Better and Faster Solutions for the Maximum Diversity Problem *

Roberto Aringhieri † Roberto Cordone ‡

April 2006

Abstract

The aim of the *Maximum Diversity Problem* (*MDP*) is to extract a subset M of given cardinality from a set of elements N, in such a way that the sum of the pairwise distances between the elements of M is maximum. This problem, introduced by Glover [7], has been deeply studied using GRASP methodologies [6, 1, 17, 2, 16]. Usually, effective algorithms owe their success more to the careful exploitation of problem-specific features than to the application of general-purpose methods. A solution for *MDP* has a very simple structure which can not be exploited for sophisticated neighborhood search. This paper explores the performance of three alternative solution approaches, that is Tabu Search, Variable Neighborhood Search and Scatter Search, comparing them with those of best GRASP algorithms in literature. We also focus our attention on the comparison of these three methods applied in their pure form.

Keywords: Maximum Diversity, Tabu Search, Scatter Search, Variable Neighborhood Search

1 Introduction

The *Maximum Diversity Problem* (*MDP*) consists in extracting from a set a maximally diversified subset of given cardinality. Let N be a set of n elements, and d_{ij} a diversity measure between pairs of elements $i \in N$ and $j \in N$ ($d_{ij} > 0$ when $i \neq j$, $d_{ij} = 0$ otherwise). The aim of the problem is to determine a subset $M \subset N$ of given cardinality m, such that the sum of the pairwise distances between the elements of M is maximum. A mathematical formulation for the *MDP* can be obtained by setting $x_i = 1$ if element $i \in N$ belongs to the solution M, $x_i = 0$ otherwise:

$$\max z = \frac{1}{2} \sum_{i \in N} \sum_{j \in N} d_{ij} x_i x_j \tag{1}$$

$$\sum_{i \in N} x_i = m \tag{2}$$

$$x_i \in \{0, 1\} \qquad i \in N \tag{3}$$

*The same version of this paper has been submitted the 14th of April 2006 to ESA conference.

†DTI, University of Milan, roberto.aringhieri@unimi.it
‡DTI, University of Milan, roberto.cordone@unimi.it

This model applies to several practical problems. A common requirement in the identification of work teams, student groups and juries is, for instance, to gather individuals with strongly diversified characteristics: work teams take advantage from including the largest possible range of skills, student groups should encourage the exchange between people with different backgrounds, juries should represent the widest variety of points of view existing in a community. The distance between individuals i and j with respect to the relevant characteristics can be modeled by a suitable function d_{ij}, and most of the time the number of individuals in the team is fixed. Other interesting applications concern the allocation of available resources for preserving biological diversity [8], VLSI design, scheduling final exams, medical treatment, and data mining [12].

The *MDP* is *strongly* $\mathcal{NP}$-hard [13] and it was introduced by Glover [7]. Apart from some mathematical programming approaches (limited to instances up to 40 elements) and from some early greedy and stingy heuristics [9, 18], the literature on the *MDP* consists of several Greedy Randomized Adaptive Search Procedures (*GRASP*) [5]. These algorithms [6, 1, 17, 2, 16] are characterized by a strong design effort focused on building good randomized starting solutions. The subsequent improvement phase is usually limited to a standard local search technique.

Usually, effective algorithms owe their success more to the careful exploitation of problem-specific features than to the application of general-purpose methods. In the case of *MDP*, this is very hard to obtain: a solution for *MDP* is a generic subset of m elements without any further requirements. Thus, it has a very simple structure which can not be exploited for sophisticated neighborhood search. For instance, a typical "trick" is to allow the search to visit promising unfeasible solutions led by a modified objective function; in the case of *MDP* there is no reasonable way to apply it. On the other side, this allows to compare the effectiveness of general-purpose methods applied in their pure form. In a previous paper [3], we have devised a Tabu Search algorithm which explores an opposite approach with respect to the literature, that is to refine the local search phase while keeping a very simple initialization procedure.

