Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 178 (2007) 71–84
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

Tabu search and GRASP for the maximum diversity problem

Abraham Duarte [a], Rafael Martí [b,*]

[a] Departamento de Informática, Estadística y Telemática, Universidad Rey Juan Carlos, Spain
[b] Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Dr. Moliner, 50, Burjassot, 46100 Valencia, Spain

Received 22 August 2005; accepted 25 January 2006
Available online 20 March 2006

Abstract

In this paper, we develop new heuristic procedures for the maximum diversity problem (MDP). This NP-hard problem has a significant number of practical applications such as environmental balance, telecommunication services or genetic engineering. The proposed algorithm is based on the tabu search methodology and incorporates memory structures for both construction and improvement. Although proposed in seminal tabu search papers, memory-based constructions have often been implemented in naïve ways that disregard important elements of the fundamental tabu search proposals. We will compare our tabu search construction with a memory-less design and with previous algorithms recently developed for this problem. The constructive method can be coupled with a local search procedure or a short-term tabu search for improved outcomes. Extensive computational experiments with medium and large instances show that the proposed procedure outperforms the best heuristics reported in the literature within short computational times.

Keywords: Global optimization; Metaheuristics; Tabu search

1. Introduction

The problem of selecting a subset of maximum diversity from a given set of elements is known as the maximum diversity problem (MDP). This problem arises in a wide range of real-world settings and it has been the subject of several previous studies beginning with the work by Kuo et al. (1993). We can find a large number of MDP applications in different contexts, such as ecological, medical or social sciences. In the former, it is well known that ecological systems depend on diversity for survivability and establishing the most viable system relies crucially on considerations of diversity maximization, as documented in Pearce (1987). In medical treatments, combating diseases, both by preventive planning and after their onset, is enhanced by programs with mote diverse lines of defense to combat the broadest spectrum of potential disease causing agents (Kuo et al., 1993). Moreover, in animal and plant genetics, the goal of obtaining new varieties by controlled breeding stocks with desired qualities of diversity can be formulated as a MDP (Porter et al., 1975). The maximum diversity problem also appears in the context of ethnicity when it is study from a historical perspective, as shown

* Corresponding author. Tel.: +34 96 386 4362.
E-mail addresses: Abraham.Duarte@urjc.es (A. Duarte), Rafael.Marti@uv.es (R. Martí).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.01.021

in Swierenga (1977), and more recently in the application of the US immigration policy that establishes the promotion of ethnic diversity among the immigrants (McConnell, 1988). Other applications include environmental balance, product design, workforce management, telecommunication services or genetic engineering (see Glover et al., 1995, 1998 for a detailed description).

Evolutionary methods based their success on maintaining a set of solutions, often called *population*, which achieved a good balance between quality and diversity. For instance, Scatter Search (Laguna and Martí, 2003) operates on a small set of solutions called the *reference set*, which is created by selecting a few distinct, good and maximally diverse solutions from a larger set of initial solutions. Therefore, in these methods, a maximum diversity problem often needs to be solved within low running times, since it is part of the optimization process applied to solve the problem at hand.

The MDP can be formally stated as follows. Let $S = \{s_i : i \in N\}$ be a set of elements where $N = \{1, 2, \ldots, n\}$ is the set of indexes and each element s_i can be represented as a vector $s_i = (s_{i1}, s_{i2}, \ldots, s_{ir})$. Let d_{ij} be the distance between elements s_i and s_j, and let $m < n$ be the desired size of the diverse set. The problem then consists of selecting m elements in S in order to maximize the sum of the distances between the selected elements:

$$\text{Maximize} \quad z = \sum_{i<j} d_{ij} x_i x_j$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = m,$$

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

where $x_i = 1$ indicates that element s_i has been selected. This formulation appears in Kuo et al. (1993), but it was not solved directly. The formulation is transformed to equivalent linear integer programs that offer greater computational efficiency. One of these formulations was used as the basis for showing that the maximum diversity problem is NP-hard.

Ghosh (1996) also proved the completeness of the problem. The author proposed a multi-start algorithm for the MDP and compared the heuristic solutions on small instances ($n \leqslant 40$) with the optima obtained from the formulation mentioned above.

In Glover et al. (1998), four different heuristics are proposed for this problem. Since different versions of this problem include additional constraints, the objective is to design heuristics whose basic moves for transitioning from one solution to another are both simple and flexible, allowing these moves to be adapted to multiple settings. Moves that are especially attractive in this context are *constructive* and *destructive* moves that drive the search to approach and cross-feasibility boundaries from different directions. Such moves are also highly natural in the maximum diversity problem, where the goal is to determine an optimal composition for a set of selected elements. The authors compare the solution obtained with their heuristics with the optimal solution in small instances ($n \leqslant 30$).

Silva et al. (2004) present several GRASP algorithms (Resende and Ribeiro, 2001) for the MDP. Specifically, the authors propose three constructive algorithms within the GRASP framework and two local search methods for this problem. Extensive computational experiments with medium instances ($n \leqslant 500$) show that their procedures outperform previous methods when they run for extremely long times (their methods employ on average more than 20 hours of CPU time on the instances with $n = 500$).

Tabu search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. One of the main components of tabu search sis its use of adaptive memory, which creates more flexible search behavior. Memory-based strategies are therefore the hallmark of tabu search approaches, founded on a quest for "integrating principles," by which alternative forms of memory are appropriately combined with effective strategies for exploiting them.

The structure of a neighborhood in tabu search goes beyond that used in local search by embracing the types of moves used in constructive and destructive processes (where the foundations for such moves are accordingly called *constructive neighborhoods* and *destructive neighborhoods*). We can implement memory structures within a constructive process to favor (or avoid) the inclusion of certain elements in the solution previously identified as attractive (or unattractive). Such expanded uses of the neighborhood concept reinforce a fundamental perspective of TS, which is to define neighborhoods in dynamic ways that can include serial or simultaneous consideration of multiple types of moves. Constructive neighborhoods have been proposed from the very beginning of the methodology, as documented in Glover and Laguna (1997); how-

ever, they have rarely been applied in TS implementations.

