
Hybrid Heuristics for the Maximum Diversity Problem

MICAEL GALLEGO

Departamento de Informática, Estadística y Telemática, Universidad Rey Juan Carlos,
Spain. Micael.Gallego@urjc.es

ABRAHAM DUARTE

Departamento de Informática, Estadística y Telemática, Universidad Rey Juan Carlos,
Spain. Abraham.Duarte@urjc.es

MANUEL LAGUNA

Leeds School of Business, University of Colorado at Boulder, USA
laguna@colorado.edu

RAFAEL MARTÍ

Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain
Rafael.Marti@uv.es

ABSTRACT

The maximum diversity problem presents a challenge to solution methods based on heuristic optimization. We undertake the development of hybrid procedures within the scatter search framework with the goal of uncovering the most effective designs to tackle this difficult but important problem. Our research revealed the effectiveness of adding simple memory structures (based on recency and frequency) to key scatter search mechanisms. Our extensive experiments and related statistical tests show that the most effective scatter search variant outperforms state-of-the-art methods.

Latest Revision: June 9, 2006

1. Introduction

The maximum diversity problem (MDP) consists of selecting a subset of m elements from a set of n elements in such a way that the sum of the distances between the chosen elements is maximized. The notion of distance between elements is customized to specific applications. As mentioned in Kuo, Glover and Dhir (1993) and Glover, Kuo and Dhir (1998), the maximum diversity problem has applications in plant breeding, social problems, ecological preservation, pollution control, product design, capital investment, workforce management, curriculum design and genetic engineering. In most applications, it is assumed that each element can be represented by a set of attributes. Let s_{ik} be the state or value of the k^{th} attribute of element i , where $k = 1, \dots, K$. Then the distance between elements i and j may be defined as:

$$d_{ij} = \sqrt{\sum_{k=1}^K (s_{ik} - s_{jk})^2}$$

In this case, d_{ij} is simply the Euclidean distance between i and j . The distance values are then used to formulate the MDP as a quadratic binary problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} x_i x_j \\ \text{Subject to} \quad & \sum_{i=1}^n x_i = m \\ & x_i = \{0, 1\} \quad 1 \leq i \leq n \end{aligned}$$

Kuo, Glover and Dhir (1993) use this formulation to show that the clique problem (which is known to be NP-complete) is reducible to the MDP. These authors also suggest the transformation of the quadratic binary model into a mixed integer program of the following form:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} y_{ij} \\ \text{Subject to} \quad & \sum_{i=1}^n x_i = m \\ & x_i + x_j - y_{ij} \leq 1 \quad 1 \leq i < j \leq n \\ & -x_i + y_{ij} \leq 0 \quad 1 \leq i < j \leq n \\ & -x_j + y_{ij} \leq 0 \quad 1 \leq i < j \leq n \\ & y_{ij} \geq 0 \quad 1 \leq i < j \leq n \\ & x_i = \{0, 1\} \quad 1 \leq i \leq n \end{aligned}$$

Note that the second set of constraints is redundant and was eliminated in the formulation presented in the 1998 article by the same authors. This formulation produces a very weak

linear programming relaxation. Specifically, when the diversity values are uniformly distributed, the LP relaxation results in $x_i = \frac{m}{n}$ for all i . Hence, most branching rules implemented in general-purpose MIP solvers fail to identify promising separation variables and branching directions. Experiments with Cplex 10.0.1 corroborate the difficult that commercial branch-bound codes encounter when approaching the maximum diversity problem with this formulation.

We do not include a discussion of previous work on the maximum diversity problem because fairly complete reviews have appeared in recent publications, including Silva, Ochi and Martins (2004) and Duarte and Martí (2005). Our main contribution is the development and testing of a solution method based on the scatter search framework that outperforms the best approximation procedures reported in the literature. Specifically, we compare our proposed procedure to two GRASP approaches developed by Silva, Ochi and Martins (2004) and the tabu search due to Duarte and Martí (2006).

2. Scatter Search Approach

Scatter search (SS) is a metaheuristic framework that explores a solution space by evolving a set of reference points. The search starts with the application of a diversification generation method that results in a population of points from which a subset is selected as the initial reference set (*RefSet*). The evolution of the reference set is induced by the application of four additional methods: subset generation, combination, improvement and update. The diversification generation method may be used again if the rebuilding of the reference set becomes necessary. Typically, the rebuilding phase is triggered after an iteration in which no new solutions become part of the current reference set. A detailed description of the method and a comprehensive list of applications appear in the book by Laguna and Martí (2003). Figure 1 summarizes the basic scatter search design.

