

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

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

and consider the element $i^\star$ with minimum $ds(i, M)$. Then, the method scans the unselected elements in search for the first exchange of $i^\star$ to improve the value of M. Note that when an improving move is identified, it is performed without examining the remaining elements in $N \setminus M$. This improvement phase is performed as long as the solution is improved.

3. Mathematical Models

Prokopyev et al. (2009) proposed the following linearization of the mathematical formulation for the Max-Mean model shown in the introduction:

$$\text{(MaxMean) Max} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij}\, z_{ij}$$

$$s.t. \quad y - z_i \leq 1 - x_i, \qquad z_i \leq y, \qquad z_i \leq x_i, \qquad z_i \geq 0, \qquad i = 1, \dots, n$$

$$y - z_{ij} \leq 2 - x_i - x_j, \qquad z_{ij} \leq y, \qquad z_{ij} \leq x_i,$$

$$z_{ij} \leq x_j, \qquad z_{ij} \geq 0, 1 \leq i < j \leq n;$$

$$\sum_{i=1}^{n} x_i \geq 2; \quad \sum_{i=1}^{n} z_i = 1, \quad x_i \in \{0,1\}$$

On the other hand, Kuo et al. (1993) proposed the following mathematical programming formulation of the Max-Sum model:

$$\text{(MaxSum) Max} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij}\, z_{ij}$$

$$\text{s.t.:} \ \sum_{i=1}^{n} x_i = m$$

$$z_{ij} \geq x_i + x_j - 1, \qquad z_{ij} \leq x_i, \quad z_{ij} \leq x_j, \quad 1 \leq i \leq j \leq n;$$

$$x_{ij}, z_{ij} \in \{0,1\}, \quad 1 \leq i \leq j \leq n$$

It is clear that to obtain the optimum solution of the Max-Mean problem, we can solve the Max-Sum model for any possible value of m $(i.e., 2, 3, \dots, n)$ 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 model. Figure 3 shows the Max-Mean value (y-axis) of each Max-Sum problem solved ($m = 2, \dots, 50$ on the x-axis) on an instance with $n = 50$ in which the distances between the elements were randomly generated in $[-10,10]$.

Figure 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 model is reached in this instance when $m = 14$. We have observed the same pattern (a concave function) 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.

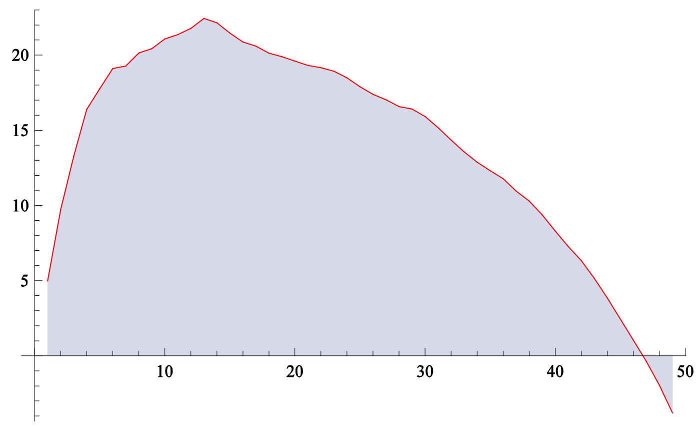

Figure 3. Max-Mean value for Max-Sum solutions

On the other hand, when distances are positives 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 model 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 $d_{ij} \leq d_{ik} + d_{jk}$. Adding it for all pairs in M we obtain:

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

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

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

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

In this section we describe our proposal to obtain high quality solutions to the Max-Mean problem. It consists of a GRASP (construction plus local search) with a Path Relinking post-processing.

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 Figure 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 (Resende and Ribeiro 2001) 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 (Resende and Werneck 2004) 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 problem, 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 α $(0 < \alpha \leq 1)$ 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 ($eval(i^\star) > 0$) and adds it to the partial solution, $M_{k+1} = M_k \cup \{i^\star\}$; otherwise the method stops. Figure 4 shows a pseudo-code of the method that we will call GRASP3.