Here we extend this seminal work introducing further metaheuristics, namely Variable Neighborhood Search and Scatter Search. The three metaheuristics proposed are based on the same common elements (a greedy algorithm and a basic Tabu Search). They differ in the intensification and diversification mechanisms used to guide the search, respectively, toward promising regions and away from unpromising ones. We remark that these mechanisms are not problem-specific but peculiar of the method.

In Section 2 we introduce some notation and the common elements of the three algorithms proposed, which are presented in Section 3. Section 4 discusses their performance in comparison to each other and to the best algorithms reported in the literature. We also report the new best known results. Conclusions and future work close the paper.

2 Common elements

Given a solution $M \subset N$ and an element $i \in N$, let the contribution of i to M be defined as $D_i = \sum_{j \in M} d_{ij}$ (clearly, $z = \frac{1}{2} \sum_{i \in M} D_i$).

Greedy algorithm

A feasible solution can be built starting from a suitable pair of elements $M^{(0)} = \{i, j\} \subset N$ (obviously, $z^{(0)} = d_{ij}$) and iteratively adding new elements one by one. At the h-th step, choose the new element as $k^{(h)} = \arg\max_{i \in N \setminus M^{(h-1)}} D_i$ obtaining $M^{(h)} = M^{(h-1)} \cup \left\{k^{(h)}\right\}$ and $z^{(h)} = z^{(h-1)} + D_{k^{(h)}}$. After each step, D_i can be easily updated by adding the value $d_{ik^{(h)}}$. Each step requires $O(n)$ time, so that the overall procedure is $O(mn)$.

Basic Tabu Search

Starting from a given feasible solution M of value z, the basic Tabu Search procedure tries to improve it by iteratively exchanging a single element s in the current solution with a single element t out of it, that is $M' = M \cup \{t\} \setminus \{s\}$. All solutions in this neighborhood are evaluated and one of them becomes the current solution. It is possible to efficiently evaluate the effect of such a move on the objective function. The value z' of each solution M' is obtained by subtracting the total contribution of the old element s (that is D_s) and adding the total contribution of the new element t (that is $D_t - d_{st}$). More formally, $z' = z - D_s + D_t - d_{st}$.

This simple neighborhood search includes a mechanism to avoid looping over already visited solutions [10]. This mechanism consists in a finite-length list of forbidden moves, named tabu list. As we want to avoid both the inclusion of a recently removed element and the removal of a recently included element, we define two independent tabu lists: list L_{in} forbids an element to enter the solution for ℓ_{in} iterations, whilst list L_{out} forbids an element to exit for ℓ_{out} iterations.

The next solution is given by the move yielding the largest z', among the ones which are not tabu. A move improving the best known solution is always performed, even if it is tabu (*aspiration criterium*). After applying such a move, the values of D_i are updated as follows: $D_i = D_i - d_{is} + d_{it}$, $i \in N$. After I_{TS} iterations the algorithm stops.

3 Three metaheuristics for MDP

Using the elements introduced in Section 2, we describe the three metaheuristics proposed for the solution of the *MDP*, focusing on the intensification and diversification strategies proper to each approach.

eXploring Tabu Search

We extend the basic Tabu Search by adding memory mechanisms, both on a short and a long term. We refer to this algorithm as *XTS-MDP*. A more detailed description can be found in [3].

The short term memory mechanism allows to intensify the search decreasing the length of the tabu list after T_i consecutive improving iterations. On the contrary, the search is diversified increasing the tabu list length after T_w consecutive worsening iterations. The length ℓ_{in} of tabu list L_{in} starts from $\ell_{in}^{(0)} = \left(\ell_{in}^m + \ell_{in}^M\right)/2$ and varies in $\left[\ell_{in}^m, \ell_{in}^M\right]$ by steps equal to $\Delta\ell_{in}$. This is

a self-adapting parameter: it becomes larger when the length of the tabu list approaches the lower or the upper limit of its range. A similar behavior, ruled by similar parameters, holds for the length ℓ_{out} of list L_{out}.