In this paper we introduce different solution methods for the MDP. Specifically, we propose two constructive methods, the first one based on a GRASP construction and the second one based on memory structures. Moreover, we propose an improved local search algorithm (as compared with those previously reported) and a short-term memory tabu search method. Our experimentation with medium ($n = 500$) and large instances ($n = 5000$) shows that the application of tabu search methodology to this problem outperforms the previous approaches identified as the best.

2. Previous approaches

Ghosh (1996) proposes a multi-start algorithm for the MDP. It consists of a construction phase and a local search post-processing. The construction performs m iterations to obtain a solution. In each iteration one element is selected according to an estimation of its contribution to the final diversity. The local search implements a straightforward "hill climbing" heuristic based on performing the best available exchange. Exchanges in this context consist of replacing one selected element with an unselected one. In mathematical terms, given the set *Sel* of selected elements, if we replace $s_v \in Sel$ with $s_u \in S\text{-}Sel$, the objective value of the new solution can be trivially computed from the value of the original one with the expression $z' = z + exchange(u, v)$ where

$$exchange(u, v) = \sum_{s_i \in Sel - \{s_v\}} d_{iu} - d_{iv}.$$

The procedure scans the set of selected elements *Sel* ($|Sel| = m$) in search of the exchange whose application results in the largest increase of the objective function. The method performs the best available exchange in each iteration until no further improvement is possible. As mentioned by other authors (see for instance Silva et al., 2004) this multi-start method produces results of relatively low quality. However, as will be shown, the local search phase has been used in several algorithms.

In Glover et al. (1998) two constructive and two destructive heuristics are proposed. The first constructive and destructive methods are based on the concept of the center of gravity of a set. The center, *s_center*(X), of a set of elements $X = \{s_i : i \in I\}$ is defined as

$$s\_center(X) = \frac{\sum_{i \in I} s_i}{|X|}.$$

Figs. 1 and 2 present, respectively, an outline of the first constructive and destructive heuristics, where S is the set of elements and *Sel* is the set of selected elements.

Algorithm C_1 basically selects, at each step, the element s_{i^*} with the maximum distance to the center from among the elements already selected. The method finishes when m elements have been selected. On the contrary, starting with all the elements selected, D_1 unselects, at each step, the element s_{i^*} with the minimum distance to the center from among the selected elements. The method finishes when n–m elements have been unselected.

The second constructive and destructive heuristics, C_2 and D_2, are variations on the first ones, where instead of constructing a center of the set, the distance between an element s_i and a set $X = \{s_j : j \in I\}$ is defined as follows:

$$d(s_i, X) = \sum_{j \in I} d(s_i, s_j).$$

C_2 selects, at each step, the element with the maximum distance to the already selected elements *Sel*, where the distance to set *Sel* is computed with the expression above. Symmetrically, D_2 unselects, at each step, the element with the minimum distance to the set *Sel* of selected elements.

1. $Sel = \emptyset$
2. Compute $s_c = s\_center(S)$

while ($|Sel| < m$)

3. Let $i^* / d(s_{i^*}, s_c) = \max_{s_i \in S} \{d(s_i, s_c)\}$
4. $Sel = Sel \cup \{s_{i^*}\}$
5. $S = S - \{s_{i^*}\}$
6. $s_c = s\_center(Sel)$

end while

Fig. 1. Constructive heuristic C_1.

1. $Sel = S$
2. Compute $s_c = s\_center\,(S)$

while ($|Sel| > m$)

3. Let $i^* / d(s_{i^*}, s_c) = \min_{s_i \in S} \{d(s_i, s_c)\}$
4. $Sel = Sel - \{s_{i^*}\}$
5. $s_c = s\_center(Sel)$

end while

Fig. 2. Destructive heuristic D_1.

As previously mentioned, Silva et al. (2004) implement three GRASP constructions: KLD, KLDv2 and MDI, and two local search procedures: GhA and Soma, where GhA is the method proposed by Ghosh (1996). Their computational experimentation showed that when running time is limited, KLD coupled with GhA produces the best quality solutions. On the other hand, when longer running times are admissible, KLDv2 coupled with GhA is the best method. Therefore we will restrict our attention to these two combinations. Nonetheless, as shown in their experimentation and confirmed in ours, all these methods present extremely long running times when solving medium-size instances.

KLD computes for each element s_i the k elements with larger values of d_{ij} and calculates the sum sd_i of these k values. Then, at each iteration, it constructs a list of candidates RCL with the k non-selected elements with larger sd_i values. Then it randomly selects an element from RCL. A reactive mechanism sets the value of parameter k according to the results obtained with four particular values given by the expression $k = m + (1 + \alpha)(n - m)/2$ for $\alpha = -0.2$, -0.1, 0.1 and 0.2. Strictly speaking, KLD is not a GRASP construction since RCL is computed offline and no adaptive mechanism is present.

KLDv2 implements a pure GRASP construction with adaptive mechanisms. Fig. 3 shows the outline of this procedure. At iteration c ($|Sel| = c - 1$) KLDv2 computes for each element s_i the k–c non-selected elements with larger values of d_{ij} and calculates the sum sd_i of these k–c values. Then, the greedy function $gf(i)$ is computed as

$$gf(i) = sd_i + \sum_{s_j \in Sel} d_{ij}.$$

RCL is now formed with the k elements with larger $gf(i)$ values, and the next element in Sel is randomly selected from RCL. The method finishes when m elements have been selected. The parameter k is set with the same reactive mechanism described in KLD.

Finally, the most distant insertion heuristic, MDI, starts off by randomly selecting one element. Then, in the second step, it selects the element with the longest distance to the element already selected. From this point on, at each step, it constructs an RCL with the unselected elements with maximum distances. Now, the distance associated to a non-selected element j is computed as the sum of two terms, the first one corresponding to the sum of distances between all the selected elements and the second term being the sum of distances between j and all the selected elements. With these values, a reactive GRASP is implemented using the same mechanism described above.