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

While ( $improve = 1$ )
        3.  Compute $CL = \{1,2,\dots,n\} \setminus M_k$
        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 $i^\star$ in $RCL$ with maximum $eval$ value

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

        Else
                9.  $improve = 0$
```

Figure 4. GRASP3 construction phase

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 elements, keeping constant the number m of selected elements. Since we do not have this size constraint in the Max-Mean model and we admit solutions with any value of m, we can consider an extended neighborhood. Specifically, based on the Variable Neighborhood Descent (VND) methodology (Hansen and Mladenovic 2003), 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 search 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:

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

Then, when we explore N_1 to remove an element from M_m, we scan the selected elements in the order given by ds, 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.

4.3 Path Relinking

The PR algorithm (Glover and Laguna, 1997) 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. Figure 5 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. 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( $iter \le GlobalIter$ )
     4. Apply the construction phase of GRASP $\Rightarrow x$.
     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 $d(x',ES) \ge dth$ ) )
          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 $x^{best}$.
```
Figure 5. GRASP with Path Relinking

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 (Figure 5). 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.

In our first experiment we consider the two mathematical models described in Section 3 and the 60 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. (2009) Max-Mean model and the Kuo et al. (1993) Max-Sum model $n-1$ times (for any possible value of $m = 2,3, \dots, n$). 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.

		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	14.66	4.5254	31.95
	Value	14.43	14.43	48.56	48.56
	m	7.4	7.40	14	14.00
25	CPU (s)	694.60	41.83	22.0547	206.55
	Value	17.32	17.32	63.21	63.21
	m	9.8	9.80	18	18.00
30	CPU	> 5 hours	102.30	240.5314	885.19
	Value	-	18.75	75.26	75.26
	m	-	10.70	21	21.00
35	CPU	> 5 hours	719.51	719.5	8392.26
	Value	-	16.94	88.95	88.95
	m	-	9	25	25

Table 1. Max-Mean solutions with Cplex 12

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 (Prokopyev et al. 2009) it only solves the Type I instances with $n = 20$ and 25, and it cannot solve the Type I instances with $n = 30$ and 35 in 5 hours of CPU time. Surprisingly, the Max-Sum model (Kuo et al. 1993) applied $n-1$ times is able to solve these later instances with a total average time of 102.3 seconds for $n = 30$ and 719.51 for $n = 35$ and in all the Type I instances tested performs better than the Max-Mean model. On the other hand, the comparison between both models on Type II instances draws a different picture. Specifically, both models are able to solve all the small instances (up to $n = 35$) and, as expected, the Max-Mean model seems better suited to solve these Max-Mean instances than the Max-Sum model applied $n-1$ times (since it consistently exhibits shorter running times).

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

In the second experiment we study the parameter α in our GRASP constructive method run for 100 iterations. Figure 6 depicts the search profile of the average best values across the 10 type I instances for five different α values. The best solutions are consistently obtained with $\alpha = 0.6$ and the worst with $\alpha = 0.3$. On the other hand we have not observed significant differences among the solutions obtained with different values α of when solving the Type II instances.

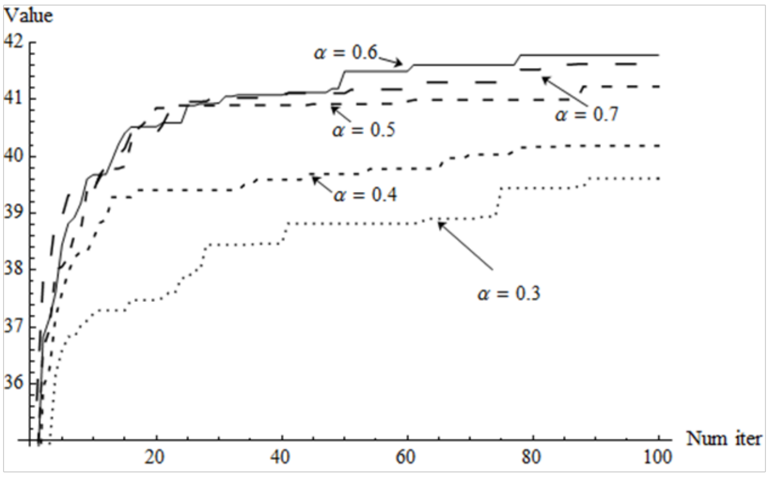

Figure 6. Search profile of GRASP constructions.