1. Construction of a set P consisting of $PopSize$ diverse solutions via the application of the diversification generation method
2. Application of the improvement method to all solutions in P
3. Construction of the initial reference set. *RefSet* consists of b solutions in P , from which $q\%$ are chosen due to their quality (as measured by the objective function value) and $(1-q)\%$ are selected due to their diversity (as measure by the distance between them and the rest of the solutions already in the reference set)
4. Application of the subset generation method to create a list of all subsets of reference solutions that will be combined
5. Application of the combination method to all subset of reference solutions generated in the previous step
6. Application of the improvement method to all the trial solutions generated by the combination method
7. Updating of the *RefSet* if any new trial solution is better than any of the current reference solutions
8. If a new solution has been included in the *RefSet* then go back to step 4, otherwise the procedure stops.

Figure 1. Basic scatter search procedure

An extension of the basic scatter search design considers the rebuilding of the *RefSet*. This means that instead of stopping when no new solutions are added to the reference set, the diversification generation method is invoked again to generate a brand new population of solutions (i.e., a new P). The new *RefSet* is built with the best $q\%$ from the current *RefSet* and $(1-q)\%$ diverse solutions from P . After the *RefSet* is rebuilt, the procedure goes back to step 4. The search stops after a given execution time limit or a limit on the number of rebuilding steps. In our implementation, instead of constructing a new P during the rebuilding step, we use the solutions remaining in P from the previous rebuilding step. If the set becomes empty before the procedure terminates, then the diversification method is employed to repopulate it.

The definition of distance between solutions is a key design issue in scatter search implementations. Distance is used to measure how diverse one solution is with respect to a set of solutions. Specifically, for the MDP, let x_i^r be the value of the i^{th} variable for the reference solution r (i.e., $r \in \text{RefSet}$). Also let x_i^t be the value of the i^{th} variable for the trial solution t . Then, the distance between the trial solution t and the solutions in the *RefSet* in our SS implementation is defined as:

$$\text{distance}(t, \text{RefSet}) = bm - \sum_{r=1}^b \sum_{i:x_i^r=1} x_i^t$$

The formula simply counts the number of times that each selected element in the trial solution t appears in the reference solutions and subtracts this value from the maximum possible distance (i.e., bm). The maximum distance occurs when no element that is selected in the trial solution t appears in any of the reference solutions. When choosing solutions to rebuild the reference set, we select the trial solution that has the maximum distance between itself and the solutions currently in the *RefSet*. Since the solutions are added one at a time, the distance calculations have to be updated before the next solution is selected.

Within the framework described above, we implemented three variants of the scatter search procedure. The differences among these implementations are related to the methods used to construct, combine and improve solutions, as indicated in Table 1. The Base method does not use any specific information about the problem context. The GRASP Hybrid implements several strategies that take advantage of characteristics that are specific to the MDP. The Tabu Search Hybrid uses both context information and memory structures that are typical to tabu search implementations (Glover and Laguna 1997).

As shown in Table 1, the base procedure generates diversification by simply selecting m elements at random. The diversification generated in this way refers to the distances between the solutions and not about the objective function values. The GRASP Hybrid employs a more elaborate procedure for generating diverse solution. The procedure, referred to as GRASP_ D -2 and developed by Duarte and Martí (2006), is based on randomizing D -2, a deterministic destructive heuristic developed by Glover, Kuo and

Dhir (1998). $D-2$ starts with the infeasible solution for which $x_i = 1$ for all i . That is, all n elements are originally selected. In order to reduce the set of selected elements to m , the procedure performs $n-m$ steps. At each step, the procedure deselects element i^* (i.e., x_{i^*} is set to zero), where i^* is such that

$$D(i^*) = \underset{i: x_i=1}{\text{Min}}(D(i)),$$

$$\text{and } D(i) = \sum_j d_{ij} x_i x_j.$$

The randomization of $D-2$ that is employed within the GRASP_ $D-2$ consists of selecting i^* from a reduced candidate list formed by all those elements i such that $D(i) \leq (1 + \alpha)D(i^*)$. The value of α is initially set to 0.5 and decreased by 0.1 after a pre-specified CPU time is consumed without improving the incumbent to a minimum of 0.1. We set this value to a 20% of the total CPU time in our implementation.