1. $S = \{s_i : i = 1, 2, .., n\}$
2. $Sel = \emptyset$

while ($|Sel| < m$)

3. For each $s_i \in S\text{-}Sel$ compute
   3.1. D_i as the set of k-c elements $s_j \in S\text{-}Sel$ with larger values of d_{ij}.
   $$sd_i = \sum_{s_j \in D_i} d_{ij}$$
   3.2. $$gf(i) = sd_i + \sum_{s_j \in Sel} d_{ij}$$
4. Compute RCL with the k elements in $S\text{-}Sel$ with larger $gf(i)$ values
5. Select randomly s_{i^*} in RCL
6. $Sel = Sel \cup \{s_{i^*}\}$
7. $S = S - \{s_{i^*}\}$

end while

Fig. 3. Constructive heuristic KLDv2.

3. New constructive methods

In this section we propose two types of constructive algorithms for the maximum diversity problem. The first one is based on GRASP constructions while the second one implements memory structures to discourage previous selections.

The GRASP methodology was developed in the late 1980s and the acronym was coined by Thomas A. Feo (Feo and Resende, 1995). Each GRASP iteration consists of constructing a trial solution and then applying an exchange procedure to find a local optimum (i.e. the final solution for that iteration). The construction phase is iterative, greedy, randomized and adaptive. It is iterative because the initial solution is built considering one element at a time. It is greedy because the addition of each element is guided by a greedy function. It is randomized because a random selection takes place and the information provided by the greedy function is used in combination with this random element. Note that the randomness in GRASP allows multiple iterations obtaining different solutions, and differentiates this method from purely deterministic greedy heuristics that can produce only one solution. Finally, it is adaptive because the element chosen at any iteration in a construction is a function of those previously chosen. (That is, the method is adaptive in

the sense of updating relevant information from one construction step to the next.) The improvement phase typically consists of a local search procedure.

Performing multiple GRASP iterations may be interpreted as a means of strategically sampling the solution space. Based on empirical observations, it has been found that the sampling distribution generally has a mean value that is inferior to the one obtained by a deterministic construction (that one with the same GRASP elements but replacing the random selection with the best available one), but the best overall trial dominates the deterministic solution with a high probability. The intuitive justification of this phenomenon is based on the ordering statistics of sampling. It implements a way of independently sampling the solution space and each construction consists of an independent algorithm. In this sense, GRASP is a memory-less method since no information is recorded from one construction to the next.

A different framework is given by the use of memory among constructions. Instead of performing an independent sampling of the solution space, constructive methods based on memory structures perform a guided selection in this space (a seminal reference can be found in Fleurent and Glover, 1999). Specifically, in tabu search constructions, the inclusion or exclusion of certain elements or groups of elements can be identified as attractive for intensification or diversification purposes. In this section we focus our attention on constructive and destructive neighborhoods in which an element is added to or dropped from the partial solution under construction. In this context we consider the inclusion of frequency memory for both intensification and diversification. In the computational section we will compare both types of approaches, those with and without memory, to solve the maximum diversity problem.

Based on each of the four methods proposed by Glover et al. (1998), we consider a GRASP construction. For example, in GRASP_C_1 we replace the greedy selection given in algorithm C_1 (see step 3 in Fig. 1) with a random selection from a Restricted Candidate List containing the elements with the longest distance to the center. Each time an element is added to the *Sel* set under construction, the center is updated and the distances recomputed. Fig. 4 shows a pseudo-code of this method.

GRASP_C_1 constructs the restricted candidate list with all the non-selected elements with a distance to the current center within a percentage α of the maximum distance to this center. This parameter α is initially set to 0.5 and is dynamically adjusted according to the best value of the constructions. If after *niter*/5 consecutive constructions the incumbent value has not been improved, we increment α by 0.1 (up to a maximum of 0.9, where *niter* is the total number of constructions). Analogously, we implement GRASP_C_2 from constructive method C_2. In a symmetric way we implement GRASP_D_1 and GRASP_D_2, from constructive procedures D_1 and D_2, based on their destructive neighborhoods. In these cases the reactive mechanism starts with α set to 0.5 and reduces its value by 0.1 every *niter*/5 consecutive constructions in which the incumbent value has not been improved (up to a minimum of 0.1). Note that in this case the construction of the *RCL* is given by the expression:

$$RCL = \{s_i \in S\text{-}Sel / d(s_i, s_c) \leqslant d_{\min} + \alpha(d_{\max} - d_{\min})\}.$$

Based on the tabu search methodology we also adapt the four constructive methods proposed by Glover et al. (1998). We record in *freq*[*i*] the number of times that element s_i has been selected in previous constructions, and compute *max_freq* as the maximum of *freq*[*i*] for all *i*. On the other hand, we record in *quality*[*i*] the average value of the previous solutions in which element s_i has been selected. Let *max_q* be the maximum of *quality*[*i*] for all *i*. Then, we modify the evaluation of the attractiveness of each non-selected element in the current construction according to these quantities to favor the selection of elements with low frequency and high quality values.

Algorithm Tabu_C_1 performs the same steps as algorithm C_1 but, instead of using the original $d(s_i, s_c)$ values, it uses the $d'(s_i, s_c)$ values according to the following expression:

1. $Sel = \emptyset$
2. Compute $s_c = s\_center(S)$

while ($|Sel| < m$)

3. $d_{\max} = \max_{s_i \in S}\{d(s_i, s_c)\}$, $d_{\min} = \min_{s_i \in S}\{d(s_i, s_c)\}$
4. $RCL = \{s_i \in S\text{-}Sel / d(s_i, s_c) \geq d_{min} + \alpha\,(d_{max} - d_{min})\}$
5. Select s_{i*} randomly in *RCL*
4. $Sel = Sel \cup \{s_{i*}\}$
5. $S = S - \{s_{i*}\}$
6. $s_c = s\_center(Sel)$

end while

Fig. 4. Constructive heuristic GRASP_C_1.

$$d'(s_i,s_c) = d(s_i,s_c) - \beta range(s_c)\frac{freq[i]}{max\_freq} + \delta range(s_c)\frac{quality[i]}{max\_q} \quad \text{where}$$

$$range(s_c) = \max_{s_j \in S\text{-}Sel} d(s_j,s_c) - \min_{s_j \in S\text{-}Sel} d(s_j,s_c).$$

Note that with the introduction of the value $range(s_c)$, we modify the distance value by only a percentage of the range of this distance. This way, the modification is smooth enough to include both the information given in the original distance and the penalty value.