In our third experiment we compare our constructive method (C3) outlined in Figure 4 with the parameter α set to 0.6, with two previous constructive methods, C1 (Prokopyev et al. 2009) and C2 (Duarte and Martí 2007). As described in Section 2 these two methods were originally proposed for the Max-MinSum and Max-Sum models respectively and we adapt them to the Max-Mean model. 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 (of this experiment) found with each method (#Best), the percent deviation from the best solutions in the experiment (Deviation) and the average CPU time (seconds on a Pentium 4).

		C1	C2	C3
	Value	26.068	28.733	41.797
	m	57.9	81.8	45.3
Type I	# Best	0	0	10
	Deviation	37.68 %	31.24 %	0.00 %
	CPU time (s)	178.9	52.5	9.7
	Value	257.599	384.219	393.556
	m	97.4	104.1	106
Type II	# Best	0	0	10
	Deviation	34.54 %	2.37 %	0.00%
	CPU time (s)	154.8	54.4	33.0

Table 2. Constructive methods

Table 2 shows that our constructive procedure C3 outperforms previous constructive methods for this problem. In the 20 instances tested it is able to obtain better solutions than C1 and C2. It is interesting to observe that the best solutions obtained with C3 present a lower number of selected elements (m) than the best solutions of C1 and C2 in Type I instances. However, Type II instances exhibit a different pattern since the average of the m value is larger in C3 than in C1 and C2. Finally it must be noted that C3 needs shorter running times than its competing methods to reach better solutions than them.

In our fourth experiment we compare our entire GRASP algorithm described in Section 4, constructive + local search, called GRASP3, with the previous methods GRASP1 (Prokopyev et al. 2009) and GRASP2 (Duarte and Martí 2007) described in Section 2. Table 3 reports the associated results on the medium sized instances ($n = 150$).

		GRASP1	GRASP2	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)	241.6	61.5	26.4
Type II	Value	360.353	389.569	393.556
	m	100.9	105.2	106
	# Best	0	3	10
	Deviation	8.44 %	1.01 %	0.00 %
	CPU time (s)	214.0	67.9	45.5

Table 3. GRASP methods

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 GRASP1 presents 9.63 % and 8.44 % and GRASP2 8.41% and 1.01% 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 GRASP2 (1.93) and finally the GRASP1 (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. Figure 7 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.

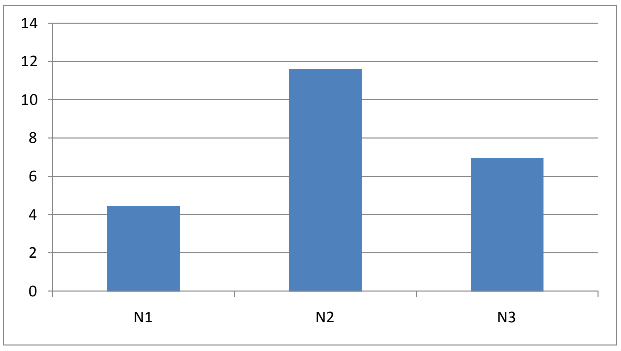

Figure 7. Average number of improvements of each GRASP3 neighborhood.

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 GRASP1 (Prokopyev et al. 2009), GRASP2 (Duarte and Martí 2007) 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.

		GRASP1	GRASP2	GRASP3	GRASP3+PR
	Value	66.796	70.163	77.232	77.740
	m	154.4	157.6	136.6	142
Type I	# Best	0	0	1	9
	Deviation	14.11%	9.79%	0.71%	0.06%
	CPU time (s)	1522.5	1092.9	1094.9	1006.1
	Value	1266.177	1311.651	1319.844	1319.844
	m	357.1	350.4	353	353
Type II	# Best	0	0	9	10
	Deviation	4.07%	0.62%	0.00%	0.00%
	CPU time (s)	1824.5	1469.3	1114.9	1250.4

Table 4. Comparison on large instances

Results in Table 4 are in line with the results obtained in the previous experiments. They confirm that GRASP3 consistently obtains better results than GRASP1 and GRASP2. As shown in the last column of Table 4, Path Relinking is able to improve the results of GRASP3 in 9 out of 10 Type I instances but only in 1 out of 10 Type II instances. According to our experimentation Type II instances seem easier for the heuristics and apparently, the results obtained with GRASP3 on them cannot be improved with any the methods considered (even if they run for long time). 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.