Procedure	Diversification Generation	Combination	Improvement
Base	Random selection of m elements from all n elements in the problem	Random selection of m elements from the union of the elements in the solutions being combined	LS, “largest improvement” local search
GRASP Hybrid	GRASP_ $D-2$, based on the randomization of the destructive heuristic $D-2$	Application of $D-2$ to the union of the elements in the solutions being combined	I_LS, “first improvement” local search
Tabu Search Hybrid	Tabu_ $D-2$, based on adding memory structures to $D-2$	Application of Tabu_ $D-2$ to the union of the elements in the solutions being combined	Local search with short term memory

Table 1. Summary of scatter search methods implemented for testing

The diversification generator within the Tabu Search Hybrid variant is also based on the destructive procedure $D-2$. At each step of the procedure, the element i^* to be deselected is such that:

$$D(i^*) = \underset{i: x_i=1}{\text{min}} \left(D(i) - \beta(\text{range}) \left(\frac{f(i)}{f_{\max}} \right) + \delta(\text{range}) \left(\frac{q(i)}{q_{\max}} \right) \right),$$

$$\text{where } \text{range} = \max_{i: x_i=1} (D(i)) - \min_{i: x_i=1} (D(i))$$

In this modified distance calculation, $f(i)$ indicates the frequency in which element i has appeared in previous solutions and $q(i)$ is the average quality (as measured by the objective function value) of past solutions that included element i . The f_{\max} and q_{\max} are the maximum values of f and q overall elements. The penalty factors β and δ are respectively set to 0.1 and 0.0001 in our experiments.

The Local Search method LS (Ghosh 1996) scans the set of selected elements in search of the best exchange to replace a selected element with an unselected one. The method performs moves as long as the objective value increases and it stops when no improving exchange can be found. The Improved Local Search method, I_LS, (Duarte and Martí 2006) selects the element i^* ($x_{i^*} = 1$) that provides the smallest contribution to the objective function value of the current solution. Then, it searches for an element j ($x_j = 0$) to be exchange with element i^* . The first element j that results in an improving move is selected and the exchange is performed without examining the remaining unselected elements. If no improving move can be found to exchange element i^* , then the selected element with the next smallest contribution is examined. This process continues until no improving exchange can be found.

LS_TS (Duarte and Martí 2006) implements a short-term tabu search method also based on exchanges. An iteration of this method begins with a random selection of an element i ($x_i = 1$). The probability of selecting element i is inversely proportional to $D(i)$. The list of unselected elements is scanned and the first improving move that exchanges elements i and j ($x_j = 0$) is selected. If no improving move is found, then the least non-improving move is chosen. The chosen exchange is performed and both elements participating in the exchange are classified tabu-active for a number of iterations (known as the tabu tenure). Tabu-active elements are not allowed to participate in any exchanges. The LS_TS method stops if after a number of consecutive iterations the incumbent solution is not modified.

The original LS_TS method was modified when added to our scatter search framework. The modification consists of using an asymmetric tabu tenure in which elements added to the solution are given shorter tabu tenures than the tenure assigned to those elements that have been deleted from the solution. Also, the tabu tenure and the maximum number of iterations have been made dependent on the number of elements in the solution. According to the experimentation reported in Duarte and Martí (2006), the tabu tenure for selected elements is set to $0.28m$, while the tabu tenure for unselected elements is set to $0.028m$. The maximum number of iterations without improvement is set to $0.1n$.

During preliminary experimentation, we observed that the diversification and combination methods yielded the same solutions more than once. In order to avoid the application of the improvement method to a solution more than once, we developed a filtering method based on the hash function in Figure 2. The method is capable of filtering 0.5% of the GRASP Hybrid solutions and 82% of the Tabu Search Hybrid solutions.

```

Let  $hash = 1$ 
for ( $i = 1, \dots, n$ )
     $hash = 31hash + 1231 x_i + 1237(1 - x_i)$ 
end for

```

Figure 2. Hash function calculation

A *hash* value is stored for each solution generated during the search. The hash value of a solution that is candidate for improvement is checked against the database of hash values. If a match is found then the solution under consideration is not subjected to the improvement method.

2.1. Combination Methods

In this subsection, we describe the mechanisms that we have developed to combine solutions during the scatter search. We use the distance matrix in Table 2 to illustrate the processes that yield new solutions as combinations of reference solutions. The dimensions of the example problem are $n = 10$ and $m = 3$.