The first execution of Tabu_C_1 produces the same solution as C_1 since $freq[i] = quality[i] = 0$ for all i. Subsequent constructions perform alternative selections based on the frequency and quality values (see in Fig. 5 a pseudo-code of the method by which *niter* constructions are performed). In a symmetric way we implement Tabu_D_1 from constructive procedure D_1.

In a similar way we adapt constructive procedures C_2 and D_2 to obtain Tabu_C_2 and Tabu_D_2, respectively. In Tabu_C_2, we modify the distance to the already selected elements computed in C_2 with the frequency and quality values as follows:

$$d'(s_i,Sel) = d(s_i,Sel) - \beta range(Sel)\frac{freq[i]}{max\_freq} + \delta range(Sel)\frac{quality[i]}{max\_q} \quad \text{where}$$

$$range(Sel) = \max_{s_j \in S\text{-}Sel} d(s_j,Sel) - \min_{s_j \in S\text{-}Sel} d(s_j,Sel).$$

The same expression is implemented in Tabu_D_2 to modify the selections according to the search information. The values of $freq[i]$ and $quality[i]$ are updated in the same way given in algorithm Tabu_C_1.

4. Local search

As mentioned in Section 2, the local search method proposed by Ghosh (1996) implements a straightforward improving heuristic based on performing, at each iteration, the best available exchange (i.e. replace $s_v \in Sel$ with $s_u \in S\text{-}Sel$). Let LS be this procedure that scans the set of selected elements *Sel* in search of the best exchange. Then, to perform a move, the method examines $m(n-m)$ exchanges and selects the best according to the evaluation:

$$exchange(u,v) = \sum_{s_i \in Sel - \{s_v\}} d_{iu} - d_{iv}.$$

The method performs moves while the objective value increases until no further improvement is possible. We have empirically found that this method provides good solutions but presents extremely long running times. Silva et al. (2004), apply this method as a post-processing to their constructions, and this explains why their global algorithms present extremely long running times (more than 20 hours of CPU time on average on the instances with $n = 500$).

We have modified LS to increase its efficiency. Specifically, we define d_i associated with each s_i in

1. $freq[i] = quality[i] = 0 \ \forall s_i \in S$

For ($iter = 1$ to $niter$)

2. $Sel = \emptyset$
3. Compute $s_c = s\_center(S)$

while ($|Sel| < m$)

For ($s_i \in S\text{-}Sel$)

4. $d'(s_i,s_c) = d(s_i,s_c) - \beta d(s_i,s_c)\frac{freq[i]}{\max\_freq} + \delta d(s_i,s_c)\frac{quality[i]}{\max\_q}$
5. Let $i^* / d'(s_{i^*},s_c) = \max_{s_i \in S}\{d'(s_i,s_c)\}$
6. $Sel = Sel \cup \{s_{i^*}\}$, $freq[i^*] = freq[i^*] + 1$
7. $S = S - \{s_{i^*}\}$
8. $s_c = s\_center(Sel)$

end while

9. Compute the solution's value z
10. $quality[i] = \frac{quality[i](freq[i]-1) + z}{freq[i]} \quad \forall s_i \in Sel$

end for

Fig. 5. Constructive heuristic TABU_C_1.

Sel, as the contribution to the objective value z of the element s_i:

$$d_i = \sum_{s_j \in Sel} d_{ij} = d(s_i, Sel).$$

Instead of computing the move value $exchange(u,v)$ for all $s_v \in Sel$ and $s_u \in S\text{-}Sel$, we select the element s_{i^*} in *Sel* with the lowest contribution to the value of the current solution. In mathematical terms, we select the element s_{i^*} with the minimum d_{i^*} value. Note that the value z of the current solution can be obtained from these values with the expression:

$$z = \frac{1}{2} \sum_{s_i \in Sel} d_i.$$

Then, instead of scanning the whole set *S-Sel* searching for the best exchange associated with s_{i^*}, we restrict our method to performing the first improving move (without examining the remaining elements in *S-Sel*). It has been well documented (see for instance Laguna et al., 1999) that in some settings, this *first* strategy provides better results than the *best* strategy and obviously takes significantly lower running times. If there is no improving move associated with d_{i^*}, we resort to the next element with the lowest d_i value and so on. This improved local search method, I_LS, performs iterations until no further improvement is possible.

Finally, we implement a short-term tabu search method, LS_TS, which is also based on exchanges. An iteration in this method begins by randomly selecting an element s_i in *Sel*. The probability of selecting element s_i is inversely proportional to its d_i value. The first improving move ($s_i \in Sel, s_j \in S\text{-}Sel$) associated with s_i is performed. (Note that if there is no improving move associated with s_i, we perform the best one available, even if it is a non-improving move.) The move is executed even when the move value $exchange(i,j)$ is not positive, resulting in a deterioration of the current objective function value. The moved elements s_i, s_j become tabu-active for *TabuTenure* iterations, and therefore they cannot be unselected (respectively selected) during this time. The LS_TS method performs iterations until in *MaxIter* consecutive iterations the best solution found cannot be improved on.

In this section we have considered three ways to improve a solution: the local search LS proposed by Ghosh (1996), our improved version of this local search method I_LS, and the final proposal based on the tabu search methodology LS_TS in which the neighborhood of a solution is systematically modified according to the tabu status. It should be noted that the proposed tabu search implements a straightforward short-term memory function. This is in line with our objective of designing a fast method capable of producing high quality solutions in short computation time. However, if longer running times are admissible, this tabu search method could be easily improved with the addition of more complex memory structures and search strategies.

5. Computational experiments

For our computational testing, we implemented in C the constructive procedures C_1, C_2, D_1 and D_2 of Glover et al. (1998), the KLD, KLDv2 and MDI of Silva et al. (2004), our four variants of the GRASP construction: GRASP_C_1, GRASP_C_2, GRASP_D_1, GRASP_D_2, and our four variants of the Tabu method: Tabu_C_1, Tabu_C_2, Tabu_D_1, Tabu_D_2. Regarding the improvement methods, we implemented the local search LS by Ghosh (1996), our improved local search method, I_LS and our short-term memory tabu search method LS_TS. In this section we analyze the performance of these 18 algorithms when solving the maximum diversity problem. The codes were compiled with Borland Builder 5.0, optimizing for maximum speed. The experiments were run on a Pentium IV at 3 GHz with 1 GB RAM.