The long term memory mechanism allows the search to escape from unpromising regions of the solution space. This mechanism is known as eXploring Tabu Search [4]. We maintain a list $\mathcal{M}$ of fixed length, composed of *second* solutions, that is the second best computed during each neighborhood exploration. The search restarts from the best solution in $\mathcal{M}$ every time any of the following conditions is verified: either the best known solution is not improved for I_{c_1} iterations or the length of one of the two tabu lists resides in the upper half of its range, that is $\left[\left(\ell^m + \ell^M\right)/2; \ell^M\right]$, for I_{c_2} consecutive iterations. The first condition suggests that the currently explored region is not promising, the second one that the short term mechanism is insufficient to diversify the search.

Variable Neighborhood Search

Variable Neighborhood Search (*VNS*) is a metaheuristic which systematically exploits the idea of neighborhood change, both in the descent to local minima and in the escape from valleys which contain them. Here we propose a simple *VNS* algorithm, named *VSN-MDP*, specifically developed to allow the basic Tabu Search to better explore the solution space. Due to the space limitations, our description traces the general scheme proposed in [11]; here we report only the specific adaptations necessary for *MDP*.

Let $N_k(M)$ be the k-th neighborhood of solution M, including all solutions M' obtained from M by replacing k elements in the solution with k elements out of it. The starting solution is generated by the greedy algorithm, choosing the first pair of elements i and j as the ones with the largest diversity d_{ij}. Each single iteration consists of at most $k_{\max}$ steps. At the k-th step, we obtain M'' from M by randomly generating a solution M' in $N_k(M)$ and improving it with the basic Tabu Search. Let ρ be the Hamming distance between M and M''. If $z'' + \alpha\rho(M, M'') > z$, that is, if M'' is better than M or it is sufficiently far away from M, then it replaces M as the current solution and the algorithm sets $k = 1$; otherwise M is unchanged and $k = k + 1$. The acceptation of worse (but distant) solutions is a known variant of *VNS* known as *skewed VNS* [11]. Parameter α is set equal to $(z_{\max} - z_{\min})/m$ in order to make the value of ρ comparable to that of z''; $z_{\max}$ and $z_{\min}$ are the best and the worst values of the solutions M'' generated from the current solution M. Therefore, as the solutions generated get worse, the search accepts them more and more easily, to move away from M. On the contrary, when a better solution is generated, $z_{\max} = z_{\min} = z(M)$ and classical *VNS* is adopted.

When k exceeds $k_{\max}$, a new iteration starts, and k gets back to 1. After I_{VNS} iterations the algorithm stops the search.

Scatter Search

Scatter Search (*SS*) is an evolutionary method which uses strategies for search intensification and diversification. The algorithm maintains a set of solutions R, named *Reference Set*, composed of two subsets B and D: B contains the best solutions computed during the search, whereas D contains solutions which largely differ from each other and from the best ones. The reference set evolves

by combining its members to obtain new solutions. The combination of two solutions in B intensifies the search, while the diversification is given by combining two solutions in D. We denote our Scatter Search algorithm as *SS-MDP*. Due to the space limitations, our description traces the general framework proposed in [15]; here we report only the specific adaptations necessary for *MDP*.

The initial generation of R has been done as follows. A starting solution is computed by the greedy algorithm, randomly selecting a pair of elements not yet used as starting point, and then improved by the basic Tabu Search. The algorithm tries to insert this solution first in B, then in D. If both insertions fail because the solution is neither good nor different enough from the reference set, it is randomly modified in order to improve the diversification, by exchanging the elements more frequently belonging to a solution with those which appear in a solution less frequently. Then, the algorithm tries to insert the resulting solution in R. In all insertion attempts, duplicate solutions are rejected.

The combination method merges two solutions M' and M'' drawn from R into a new one in the following way. First, we sum the contributions D'_i and D''_i which element $i \in N$ provides to solutions M' and M''. Clearly, the contribution is null if the element does not belong to the solution. Then, we rank the elements by decreasing values of the total contribution. Finally, the new solution is composed by selecting the first m elements and improved by basic Tabu Search.