	2	3	4	5	6	7	8	9	10
1	2.65	2.83	2.65	2.00	2.83	2.45	2.65	3.00	2.65
2		3.32	3.74	1.73	1.00	2.65	3.16	1.41	2.00
3			2.65	3.46	3.16	2.45	2.65	2.65	4.12
4				3.87	3.61	3.87	2.00	3.46	3.74
5					2.00	2.00	3.87	2.24	1.73
6						2.83	3.32	1.73	1.73
7							3.87	2.24	3.32
8								3.16	4.00
9									2.83

Table 2. Distances between elements

Suppose that during the scatter search two reference r_1 and r_2 solutions are to be combined. For ease of notation, we represent each solution as the set of the selected elements: $r_1 = \{1, 4, 6\}$ and $r_2 = \{1, 5, 10\}$. The objective function values associated with these solutions are 9.09 ($2.65 + 2.38 + 3.61$) and 6.38 ($2 + 2.65 + 1.73$), respectively.

Random Selection

This method consists of selecting m elements from the union of the elements in both reference solutions. In our example, the union of the elements is $U = \{1, 4, 5, 6, 10\}$. The random selection consists of choosing 3 elements from U . For example, the new trial solution may be $t = \{1, 4, 10\}$ with objective function 9.04 ($2.65 + 2.65 + 3.74$).

D-2 Selection

This method consists of the application of the destructive heuristic *D-2* (Glover, Kuo and Dhir, 1998) to the union of the elements in the reference solutions being combined. The method starts with the selection of all elements in the union and then it deselects one element at a time until there are only m selected elements remaining. The element i that is deselected at each step is the one with the minimum $D(i)$ value. In our example, the union consists of 5 elements and therefore the method performs two steps only. In the first step, the D values are:

$$\begin{aligned}
D(1) &= d(1,4) + d(1,5) + d(1,6) + d(1,10) = 2.65 + 2 + 2.83 + 2.65 = 10.13 \\
D(4) &= d(4,1) + d(4,5) + d(4,6) + d(4,10) = 2.65 + 3.87 + 3.61 + 3.74 = 13.87 \\
D(5) &= d(5,1) + d(5,4) + d(5,6) + d(5,10) = 2 + 3.87 + 2 + 1.73 = 13.22 \\
D(6) &= d(6,1) + d(6,4) + d(6,5) + d(6,10) = 2.83 + 3.61 + 2 + 1.73 = 10.17 \\
D(10) &= d(10,1) + d(10,4) + d(10,5) + d(10,6) = 2.65 + 3.74 + 1.73 + 1.73 = 9.85
\end{aligned}$$

The minimum D value corresponds to element 10 and therefore this element is deselected. The updated union is $U = \{1, 4, 5, 6\}$. For the next step, the corresponding D values are 7.48, 10.13, 7.87 and 8.44. Therefore, element 1 is deselected and the trial solution that results from the application of this combination method is $t = \{4, 5, 6\}$ with an objective function value equal to 9.48.

Tabu_D-2 Selection

This method consists of the application of the Tabu_D-2 procedure to the union of the elements of the reference solutions being combined. This method uses information about solutions generated in the past as well as information associated with those solutions combined in previous iterations. To illustrate how this procedure works, we will assume that after a number of iterations, the information in Table 3 is available.

Element	Frequency	Quality
1	13	7.73
2	19	7.25
3	17	8.69
4	8	9.37
5	45	6.64
6	17	7.71
7	16	8.42
8	16	9.09
9	19	7.44
10	13	8.50

Table 3. Information about past generated and combined solutions

We use the data in Table 3 to calculate the modified D values in the same way as described above for the diversification generator based on Tabu_D-2.

$$\begin{aligned}
D(1) &= 10.13 - (0.02)(13) + (0.000043)(7.73) = 9.87 \\
D(4) &= 13.87 - (0.02)(8) + (0.000043)(9.37) = 13.71 \\
D(5) &= 13.22 - (0.02)(45) + (0.000043)(6.64) = 12.32 \\
D(6) &= 10.17 - (0.02)(17) + (0.000043)(7.71) = 9.83 \\
D(10) &= 9.85 - (0.02)(13) + (0.000043)(8.50) = 9.59
\end{aligned}$$

The minimum D value corresponds to element 10 and therefore this element is deselected. The updated union is $U = \{1, 4, 5, 6\}$. For the next step, the corresponding D values are 7.30, 10, 7.24 and 8.20. Therefore, element 5 is deselected and the trial solution that results from the application of this combination method is $t = \{1, 4, 6\}$ with an objective function value equal to 9.09.