Finally we depict in Figure 8 the search profile of the four methods on Type I instances for a 1200 seconds run. This figure clearly shows that from the very beginning GRASP3 obtains better results than GRASP1 and GRASP2. On the other hand, in the GRASP3+PR execution, Path Relinking is applied after 100 GRASP3 iterations, which occurs approximately after 700 seconds. Then it starts to apply PR to pairs of solutions in the elite set, obtaining new better solutions after 300 seconds (1000 seconds of total running time). This diagram and the results in Table 4 show that Path Relinking constitutes a good post-processing for GRASP solutions.

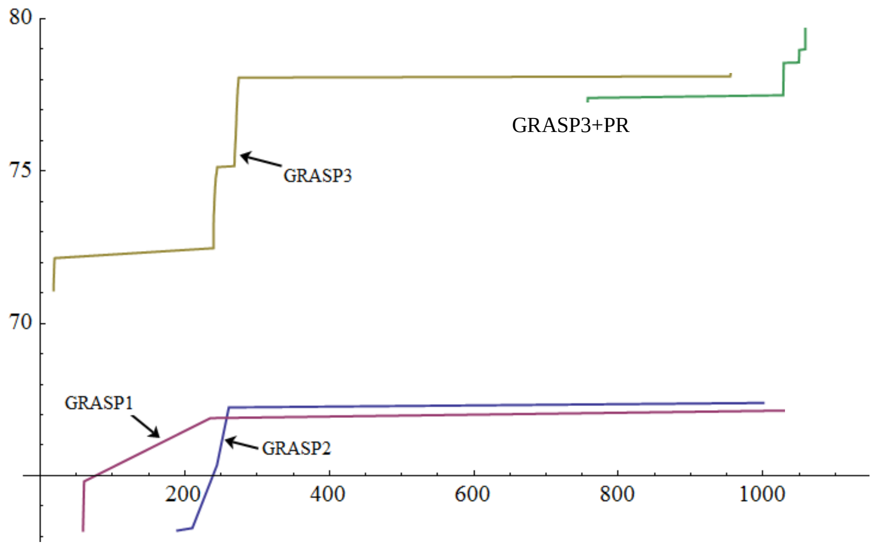

Figure 8. Search profile on large type I instances

6. Conclusions

The Max-Mean 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.

Acknowledgments

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

References

Duarte A, Martí R. "Tabu Search and GRASP for the Maximum Diversity Problem", European Journal of Operational Research; 178, 71-84, (2007)

Ghosh, J., "Computational aspects of the maximum diversity problem", Operations Research Letters 19, pp. 175-181, (1996)

Glover, F., Kuo, C.C. and Dhir K.S., "A discrete optimization model for preserving biological diversity", Appl. Math. Modeling 19, pp. 696-701, (1995)

Glover, F., Kuo, C.C. and Dhir K.S., "Heuristic Algorithms for the Maximum Diversity Problem", Journal of Information and Optimization Sciences, vol. 19 (1), 109-132, (1998)

Glover, F., M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, (1997).

Hansen, P. and N. Mladenovic, "Variable neighborhood search", in Handbook of Metaheuristics, Glover, F. and Kochenberger, G. (Eds.), pp. 145-184, (2003)

Kuo, M., F. Glover, K. Dhir. "Analyzing and modeling the maximum diversity problem by zero-one programming", Decision Sciences 24, pp. 1171-1185, (1993)

Prokopyev O.A., Kong N, Martinez-Torres DL. "The equitable dispersion problem", European Journal of Operational Research, 197: 59-67, (2009)

Resende MGC, Martí R, Gallego M, Duarte A. "GRASP with path relinking for the max-min diversity problem", Computers and Operations Research, 37: 498-508, (2010)

Resende, M.G.C. and Ribeiro, C.C. "Greedy Randomized Adaptive Search Procedures", State of-the-art Handbook in Metaheuristics, F. Glover and G. Kochenberger (Eds.), Kluwer Academic Publishers, Boston, pp. 219-250, (2001)

Resende,M., Werneck, R., "A hybrid heuristic for the p-median problem", Journal of heuristics, Vol 10, issue 1, pp 59-88, (2004)