After creating all possible combinations, the algorithms tries to insert them in R. If R is modified, the combination process is repeated using only pairs which the newly added solutions. Although it is possible to combine three or more solutions, the combination of pairs can be sufficient to obtain good results improving also the algorithm running time [14].

When R is no longer modified by the combination phase, the current iteration terminates, a new subset D is generated in the same way described before and a new iteration starts. The algorithm stops after I_{SS} iterations.

4 Computational results

In this section we report the computational results of the three algorithms previously described and compare them with those obtained by various GRASP [6, 1, 17, 2, 16]. Due to the limited amount of space, we will discuss in deeper detail the comparison between *XTS-MDP* and the best known results in the literature, since, among the three approaches proposed, Tabu Search appears to be the best compromise between solution quality and computational time. Then, we will comment on the different performance of *VNS-MDP* and *SS-MDP*. Before discussing the computational results, however, we introduce the computational environment, the benchmark instances and the tuning of algorithm parameters.

Setting up the computational experiments

Our algorithm is coded using the C standard 2 and runs on a Linux machine with G++ 3.3.6 compiler. The PC is an Intel Pentium 4 Mobile 2.8Ghz with 512MB of main memory.

For our experiments, we have used two sets of benchmark instances available in the literature: benchmark B_1, proposed in [1], consists of 40 instances with

n ranging from 50 to 250 and m from $0.2n$ to $0.4n$; benchmark B_2, proposed in [17], consists of 20 instances with n ranging from 100 to 500 and m from $0.1n$ to $0.4n$. These instances are also available at `http://www.dti.unimi.it/\homediraringhieri`.

Preliminary computational experiments have been done in order to tune the parameters of our three algorithms. For the basic Tabu Search, ℓ_{in} starts from 11 and varies in $[8; 14]$, ℓ_{out} starts from 5 and varies in $[3; 7]$. The length of the lists increases after $T_w = 5$ worsening moves, it decreases after $T_i = 3$ improving moves. The update steps $\Delta\ell_{in}$ and $\Delta\ell_{out}$ depend on the current length:

$$\Delta\ell_{in} = \begin{cases} 2 & \ell_{in} = \ell_{in}^m \text{ or } \ell = \ell_{in}^M \\ 1 & \ell_{in}^m < \ell_{in} < \ell_{out}^M \end{cases} \quad \text{and} \quad \Delta\ell_{out} = \begin{cases} 2 & \ell_{out} = \ell_{out}^m \text{ or } \ell = \ell_{out}^M \\ 1 & \ell_{out}^m < \ell_{out} < \ell_{out}^M \end{cases}.$$

As for *XTS-MDP*, the number of second solutions is $|\mathcal{M}| = 15$, $I_{c_1} = 300$ and $I_{c_2} = 25$. The total number of iterations is $I_{TS} = 2000$. For *VNS-MDP*, $k_{\max} = 9$, $I_{\text{VNS}} = 100$ and $I_{TS} = 300$. For *SS-MDP*, $|B| = |D| = 10$, $I_{\text{SS}} = 5$ and $I_{TS} = 40$.

The previously best known results for benchmarks B_1 and B_2 have been obtained by 15 different algorithms under distinct environment conditions: Ghosh's GRASP heuristic [6, 1], Andrade's GRASP heuristic [1], Silva's six GRASP heuristics (named from G3 to G8) [17], Andrade's six GRASP heuristics with path-relinking (named from T1E1 to T3E2) [2] and Hybrid GRASP with Data Mining (DM-GRASP) proposed in [16].

Results for benchmark B_1

It is not easy to establish a comparison on this benchmark, since the best known values have been obtained from all the competing algorithms except for DM-GRASP, but detailed values are available only for the six GRASP algorithms with path-relinking, as also reported in [2].