3. Alternative Scatter Search Designs

In order to determine the best configuration of our scatter search procedure, we performed 5 preliminary experiments. We used two types of problem instances:

- Type I — Diversity values are real numbers uniformly distributed between 0 and 1000
- Type II — Diversity values are real numbers uniformly distributed between 0 and 10

We generate 10 problems of each type with $n = 500$ and $m = 50$. We use the outcomes of our experiments to calculate the average percent deviation ($pDev$) of the solutions obtained by each procedure when compared to the best solutions during the given experiments. We also report on the number of best solutions ($nBest$) found by each method.

3.1. Search Framework and Improvement Method

The objective of this experiment is to determine the relative merit of the scatter search variants described in Table 4. We set a time limit of 3 CPU minutes and we run each procedure with and without the improvement method. The results are summarized in Table 4.

Procedure	Improvement	Average Deviation	Number of Best
Base	No	13.7%	0
	Yes	0.68%	0
GRASP Hybrid	No	1.16%	0
	Yes	0.18%	2
Tabu Search Hybrid	No	1.07%	0
	Yes	0.00%	20

Table 4. Summary of results for preliminary experiment 1

The results in Table 4 indicate the advantage of using an improvement method within our design. In terms of average deviation from the best solutions, the improvement method has the largest impact in the case of the Base design. Also, the improvement method makes a significant difference in the number of best solutions found in the Tabu Search Hybrid. In general, we conclude that the procedures embedded in the Tabu Search Hybrid variant results in the best scatter search configuration. For the remaining of this paper, the scatter search that we use is the one that implements the Tabu Search Hybrid procedures (including the improvement method), as described in Table 1.

3.2. Reference Set Configuration

The objective of this experiment is testing the selective application of the improvement method. So far, our scatter search is such that every solution generated by the diversification generation method or by the combination method is subjected to the improvement method. Since the execution of the improvement method is computationally expensive, applying it to every solution may prevent the search from visiting additional solutions during the allotted search time. Therefore, in this experiment, we test the idea of selectively applying the improvement method to a subset of the solutions that are visited during the search. Specifically, we skip step 2 in Figure 1 and select the best $(q\%)b$ solutions from P . Once these solutions have been added to the *RefSet*, then the improvement method is applied to these solutions only. Similarly, after the application of the combination method, the improvement method is not applied to all the resulting solutions. Instead, the best $(q\%)b$ are selected and the improvement method is applied to this subset only. The selective application of the improvement method may result in a trajectory that misses a high-quality solution that could have been found when applying the improvement method to a relatively inferior solution. This is why we designed this preliminary experiment with the goal of identifying parameter settings (i.e., values for b and q) that would be effective under the selective application of the improvement method. Table 5 summarizes the results of this experiment.

b	$q\%$	Improve All		Selective Improvement	
		Avg. Deviation	Num. of Best	Avg. Deviation	Num. of Best
8	90%	0.0200%	15	0.1065%	5
	70%	0.0094%	14	0.0653%	11
	50%	0.0181%	16	0.0690%	9
	30%	0.0017%	19	0.0663%	11
10	90%	0.0148%	14	0.0937%	7
	70%	0.0134%	15	0.0777%	8
	50%	0.0069%	16	0.0924%	8
	30%	0.0017%	19	0.1282%	7
20	90%	0.0937%	17	0.0622%	11
	70%	0.0017%	19	0.0569%	11
	50%	0.0017%	19	0.0397%	14
	30%	0.0000%	20	0.0616%	11
40	90%	0.0000%	20	0.0221%	17
	70%	0.0017%	19	0.0437%	13
	50%	0.0000%	20	0.0443%	13
	30%	0.0000%	20	0.0488%	12

Table 5. Summary of results for preliminary experiment 2

Regarding the average deviation, the results in Table 5 reveal that the best average performance when applying the improvement method to all solutions is achieved when q is set to 30. Likewise, the best average deviation for the selective procedure is achieved when q is set to 70. When considering average deviation and number of best solutions, the “Improve All” variant outperforms the selective application of the improvement method.

Given the effectiveness of the “Improve All” version with $q = 30$, we performed an additional experiment where we varied the values of b from 4 to 160. The experiment

showed that the best results are found when $b = 12$ and therefore we set the parameters to these preferred values, 12 for b and 30 for q .