We tested the procedures on four sets with a total of 120 instances:

(1) *Silva instances*: 20 $n \times n$ matrices with random integers generated from a $[0,9]$ uniform distribution with $n \in [100,500]$ elements and $m \in [0.1n, 0.4n]$. These 20 instances are the largest instances introduced in Silva et al. (2004).
(2) *Random Type I instances*: matrices with real numbers generated from a $(0,10)$ uniform distribution. We generate 20 instances with $n = 500$ and $m = 50$, another 20 instances with $n = 500$ and $m = 200$, and another 20 instances with $n = 2000$ and $m = 200$.
(3) *Random Type II instances*: matrices with real numbers generated from a $(0,1000)$ uniform distribution. We generate 20 instances with $n = 500$.
(4) *Glover instances*: 20 $n \times n$ matrices in which the values are the distances between each pair of points with coordinates randomly generated

in $[0, 10]$. These n points have r coordinates with r ranging from 2 to 21. Glover et al. (1998) introduced these instances with small sizes ($n \leqslant 30$). We have generated 20 instances with $n = 500$.

We start our experimentation by considering the constructive methods. These previously published methods are implemented with the search parameters recommended by their authors. Our GRASP variants are reactive and the value of the α parameter is dynamically adjusted. The first experiment has the goal of finding appropriate values for the two critical search parameters β and δ in the tabu constructions. Specifically we consider $\beta = 0.1$, 0.05, 0.01, 0.005 and $\delta = 0.001$, 0.0005 and 0.0001 and run the method for 10 seconds. In this preliminary experiment we use eight instances from the *Silva* set, eight random *Type I* instances with $n = 500$, eight random *Type I* instances with $n = 2000$, and eight random *Type II* instances. Table 1 reports, for each set of instances and each parameter combination, the average percentage deviation of the solution obtained with the Tabu_C_2 method and each parameter combination with respect to the best solution found (when running the seven variants). Similarly, Table 2 shows these values for the Tabu_D_2 method.

Tables 1 and 2 show that the best solutions on average are obtained with the values $\beta = 0.1$ and $\delta = 0.0001$ in both algorithms. Therefore, in subsequent experiments we will use these values.

In our next experiment we compare the best constructive methods known for the maximum diversity problem with our proposals. As documented in Glover et al. (1998), C_2 and D_2 provide better solutions than C_1 and D_1. Our results are in line with their previous experience and therefore we do not report results for C_1, D_1, GRASP_C_1, GRASP_D_1, Tabu_C_1 and Tabu_D_1, since we have found that C_2, D_2 and their variants systematically provide better results than them (and we want to focus on the comparison among competitive methods).

Tables 3–7 compare the GRASP_C_2, GRASP_D_2, Tabu_C_2, Tabu_D_2 and the previous approaches C_2, D_2, KLD, KLDv2 and MDI. These tables show, for each procedure, the average percentage deviation from the best solution found, the number of best solutions, and the number of constructions performed with each method. Since optimal solutions are not known, deviations are reported considering the best solution found during each experiment. We have limited the execution of each method to 10 seconds; therefore, the stopping parameter of each method is set to achieve this target value.

Tables 3–8 show that considering the four constructive methods by Glover et al. (1998), D_2 consistently outperforms the other three methods (we do not report the results for C_1 and D_1 to keep the size

Table 1
Preliminary experiment with Tabu_C_2

β	δ	Silva ($n \leqslant 500$), %	Type I ($n = 500$), %	Type I ($n = 2000$), %	Type II ($n = 500$), %
0.1	0	0.665	0.162	0.566	0.211
0.05	0	0.675	0.393	0.479	0.143
0.01	0	0.559	0.588	0.272	0.264
0.005	0	0.614	0.759	0.615	0.087
0.1	0.001	0.443	0.494	0.283	0.041
0.1	0.0005	0.437	0.530	0.283	0.027
0.1	0.0001	0.228	0.473	0.256	0.126

Table 2
Preliminary experiment with Tabu_D_2

β	δ	Silva ($n \leqslant 500$), %	Type I ($n = 500$), %	Type I ($n = 2000$), %	Type II ($n = 500$), %
0.1	0	0.280	0.182	0.041	0.340
0.05	0	0.731	0.011	0.022	0.259
0.01	0	0.846	0.290	0.009	0.103
0.005	0	1.123	0.372	0.023	0.151
0.1	0.001	0.224	0.003	0.019	0.321
0.1	0.0005	0.119	0.248	0.024	0.321
0.1	0.0001	0.000	0.171	0.028	0.321

Table 3
Best constructive methods – Silva instances

	C_2	D_2	KLD	KLDv2	MDI	GRASP_C_2	GRASP_D_2	Tabu_C_2	Tabu_D_2
Deviation	1.722%	1.079%	7.927%	15.657%	4.864%	0.297%	0.162%	0.621%	0.315%
#Best	0	0	0	0	0	6	13	1	8
#Const.	5140.5	2663.6	18,205.8	351.8	304.6	4273.7	1984.2	3749.4	1673.9

Table 4
Best constructive methods – Type I instances ($n = 500$, $m = 50$)

	C_2	D_2	KLD	KLDv2	MDI	GRASP_C_2	GRASP_D_2	Tabu_C_2	Tabu_D_2
Deviation	1.027%	0.729%	12.215%	17.240%	9.791%	0.174%	0.277%	0.473%	0.245%
#Best	0	3	0	0	0	10	4	2	5
#Const.	3002.9	840.0	7810.6	658.9	416.3	2719.9	663.7	2481.1	871.0

Table 5
Best constructive methods – Type I instances ($n = 2000$, $m = 200$)

	C_2	D_2	KLD	KLDv2	MDI	GRASP_C_2	GRASP_D_2	Tabu_C_2	Tabu_D_2
Deviation	0.392%	0.258%	8.806%	11.946%	6.497%	0.447%	1.586%	0.007%	0.258%
#Best	0	1	0	0	0	0	0	19	1
#Const.	151.5	33.9	552.6	16.7	7.1	135.5	27.7	128.3	34.2

Table 6
Best constructive methods – Type I instances ($n = 500$, $m = 200$)