XTS-MDP equals the best result reported in the literature for 32 instances out of 40. Table 1 presents the remaining 8 instances. The four columns report, respectively, the name of the instances, the result of *XTS-MDP*, the best known in the literature, the difference between them. The best result is bolded: we improve it in 5 cases out of 8 and in 2 cases the difference is remarkable; our performance is worse in 3 cases, one of which is remarkable. We remind that this comparison opposes *XTS-MDP* to 14 variants of 4 different algorithms.

A one-to-one comparison is only possible with the six GRASP algorithms with path-relinking [2]: *XTS-MDP* provides the same result as *T3E2* (the best of the six) in 29 instances, it proves worse on instance B250m50 and better for the remaining 10 instances. Moreover, *XTS-MDP* is from 100 to 250 times faster (less than one minute against several minutes or hours). Of course, the machine employed (a 550 MHz Intel Pentium III PC with 384 MB of RAM) is slower, but this does not fully account for such a difference.

Algorithm *SS-MDP* reproduces all the results achieved by *XTS-MDP* (see Table 1) improving the one obtained on the hard instance B250m50 up to 7389784, which is better than the previous best known value. As a consequence, the performance of *SS-MDP* is better than the literature in 6 instances out of 8 (3 of which remarkably) and it is slightly worse in the remaining 2. The

Instance	*XTS-MDP*	Literature	Δ
A250m50	**12 654**	12 653	1
B200m80	17 544 447	**17 544 448**	-1
B250m50	7 379 797	**7 388 997**	-9 200
B250m100	**27 168 460**	27 162 906	5 554
C100m20	**1 207 522**	1 205 722	1 800
D150m60	**13 611 262**	13 611 261	1
D200m80	**24 133 321**	24 133 320	1
D250m100	37 753 118	**37 753 120**	-2

Table 1: Comparison between the best *XTS-MDP* results and the best results in the literature (when different) on benchmark B_1.

computational time ranges from 1 second to 1 hour; therefore, also *SS-MDP* results much faster than the GRASP algorithms in the literature.

Algorithm *VNS-MDP* equals the best known result in 30 instances out of 40, it provides better solutions in 3 cases and worse ones in the remaining 7. The computational times are comparable to the ones reported for the GRASP algorithms, that is rather long. When compared to *XTS-MDP*, *VNS-MDP* is always dominated, apart from the hard instance B250m50, in which it achieves the same result as *SS-MDP*.

The performance of the three algorithms presented on the hard instance B250m50 supports the need to introduce accurate techniques to intensify and diversify the search. The mechanism adopted by Tabu Search, in fact, proves less competitive with respect to the ones adopted by *VNS* and Scatter Search, which were both able to find the new best known result. Some insight can be gained by studying in deeper detail the behavior of *SS-MDP*: during the computation, the subset B including the best visited solutions is updated by inserting 45 new solutions, 29 of which obtained combining at least one solution drawn from subset D. Moreover, 8 of these solutions were still present in B at the end of the computation. We remind that B consists of 10 solutions. This proves that subset D plays a relevant role in the performance of the algorithm.

Finally, Table 2 reports the previous best known results in literature, the result of our three algorithms and the new best known values for B_1. For each instance, we bold the best results unless they are all equal.

Results for benchmark B_2

The results on benchmark B_2 can be compared in a more complete way. All the best known results have been obtained by the DM-GRASP in [16] except for instance "n500m150" whose best is computed by G5 algorithm in [17], contrary to what stated in [16]. Both the values and the computational times are available.

For 17 instances out of 20 *XTS-MDP* equals the best result reported in the literature. For instances "n400m160" and "n500m50", it increases the best known results by 4 and 10 respectively. For the instance "n500m150", *XTS-MDP* computes the same result as DM-GRASP which is 56572 whilst G5 computes 58605.