3.3. Subset Generation Strategies

The purpose of this experiment is identifying the most effective method for generating subsets of reference solutions that are in turn the input to the combination method. For this experiment, we consider combinations of 2, 3, 4 and 5 solutions. Our subset generation method (see step 4 in Figure 1) operates as described in Section 2 of Chapter 5 in Laguna and Martí (2003). All subsets of size 2 are considered. That is, all pairs of reference solutions are added to the list of subsets. Subsets of size 3 are constructed by considering each subset of size 2 and adding the best reference solution that is not part of the subset. Subsets of higher dimensions are constructed following the same logic. That is, subsets of size 4 are based on subsets of size 3. Likewise, subsets of size 5 are constructed by adding a solution to subsets of size 4. This mechanism avoids the exponential explosion in the number of subsets generated had we considered all possible subsets of size 3, 4 and 5.

The experiment consists of using the scatter search procedure, as configured after the previous preliminary experiments, and testing the merit of four variants of the subset generation method:

- SG1: Generate all subsets of size 2. This method generates all pairs of reference solutions and therefore it results in $b(b-1)/2$ subsets that are passed to the combination method.
- SG2: Generate all subsets of size 2 and then augment each pair to generate subsets of size 3. The way a solution is added to each pair creates duplicates and therefore $b(b-1)$ is an upper bound on the number of subsets generated by this variant.
- SG3: Augment SG2 with subsets of size 4 that are generated by adding a solution to the subsets of size 3.
- SG4: Augment SG3 with subsets of size 5 that are generated by adding a solution to the subsets of size 4.

The results of running this experiment are summarized in Table 6.

Subset Generation Method	Average Deviation	Number of Best
SG1	0.0000%	20
SG2	0.0017%	19
SG3	0.0000%	20
SG4	0.0042%	18

Table 6. Summary of results for preliminary experiment 3

The results of this preliminary experiment indicate that there is no additional gain that could be realized by generating and combining subsets with more than 2 solutions. Hence, we perform step 2 (see Figure 1) of the scatter search implementation by limiting the subset generation to all pairs of reference solutions. These results are in line with similar experiments for other combinatorial optimization problems (Campos, et al. 2001).

4. Computational Experiments

This section describes the computational experiments that we performed to compare our proposed procedure to state-of-the-art methods for solving the maximum diversity problem. Our scatter search implementation follows the basic framework outlined in Figure 1. The diversification generation, combination and improvement methods are those corresponding to the Tabu Search Hybrid in Table 1. The improvement method is applied to all the combined solutions (as described in Section 3.2) with $b = 12$ and $q = 30$. Finally, the subset generation method is limited to generating subset of size 2. This procedure (labeled SS in subsequent tables) is compared to the following solution methods:

- KLD (Silva, Ochi and Martins, 2004) with local search (Ghosh, 1996)
- KLDv2 (Silva, Ochi and Martins, 2004) with local search (Ghosh, 1996)
- Tabu_D-2 with LS_TS (Duarte and Martí, 2006)

For this comparison, we use the same four data sets employed in Duarte and Martí (2006):

SOM: This data set consists of 20 matrices with random numbers between 0 and 9 generated from an integer uniform distribution. The problem sizes are such that for $n = 100$, $m = 10, 20, 30$ and 40 ; for $n = 200$, $m = 20, 40, 60$ and 80 ; for $n = 300$, $m = 30, 60, 90$ and 110 ; for $n = 400$, $m = 40, 80, 120$, and 160 ; and for $n = 500$, $m = 50, 100, 150$ and 200 . These instances were generated by Silva, Ochi and Martins (2004).

GKD: This data set consists of 20 matrices for which the values were calculated as the Euclidean distances from randomly generated points with coordinates in the 0 to 10 range. The number of coordinates for each point is also randomly generated between 2 and 21. Glover, Kuo and Dhir (1998) developed this data generator and constructed instances with $n = 30$. We generated instances with $n = 500$ and $m = 50$.

Type I: We generate 20 instances of Type I, as described in section 2, with $n = 2000$ and $m = 200$.

Type II: We generate 20 instances of Type II, as described in section 2, with $n = 500$ and $m = 50$.

In these experiments, we observed the solution quality obtained by each method after 30 seconds and after 3 minutes of search time. We also included a 30-minute run for Type I problems. All the experiments were conducted on a Pentium 4 computer at 3 GHz with 3 GB of RAM. We coded all the procedures in Java and executed them in the Java Runtime Environment 1.5. Tables 7 and 8 show the summary of our results.