	C_2	D_2	KLD	KLDv2	MDI	GRASP_C_2	GRASP_D_2	Tabu_C_2	Tabu_D_2
Deviation	0.487%	0.034%	3.208%	7.627%	0.750%	0.630%	0.127%	0.207%	0.001%
#Best	0	9	0	0	0	0	1	0	19
#Const.	1493.1	1205.7	7468.9	66.3	56.3	1203.3	917.3	987.8	1149.9

Table 7
Best constructive methods – Type II instances ($n = 500$, $m = 50$)

	C_2	D_2	KLD	KLDv2	MDI	GRASP_C_2	GRASP_D_2	Tabu_C_2	Tabu_D_2
Deviation	1.270%	1.219%	12.175%	17.358%	9.862%	0.207%	0.470%	0.493%	0.374%
#Best	0	0	0	0	0	10	2	4	4
#Const.	3050.1	858.5	7787.7	660.8	416.5	2690.3	655.8	2483.5	867.1

Table 8
Best constructive methods – Glover instances

	C_2	D_2	KLD	KLDv2	MDI	GRASP_C_2	GRASP_D_2	Tabu_C_2	Tabu_D_2
Deviation	0.021%	0.006%	6.371%	7.160%	7.325%	0.336%	0.024%	0.018%	0.006%
#Best	5	11	0	0	0	0	7	6	11
#Const.	2441.8	963.7	5998.1	417.9	550.0	2389.1	897.6	2149.6	971.0

of the tables relatively small) since it always obtains lower percentage deviations than them. Considering the three methods by Silva et al. (2004), KLD, KLDv2 and MDI, the best one in terms of solution quality is the MDI in all the set of instances considered (except in Glover instances, where the best

algorithm is KLD). However, the D_2 method always outperforms these three approaches. It should be noted that in these experiments all the methods run for 10 seconds, and within this time limit, D_2 is able to perform 1094.2 constructions on average, while MDI only performs 291.8 constructions on average.

Regarding our new approaches, Tables 3–8 show that in some sets of instances the GRASP constructions provide the best results. This is the case of Tables 3, 4 and 7 in which the GRASP_C_2 and GRASP_D_2 methods present average percentage deviations of 0.297, 0.162, respectively in Silva instances (Table 3), 0.174, 0.277 in Type I instances (Table 3), and 0.207, 0.470 in Type II instances (Table 7). On the other hand, in the other types of instances, the tabu variants obtain the best results. Specifically, in Table 5 the Tabu_C_2 method presents an average percentage deviation of 0.007, while in Tables 6 and 8 the Tabu_D_2 method achieves a percentage deviation of 0.001 and 0.006, respectively.

Considering Type I instances with $n = 500$, in Table 4 ($m = 50$) we can observe relatively large deviations from the best solution found in some methods (see for example the 17.240% for the KLDv2). On the other hand, in Table 6 ($m = 200$) the average percent deviations are considerably smaller (with a maximum of 7.627 for the KLDv2). In this sense we can say that, in this type of instances, the larger the density of the graph, the easier the problem is to solve.

Overall the results in Tables 3–8 indicate that the best method is the Tabu_D_2, closely followed by GRASP_C_2. The ranking of the methods considering the average value of the average percentage deviation across the six tables is: Tabu_D_2 (0.200%), Tabu_C_2 (0.303%), GRASP_C_2 (0.348%), GRASP_D_2 (0.441%), D_2 (0.554%), C_2 (0.820%), MDI (6.515%), KLD (8.450%), and KLDv2 (12.831%).

In our next experiment we undertake to compare the best methods for solving the maximum diversity problem. Specifically, we consider the two best constructions, Tabu_D_2 and GRASP_C_2, and the three improving methods described in Section 4: LS, I_LS and LS_TS. Each combination produces a multi-start method in which an iteration consists of a construction phase, performed with one of the two former methods, and an improving phase, performed with one of the three later methods. After preliminary experimentation the tabu search parameters in the LS_TS method, *TabuTenure* and *MaxIter*, were set to 14 and 25, respectively. Tables 9–14 report the comparison among these six combinations. As in previous experiments, these tables show, for each procedure, the average percentage deviation from the best solution found, the number of best solutions that each method is able to match, and the number of constructions with their corresponding improvement performed with each method. Since optimal solutions are not known, deviations are reported considering the best solution found during each experiment.

Tables 9–14 show that in all the sets of instances and in both types of constructions, the local search by Ghosh (LS) provides lower quality results than the improved local search (I_LS), with the exception of the Glover instances (Table 14) in which all the variants are able to obtain the best result. Moreover, the LS_TS method, which incorporates memory structures, improves upon the other two local search variants in all the cases. For instance, in Table 10 (Type I instances with $n = 500$ and $m = 50$) we can observe in the Tabu_D_2 method, an average percentage deviation of 0.182%, 0.092% and 0.038% for the LS, I_LS and LS_TS respectively. Similarly, in the same table, the GRASP_C2 method presents 0.265%, 0.185% and 0.052% deviations for the LS, I_LS and LS_TS variants. Comparing the improved local search, I_LS, with the original local search LS by Ghosh (1996), we can conclude from the number of constructions, #Const, (with their corresponding improvement) that the former is much more efficient than the LS method since it is able to perform about four times the number of iterations.

Considering the 120 instances together and the number of best solutions that each method is able to match, the Tabu_D_2 construction coupled with the LS_TS improvement is the best method overall, since it is able to obtain 61 best solutions, which compares favorably with the other combinations (53 for the GRASP_C_2 + LS_TS, 51 for the Tabu_D_2 + I_LS, 39 for the Tabu_D_2 + Ghosh, 38 for the GRASP_C_2 + I_LS and 36 for the GRASP_C_2 + Ghosh). It should be noted that, as expected, the tabu search improvements are more time-consuming than the memory-less improvements I_LS (see a lower number of constructions #Const. for the LS_TS in all the cases); however, they are memory-guided, and are able to reach better solutions in a lower number of iterations than the greedy I_LS improvement method.