Instance	Literature [6, 1, 17, 2]	Proposed algorithms *XTS-MDP*	*SS-MDP*	*VNS-MDP*	New best known
a050m10	491.9	491.9	491.9	491.9	491.9
a050m20	1931.5	1931.5	1931.5	1931.5	1931.5
a100m20	2007.1	2007.1	2007.1	2007.1	2007.1
a100m40	7730.0	7730.0	7730.0	7730.0	7730.0
a150m30	4552.1	4552.1	4552.1	4552.1	4552.1
a150m60	17482.4	17482.4	17482.4	17482.4	17482.4
a200m40	8132.1	8132.1	8132.1	8132.1	8132.1
a200m80	31048.6	31048.6	31048.6	31048.6	31048.6
a250m50	12653.0	**12654.0**	**12654.0**	12653.0	12654.0
a250m100	48384.3	48384.3	48384.3	48384.3	48384.3
b050m10	334976	334976	334976	334976	334976
b050m20	1171416	1171416	1171416	1171416	1171416
b100m20	1267277	1267277	1267277	1267277	1267277
b100m40	4544642	4544642	4544642	4544642	4544642
b150m30	2758381	2758381	2758381	2758381	2758381
b150m60	9960461	9960461	9960461	9960461	9960461
b200m40	4788086	4788086	4788086	4788086	4788086
b200m80	**17544448**	17544447	17544447	17544447	17544448
b250m50	7388997	7379797	**7389784**	**7389784**	7389784
b250m100	27162906	**27168460**	**27168460**	**27168460**	27168460
c050m10	316409	316409	316409	316409	316409
c050m20	1094343	1094343	1094343	1094343	1094343
c100m20	1205722	**1207522**	**1207522**	1205722	1207522
c100m40	4219476	4219476	4219476	4219476	4219476
c150m30	2613286	2613286	2613286	2613286	2613286
c150m60	9374611	9374611	9374611	9374611	9374611
c200m40	4630545	4630545	4630545	4630545	4630545
c200m80	16759895	16759895	16759895	16759895	16759895
c250m50	7178043	7178043	7178043	7178043	7178043
c250m100	26047022	26047022	26047022	26047022	26047022
d050m10	381379	381379	381379	381379	381379
d050m20	1502908	1502908	1502908	1502908	1502908
d100m20	1570800	1570800	1570800	1570800	1570800
d100m40	6067776	6067776	6067776	6067776	6067776
d150m30	3502567	3502567	3502567	3502567	3502567
d150m60	13611261	**13611262**	**13611262**	**13611262**	13611262
d200m40	6207580	6207580	6207580	6207580	6207580
d200m80	24133320	**24133321**	**24133321**	**24133321**	24133321
d250m50	9685430	9685430	9685430	9685430	9685430
d250m100	**37753120**	37753118	37753118	37753118	37753120

Table 2: New best known results for B_1

The computational times of *XTS-MDP* ranges from 0.1 to 627 seconds while DM-GRASP ranges from 308 to 557432 seconds. Note that the computational times reported in [16] refer to the average of 10 runs, thus we have multiplied the times by 10 before comparing. Our computational times also refer to the whole run and not to the time to compute the best solution. For the instance "n500m150", *XTS-MDP*, DM-GRASP and G5 require respectively 446, 177421 and 87153 seconds. The computational times show a huge difference. Once again, the different machine employed (an AMD Athlon 1.3 GHz with 256 MB for G5 and PIV 1.7 GHz with 256 MB for DM-GRASP) cannot explain the whole difference.

Algorithm *SS-MDP* achieves the same results as *XTS-MDP* in a much longer computational time (from 3 to 26000 seconds), which is anyway much lower than the time required by DM-GRASP and G5.

	VNS-MDP		DM-GRASP		
Instance	z	CPU	z	CPU	Δ
n200m40	4448	5032.75	**4450**	1305.0	-2
n200m60	9434	3996.29	**9437**	3320.0	-3
n300m30	2691	3854.05	**2694**	2121.0	-3
n300m60	9688	22797.62	**9689**	8594.0	-1
n300m120	35879	34307.00	**35881**	30620.0	-2
n400m120	36315	58620.00	**36317**	70064.0	-2
n400m160	**62487**	55080.00	62483	100662.0	4
n500m50	**7141**	23960.00	7131	19408.0	10
n500m100	26254	34500.78	**26258**	83435.0	-4
n500m200	97330	44941.91	**97344**	263660.0	-14

Table 3: Comparison between *VNS-MDP* and DM-GRASP (when different) on benchmark B_2 (time in seconds).