Data Set	Time	KLD	KLDv2	Tabu_D-2	SS
SOM	30 sec.	1.056%	1.463%	0.138%	0.002%
	3 min.	0.178%	0.187%	0.095%	0.000%
GKD	30 sec.	0.000%	0.000%	0.000%	0.000%
	3 min.	0.000%	0.000%	0.000%	0.000%
Type II	30 sec.	0.857%	1.083%	0.245%	0.010%
	3 min.	0.525%	0.607%	0.203%	0.000%
Type I	30 sec.	9.807%	100.000%	0.453%	0.453%
	3 min.	9.807%	9.828%	0.331%	0.331%
	30 min	1.018%	0.923%	0.233%	0.000%

Table 7. Comparison of average percent deviation at several times during the search

Data Set	Time	KLD	KLDv2	Tabu_D-2	SS
SOM	30 sec.	6	6	4	17
	3 min.	7	9	6	20
GKD	30 sec.	20	19	20	20
	3 min.	20	20	20	20
Type II	30 sec.	1	0	0	15
	3 min.	1	0	0	20
Type I	30 sec.	0	0	0	0
	3 min.	0	0	0	0
	30 min.	0	0	0	20

Table 8. Comparison of number of best solutions at several times during the search

Tables 7 and 8 show the merit of the proposed procedure. Our scatter search implementation consistently produces the best solutions with percent deviations that in some cases are orders of magnitude smaller than those of the competing methods. The problem instances in the GKD set do not provide a way of differentiating the performance of the methods that we are comparing. They are either easy to solve and all the methods are capable of finding the optimal solutions in a very short period of time or the problems are difficult and all the methods are attracted to the same local optima. We speculate that the former is true. Figure 3 shows the typical search profile for the methods that we compared. This run corresponds to the SOM set with a time limit of 3 minutes.

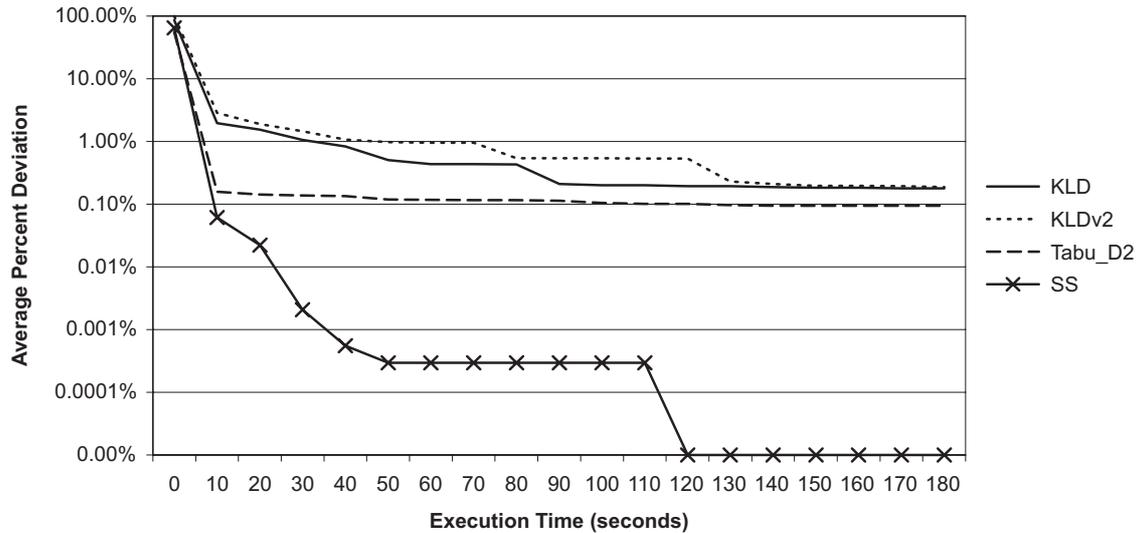


Figure 3. Search profile for a 3-minute run of the SOM set

We have applied a statistical test to the data used to generate Table 7. The results from the 30-second runs were not used because KLDv2 is not able to obtain solutions for Type I instances within the allotted time (see the 100% deviation in Table 7). We applied the Friedman test for paired samples to the best solutions obtained by each method. The resulting significance level of 0.000 clearly indicates that there are statistically significant differences among the four methods tested. Specifically, the rank values produced by this test are 1.98, 1.86, 2.82 and 3.36 for the KLD, KLDv2, Tabu_D-2 and SS, respectively. This indicates that among the procedures that we tested, SS is the best at obtaining solutions with large diverse scores, followed by Tabu_D-2, KLD and finally KLDv2.