Table 9
Best methods (construction + improvement) – 20 Silva instances

	Tabu_D_2			GRASP_C_2		
	LS	I_LS	LS_TS	LS	I_LS	LS_TS
Deviation	0.0443%	0.0446%	0.0130%	0.0563%	0.0377%	0.0332%
#Best	9	13	13	10	12	13
#Const.	443.1	1463.1	864.1	132.7	925.4	644.4

Table 10
Best methods (construction + improvement) – Type I instances ($n = 500$, $m = 50$)

	Tabu_D_2			GRASP_C_2		
	LS	I_LS	LS_TS	LS	I_LS	LS_TS
Deviation	0.182%	0.092%	0.038%	0.265%	0.185%	0.052%
#Best	3	6	9	0	1	8
#Const.	124.6	728.3	523.7	237.0	555.4	432.0

Table 11
Best methods (construction + improvement) – Type I instances ($n = 2000$, $m = 200$)

	Tabu_D_2			GRASP_C_2		
	LS	I_LS	LS_TS	LS	I_LS	LS_TS
Deviation	0.284%	0.101%	0.099%	0.334%	0.232%	0.204%
#Best	1	7	6	0	2	4
#Const.	4.3	16.7	14.8	3.1	13.7	12.0

Table 12
Best methods (construction + improvement) – Type I instances ($n = 500$, $m = 200$)

	Tabu_D_2			GRASP_C_2		
	LS	I_LS	LS_TS	LS	I_LS	LS_TS
Deviation	0.038%	0.024%	0.013%	0.036%	0.028%	0.018%
#Best	4	3	5	5	0	3
#Const.	66.2	315.4	198.5	47.9	137.2	76.1

Table 13
Best methods (construction + improvement) – Type II instances ($n = 500$, $m = 50$)

	Tabu_D_2			GRASP_C_2		
	LS	I_LS	LS_TS	LS	I_LS	LS_TS
Deviation	0.141%	0.094%	0.047%	0.271%	0.162%	0.129%
#Best	2	2	8	1	3	5
#Const.	312.9	728.2	517.3	237.3	581.2	138.5

Table 14
Best methods (construction + improvement) – Glover instances

	Tabu_D_2			GRASP_C_2		
	LS	I_LS	LS_TS	LS	I_LS	LS_TS
Deviation	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
#Best	20	20	20	20	20	20
#Const.	243.3	602.1	397.5	474.4	790.4	510.3

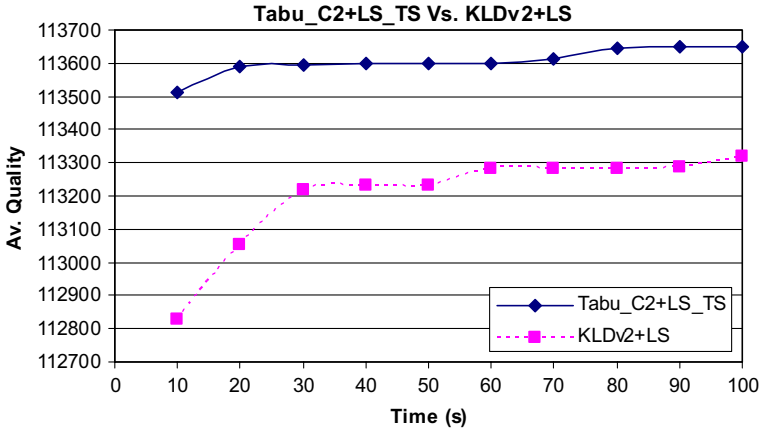

Fig. 6. Average value of the best solution found over time.

In our last experiment, we consider 10 large Type I instances with $n = 2000$ and $m = 200$. The experiment has the goal of showing how the average value of the best solution found improves over time. We compare our best approach Tabu_D_2 + LS_TS with the best previously published method: KLDv2 + LS (see Silva et al., 2004). Both procedures were run for 100 seconds and the best solution found was reported every 10 seconds. The results of this experiment are shown in Fig. 6.

The diagram in Fig. 6 clearly shows that the Tabu_D2 + LS_TS method is able to obtain high quality solutions from the first iterations (within 10 seconds). When the optimization process starts, this procedure quickly moves to the range of high quality solutions and maintains its lead during the rest of the solution time. On the contrary, the KLDv2 + LS needs 30 seconds to obtain solutions of relatively good quality, although inferior to those obtained with Tabu_D2 + LS_TS, and from this point on it only experiences a moderate improvement.

The tables in Appendix A contain the best solutions achieved with our tabu search method, Tabu_D_2 + LS_TS, in each particular instance within 10 seconds of CPU time on a Pentium IV at 3 GHz.

6. Conclusions

The objective of our study has been to compare memory-based with memory-less designs to solve the maximum diversity problem. Unlike local search methodologies, memory structures have not yet been extensively studied in the context of constructive methods. In this paper we have proposed four constructive methods based on tabu search which incorporates memory structures, and four memory-less constructions based on GRASP methodology. These eight procedures can be coupled with an improving phase. We have proposed two variants, an efficient local search (as compared with a previous development) and a short-term tabu search improvement phase. Overall, experiments with 120 instances clearly show the effectiveness of the use of memory in both construction and improvement for solving this problem.

We have identified seven previous relevant procedures for this problem (Glover et al., 1998; Silva et al., 2004). Our Adaptive Memory Programming implementation (construction + local search with memory structures) was shown to be competitive in the problem instances considered, outperforming those seven previous approaches.

The proposed procedures can be used in the context of evolutionary methods. These methods based their performance on maintaining a good balance between quality and diversity during the search process. One of the key steps is the construction of the set of solutions that will be subject to combination. In some methods, for example Scatter Search (Laguna and Martí, 2003) the set of solutions in which the search takes place is selected from among the solutions in a bigger initial set. Our proposals can be used in this initial step to obtain a set of good and maximal diverse solutions. Then, the maximum diversity problem would be solved to obtain this "representative" set of good an different solutions, obtaining thus a balance between quality and diversity from the very beginning of the search.

Acknowledgements

Research by Rafael Martí is partially supported by the *Ministerio de Educación y Ciencia* (refs. TIN2004-20061-E and TIC2003-C05-01) and by the *Agencia Valenciana de Ciència i Tecnologia* (ref. GRUPOS03/189).

Appendix A

See Tables 15 and 16.