Algorithm *VNS-MDP* equals the best known results in the literature for 9 instances. Table 3 discusses the remaining 11 instances. The six columns report, respectively, the name of the instance, the result and computational time for *VNS-MDP*, the result and computational time for DM-GRASP, the difference between our result and the best known one. The best result for each instance is bolded. The computational times are all expressed in seconds, and the ones reported in [16] have been multiplied by 10 because they referred to the average of 10 runs, while the results refer to the best over 10 runs. *VNS-MDP* yields better results in 2 cases and worse results in 9 cases; in all cases the differences are not remarkable. Moreover, the computational times are similar.

We observe that *XTS-MDP* and *SS-MDP* dominate the best competitor algorithm which is DM-GRASP. Moreover, although *VNS-MDP* is the worst performing algorithm among the three proposed, its results and its computational time are comparable to DM-GRASP.

Finally, Table 4 reports the previous best known results in literature, the result of our three algorithms and the new best known values for B_2. For each instance, we bold the best results unless they are all equal.

Instance	Literature [6, 1, 17, 2, 16]	Proposed algorithms *XTS-MDP*	*SS-MDP*	*VNS-MDP*	New best known
n100m10	333	333	333	333	333
n100m20	1195	1195	1195	1195	1195
n100m30	2457	2457	2457	2457	2457
n100m40	4142	4142	4142	4142	4142
n200m20	1247	1247	1247	1247	1247
n200m40	**4450**	**4450**	**4450**	4448	4450
n200m60	**9437**	**9437**	**9437**	9434	9437
n200m80	16225	16225	16225	16225	16225
n300m30	**2694**	**2694**	**2694**	2691	2694
n300m60	**9689**	**9689**	**9689**	9688	9689
n300m90	20743	20743	20743	20743	20743
n300m120	**35881**	**35881**	**35881**	35879	35881
n400m40	4658	4658	4658	4658	4658
n400m80	16956	16956	16956	16956	16956
n400m120	**36317**	**36317**	**36317**	36315	36317
n400m160	62483	**62487**	**62487**	**62487**	62487
n500m50	7131	**7141**	**7141**	**7141**	7141
n500m100	**26258**	**26258**	**26258**	26254	26258
n500m150	**58605**	56572	56572	56568	58605
n500m200	**97344**	**97344**	**97344**	97330	97344

Table 4: New best known results for B_2

5 Conclusions

In this paper, we have presented three algorithms for the MDP, a problem with applications in a wide range of different fields. All previously proposed algorithms of some effectiveness are GRASP procedures [6, 1, 17, 2, 16].

Extending our seminal work [3], we have focused our attention on the comparison of general-purpose metaheuristics exploiting the fact that *MDP* has no specific features to take advantage of.

Algorithm *XTS-MDP* includes a short term memory mechanism tuning the length of the tabu lists and a long term memory mechanism which, under suitable conditions, restarts the search from a set of promising solutions previously taken into account but not yet visited. Algorithm *VNS-MDP* generates new starting solutions in a progressively enlarging neighborhood. Algorithm *SS-MDP* manages a pool of solutions, selected for their quality or their reciprocal diversity, and combines them to generate new starting solutions.

Regarding to the solution quality, *SS-MDP* is the best performing algorithm, closely followed by *XTS-MDP* (their results differ only on one instance). *VNS-MDP* is the worse of the three. As for the running time, *XTS-MDP* is by far the fastest of the three algorithms. *SS-MDP* and *VNS-MDP* are slower than *XTS-MDP*: however, the former is still faster than competitors whilst the latter takes approximately the same time.