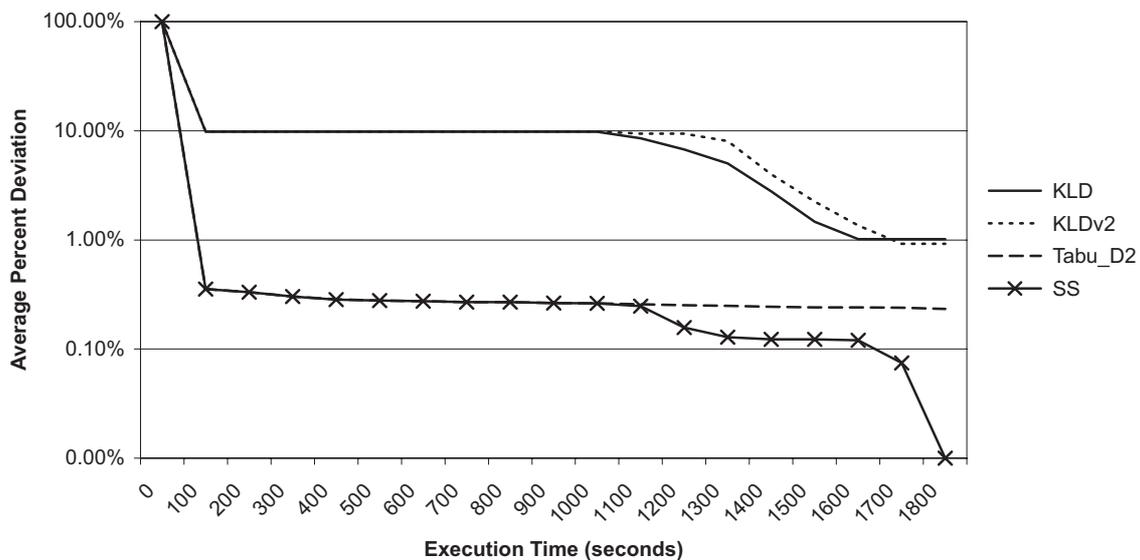


Figure 4. Search profile for a 30-minute run on Type I instances

It is interesting to point out that for the large Type I instances, there is no significant difference in performance between Tabu_*D*-2 and SS when the procedures are terminated after 3 minutes. This is due to the time that SS spends generating the initial population of solutions and, hence, within the 3-minute limit, it does not reach the phase where reference solutions are combined to generate others. Differences between the two methods are only detected after 20 minutes of search time, as shown in Figure 4.

5. Conclusions

We have described the development and implementation of a scatter search procedure for the solution of the maximum diversity problem. We arrived to our final design by way of performing a series of preliminary experiments. The final design is then compared to state-of-the-art methods and the outcome of our experiments seems quite conclusive in regard to the merit of the procedure that we propose. To the best of our knowledge, our work is the first one to test several hybridized procedures within the scatter search framework. We believe that the performance boost that we achieved by the use of simple memory mechanisms (some based on recency and some based on frequency information) within a scatter search design is a valuable lesson for future implementations.

Acknowledgments

This research has been partially supported by the *Ministerio de Educación y Ciencia* of Spain (Grant Refs. TIN2005-08943-C02-02, TIN2006-02696).

References

- Campos, V., F. Glover, M. Laguna and R. Martí (2001) "An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem," *Journal of Global Optimization*, vol. 21, pp. 397-414.
- Duarte, A. and R. Martí (2006) "Tabu Search and GRASP for the Maximum Diversity Problem," to appear in the *European Journal of Operational Research*.
- Ghosh, J. B. (1996) "Computational Aspects of the Maximum Diversity Problem," *Operations Research Letters*, vol. 19, pp. 175-181.
- Glover, F., M. Laguna (1997) *Tabu Search*. Kluwer Academic Publisher.
- Glover, F., C. C. Kuo, and K. S. Dhir (1998) "Heuristic Algorithms for the Maximum Diversity Problem," *Journal of Information and Optimization Sciences*, vol. 19, no. 1, pp. 109-132.
- Kuo, C. C., F. Glover and K. S. Dhir (1993) "Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming," *Decision Sciences*, vol. 24, no. 6, pp. 1171-1185.
- Laguna, M. and R. Martí (2003) *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers: Boston.
- Silva, G. C., L. S. Ochi and S. L. Martins (2004) "Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem," *Lecture Notes in Computer Science* 3059, Springer: Berlin Heidelberg, pp. 498-512.