Table 15
Tabu_D_2 + LS_TS results within 10 seconds

Instance	Value	Instance	Value	Instance	Value
Silva.1 (100, 10)	333	Type2.1	777,890.13	Glover.1	19,485.21
Silva.2 (100, 20)	1195	Type2.2	779,963.81	Glover.2	19,701.57
Silva.3 (100, 30)	2457	Type2.3	776,471.44	Glover.3	19,547.23
Silva.4 (100, 40)	4142	Type2.4	775,308.75	Glover.4	19,596.49
Silva.5 (200, 20)	1247	Type2.5	775,611.44	Glover.5	19,602.63
Silva.6 (200, 40)	4450	Type2.6	774,360.06	Glover.6	19,421.56
Silva.7 (200, 20)	9437	Type2.7	776,388.75	Glover.7	19,534.33
Silva.8 (200, 80)	16,225	Type2.8	778,223.50	Glover.8	19,487.36
Silva.9 (300, 30)	2694	Type2.9	774,801.75	Glover.9	19,221.66
Silva.10 (300, 60)	9689	Type2.10	773,941.00	Glover.10	19,703.37
Silva.11 (300, 90)	20,734	Type2.11	776,950.31	Glover.11	19,587.14
Silva.12 (300, 120)	35,878	Type2.12	772,770.69	Glover.12	19,360.25
Silva.13 (400, 40)	4655	Type2.13	780,191.94	Glover.13	19,366.73
Silva.14 (400, 80)	16,948	Type2.14	781,655.94	Glover.14	19,458.59
Silva.15 (400, 120)	36,298	Type2.15	780,300.63	Glover.15	19,422.18
Silva.16 (400, 160)	62,456	Type2.16	775,436.06	Glover.16	19,680.24
Silva.17 (500, 50)	7133	Type2.17	774,662.81	Glover.17	19,331.42
Silva.18 (500, 100)	26,254	Type2.18	775,501.06	Glover.18	19,461.42
Silva.19 (500, 150)	56,572	Type2.19	778,802.75	Glover.19	19,477.35
Silva.20 (500, 200)	97,344	Type2.20	778,571.75	Glover.20	19,604.88

Table 16
Tabu_D_2 + LS_TS results in Type I instances within 10 seconds

(500, 50)	Value	(500, 200)	Value	(2000, 200)	Value
Type1_55.1	7833.82	Type1_52.1	107,394.77	Type1_22.1	113,840
Type1_55.2	7754.9	Type1_52.2	107,155.99	Type1_22.2	113,836
Type1_55.3	7749.64	Type1_52.3	107,247.66	Type1_22.3	113,531
Type1_55.4	7759.12	Type1_52.4	106,986.30	Type1_22.4	113,425
Type1_55.5	7748.51	Type1_52.5	106,922.26	Type1_22.5	113,416
Type1_55.6	7763.04	Type1_52.6	107,164.53	Type1_22.6	113,961
Type1_55.7	7752.7	Type1_52.7	107,040.06	Type1_22.7	113,701
Type1_55.8	7735.16	Type1_52.8	107,014.24	Type1_22.8	113,728
Type1_55.9	7753.93	Type1_52.9	107,476.19	Type1_22.9	113,698
Type1_55.10	7778.65	Type1_52.10	107,195.75	Type1_22.10	113,571
Type1_55.11	7769.61	Type1_52.11	107,144.27	Type1_22.11	113,631
Type1_55.12	7757.65	Type1_52.12	106,812.73	Type1_22.12	113,648
Type1_55.13	7783.80	Type1_52.13	107,631.87	Type1_22.13	113,456
Type1_55.14	7791.08	Type1_52.14	107,403.87	Type1_22.14	113,602
Type1_55.15	7718.71	Type1_52.15	107,006.89	Type1_22.15	113,850
Type1_55.16	7792.77	Type1_52.16	107,370.02	Type1_22.16	113,617
Type1_55.17	7785.98	Type1_52.17	107,056.92	Type1_22.17	113,757
Type1_55.18	7756.26	Type1_52.18	106,953.64	Type1_22.18	113,733
Type1_55.19	7755.41	Type1_52.19	107,008.59	Type1_22.19	113,558
Type1_55.20	7733.86	Type1_52.20	106,735.14	Type1_22.20	113,574

References

Feo, T., Resende, M.G.C., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133.

Fleurent, C., Glover, F., 1999. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS Journal on Computing 11 (2), 198–204.

Ghosh, J.B., 1996. Computational aspects of the maximum diversity problem. Operations Research Letters 19, 175–181.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publisher.

Glover, F., Kuo, C.C., Dhir, K.S., 1995. A discrete optimization model for preserving biological diversity. Applied Mathematical Modeling 19, 696–701.

Glover, F., Kuo, C.C., Dhir, K.S., 1998. Heuristic algorithms for the maximum diversity problem. Journal of Information and Optimization Sciences 19 (1), 109–132.

Kuo, C.C., Glover, F., Dhir, K.S., 1993. Analyzing and modeling the maximum diversity problem by zero-one programming. Decision Sciences 24 (6), 1171–1185.

Laguna, M., Martí, R., 2003. Scatter Search. Methodology and Implementations in C. Kluwer Academic Publishers.

Laguna, M., Martí, R., Campos, V., 1999. Intensification and diversification with elite tabu search solutions for the linear ordering problem. Computers and Operations Research 26, 1217–1230.

McConnell, S., 1988. The new battle over immigration. Fortune 117 (10), 89–102.

Pearce, D., 1987. Economics and genetic diversity. Futures 19 (6), 710–712.

Porter, W.M., Eawal, K.M., Rachie, K.O., Wien, H.C., Willians, R.C., 1975. Cowpea germplasm catalog No. 1, International Institute of Tropical Agriculture, Ibadan, Nigeria.

Resende, M.G.C., Ribeiro, C.C., 2001. Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (Eds.), State-of-the-art Handbook in Metaheuristics. Kluwer Academic Publishers, Boston, pp. 219–250.

Silva, G.C., Ochi, L.S., Martins, S.L., 2004. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problemLecture Notes in Computer Science, vol. 3059. Springer, pp. 498–512.

Swierenga, R.P., 1977. Ethnicity in historical perspective. Social Science 52 (1), 31–44.