In conclusion, *XTS-MDP* appears to be the best compromise between solution quality and performance. We observe also that the performances of our

worst algorithm (*VNS-MDP*) are comparable with those of best algorithms previously proposed in literature.

Ongoing works concern the improvement of *VNS-MDP* and the implementation of a new algorithm based on the *Iterated Local Search* framework.

Acknowledgments

The authors wish to thank Yari Melzani, Gian Paolo Ghilardi, Alberto Ghilardi, Andrea Beretta and Marco Tadini for their help in performing the computational experiments.

References

[1] P. M. D. Andrade, A. Plastino, L. S. Ochi, and S. L. Martins. GRASP for the Maximum Diversity Problem. In *Proceedings of the Fifth Metaheuristics International Conference (MIC 2003)*, 2003.

[2] P. M. D. Andrade, L. S. Plastino, and S. L. Martins. GRASP with path-relinking for the maximum diversity problem. In S. Nikoletseas, editor, *Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA 2005)*, volume 3539 of *Lecture Notes in Computer Science*, pages 558–569. Springer Berlin / Heidelberg, 2005.

[3] R. Aringhieri, R. Cordone, and Y. Melzani. Tabu search vs. GRASP for the Maximum Diversity Problem. Note del Polo 89, Università degli Studi di Milano, Crema, December 2005. [submitted for publication to *4OR*].

[4] M. Dell'Amico and M. Trubian. Solution of large weighted equicut problems. *European Jurnal of Operational Research*, 106:500–521, 1998.

[5] P. Festa and M. G. C. Resende. GRASP: An annotated bibliography. In C. C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[6] J. B. Ghosh. Computational aspects of maximum diversity problem. *Operation Research Letters*, 19:175–181, 1996.

[7] F. Glover, G. Hersh, and C. McMillian. Selecting subset of maximum diversity. MS/IS 77-9, University of Colorado at Boulder, 1977.

[8] F. Glover, C. C. Kuo, and K. S. Dhir. A discrete optimization model for preserving biological diversity. *Appl. Math. Modelling*, 19(11):696–701, November 1995.

[9] F. Glover, C. C. Kuo, and K. S. Dhir. Integer programming and heuristic approaches to the minimum diversity problem. *Journal of Business and Management*, 4(1):93–111, 1996.

[10] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[11] P. Hansen and N. Mladenovic. Variable neighborhood search. In F. Glover and G. Kochenagen, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publishers, 2003.

[12] G. Kochenberger and F. Glover. Diversity data mining. Working Paper Series HCES-03-99, The University of Mississipi, 1999.

[13] C. C. Kuo, F. Glover, and K.S. Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Science*, 24:1171–1185, 1993.

[14] M. Laguna and V.A. Armentano. Lessons from applying and experimenting with scatter search. In C. Rego and B. Alidaee, editors, *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, chapter 10, pages 229 – 246. Kluwer Academic Publishers, 2005.

[15] M. Laguna and R. Martí. *Scatter Search: methodology and Implementations in C.* Kluwer Academic Publishers, 2003.

[16] L.F. Santos, M.H. Ribeiro, A. Plastino, and S.L. Martins. A Hybrid GRASP with Data Mining for the Maximum Diversity Problem. In Andrea Roli Michael Sampels Mara J. Blesa, Christian Blum, editor, *Hybrid Metaheuristics, Second International Workshop*, volume 3636 of *Lecture Notes in Computer Science*, pages 116–127. Springer Berlin / Heidelberg, 2005.

[17] G. C. Silva, L. S. Ochi, and S. L. Martins. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. In *Proceedings of the 3rd International Workshop on Efficient and Experimental Algorithms (WEA 2004)*, volume 3059 of *Lectures Notes on Computer Science*, pages 498–512. Springer Berlin / Heidelberg, 2004.

[18] R. Weitz and S. Lakshminarayanan. An empirical comparison of heuristic methods for creating maximally diverse group. *Journal of the Operational Research Society*, 49:635–646, 1998.