Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem

Conference Paper *in* Lecture Notes in Computer Science · May 2004

Impact Factor: 0.51 · DOI: 10.1007/978-3-540-24838-5_37 · Source: DBLP

3 authors, including:

Geiza Cristina da Silva
Federal University of Pernambuco
7 PUBLICATIONS **75** CITATIONS

SEE PROFILE

Simone Martins
Universidade Federal Fluminense
41 PUBLICATIONS **562** CITATIONS

SEE PROFILE

Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem

Geiza C. Silva, Luiz S. Ochi, and Simone L. Martins

Universidade Federal Fluminense, Departamento de Ciência da Computação
Rua Passo da Pátria, 156 - Bloco E - 3 andar - Boa Viagem
24210-240, Niterói, RJ, Brazil
{gsilva, satoru, simone}@ic.uff.br

Abstract. The maximum diversity problem (MDP) consists of identifying optimally diverse subsets of elements from some larger collection. The selection of elements is based on the diversity of their characteristics, calculated by a function applied on their attributes. This problem belongs to the class of NP-hard problems. This paper presents new GRASP heuristics for this problem, using different construction and local search procedures. Computational experiments and performance comparisons between GRASP heuristics from literature and the proposed heuristics are provided and the results are analyzed. The tests show that the new GRASP heuristics are quite robust and find good solutions to this problem.

1 Introduction

The maximum diversity problem (MDP) [5–7] consists of identifying optimally diverse subsets of elements from some larger collection. The selection of elements is based on the diversity of their characteristics, calculated by a function applied on their attributes. The goal is to find the subset that presents the maximum possible diversity. There are many applications [10] that can be solved using the resolution of this problem, such as medical treatment, selecting jury panel, scheduling final exams, and VLSI design. This problem belongs to the class of NP-hard problems [6].

Glover et al. [6] presented mixed integer zero-one formulation for this problem, that can be solved for small instances by exact methods. Bhadury et al. [3] developed an exact algorithm using a network flow approach for the diversity problem of working groups for a graduate course.

Some heuristics are available to obtain approximate solutions. Weitz and Lakshminarayanan [12] developed five heuristics to find groups of students with the most possible diverse characteristics, such as nationality, age and graduation level. They tested the heuristics using instances based on real data and implemented an exact algorithm for solving them and the heuristic LCW (Lofti-Cerveny-Weitz method) was considered the best for solving these instances.

Constructive and destructive heuristics were presented by Glover et al. [7], who created instances with different size of population (maximum value was 30) and showed that the proposed heuristics obtained results close (2 %) to the ones obtained by the exact algorithm, but much faster.

Kochenberger and Glover [10] showed results obtained using a tabu search and Katayama and Naribisa [9] developed a memetic algorithm. Both report that computational experiments were carried out, but they did not compare the performance of their algorithms with exact or other heuristics procedures.

Ghosh [5] proposed a GRASP (Greedy Randomized Adaptive Search Procedure) that obtained good results for small instances of the problem. Andrade et al. [2] developed a new GRASP and showed results for instances randomly created with a maximum population of 250 individuals. This algorithm was able to find some solutions better than the ones found by the Ghosh algorithm.

GRASP [4] is an iterative process, where each iteration consists of two phases: construction and local search. In the construction phase a feasible solution is built, and its neighborhood is explored by a local search. The result is the best solution found over all iterations. In Section 2 we describe three construction procedures developed using the concept of reactive GRASP introduced by Prais and Ribeiro [11], and two local search strategies. In Section 3 we show computational results for different versions of GRASP heuristics created by the combination of a constructive algorithm and a local search strategy described in Section 2. Concluding remarks are presented in Section 4.

2 GRASP heuristics

The construction phase of GRASP is an iterative process where, at each iteration, the elements $c \in C$ that do not belong to the solution are evaluated by a greedy function $g : C \rightarrow \Re_+$, that estimates the gain of including it in the partial solution. They are ordered by their estimated value in a list called restricted candidate list (RCL) and one of them is randomly chosen and included in the solution. The size of the RCL is limited by a parameter α. For a maximization problem, only the elements whose g values are in the range $[(1-\alpha)g_{max}, g_{max}]$ are placed in RCL. This process stops when a feasible solution is obtained.

Prais and Ribeiro [11] proposed a new procedure called Reactive GRASP, for which the parameter α used in the construction phase is self adjusted for each iteration. For the first construction iteration, an α value is randomly selected from a discrete set $A = \{\alpha_1, \ldots, \alpha_m\}$. Each element α_i has a probability p_i associated and, initially, a uniform distribution is applied, thus we have $p_i = 1/m, i = 1, \ldots, m$. Periodically the probability distribution $p_i, i = 1, \ldots, m$ is updated using information collected during the former iterations. The aim is to associate higher probabilities to values of α that lead to better solutions and lower ones to values of α that guide to worse solutions.

The solutions generated by the construction phase are not guaranteed to be locally optimal. Usually a local search is performed to attempt to improve each constructed solution. It works by successively replacing the current solution by a

better one from its neighborhood, until no more better solutions are found. Normally, this phase demands great computational effort and execution time, so the construction phase plays an important role to diminish this effort by supplying good starting solutions for the local search. We implemented a technique widely used to accomplish this task, that leads to a more greedy construction. For each GRASP iteration, the construction algorithm is executed X times generating X solutions and only the best solution is selected to be used as the initial solution for the local search phase.

In the next subsections, we describe the construction and local search algorithms developed for the GRASP heuristics, using the concepts discussed in this section.

2.1 Construction phase.

Let $E = \{e_i : i \in N\}, N = \{1, 2, \ldots, n\}$ be a population of n elements and e_{il}, $l \in L = \{1, 2, ..., l\}$ the l values of the attributes of each element. In this paper, we measure the diversity between any two elements i and j by the Euclidean distance calculated as $d_{ij} = \sqrt{\sum_{k=1}^{l}(e_{ik} - e_{jk})^2}$. Let M be a subset of N and the overall diversity be $z(M) = \sum_{i<j:i,j\in M} d_{ij}$. The MDP problem consists of maximizing the cost function $z(M)$, subject to $|M| = m$.

We describe three construction algorithms developed to be used in GRASP heuristics where all of them use the techniques described before: Reactive GRASP and filtering of constructed solutions.

K larger distances heuristic (KLD). This algorithm constructs an initial solution by randomly selecting an element from a RCL of size K at each construction iteration. The RCL is created by selecting for each element $i \in N$, the K elements $j \in N\backslash\{i\}$, that exhibit larger values of d_{ij} and sum these K values of d_{ij}, obtaining s_i. Then, we create a list of all elements i sorted in descending order by their s_i values and select the K first elements to compose the RCL list.

The procedure developed to implement the reactive GRASP starts considering m_it to be the total number of GRASP iterations. In the first block of iterations $B_1 = 0.4m\_it$, we evaluate four different values for $K \in \{K_1, K_2, K_3, K_4\}$ and the evaluation is done by dividing the block into four equal intervals $c_i, i = 1, \ldots, 4$. We use the value K_i for all iterations belonging to interval $c_j, i = j$. The values of K_i are shown in Tab. 1, where $\mu = (n - m)/2$. After the execution of

Table 1. K values for block B_1

i	c_i	K
1	$[1, \ldots, 0.1m\_it]$	$m + \mu - 0.2\mu$
2	$(0.1m\_it, \ldots, 0.2m\_it]$	$m + \mu - 0.1\mu$
3	$(0.2m\_it, \ldots, 0.3m\_it]$	$m + \mu + 0.1\mu$
4	$(0.3m\_it, \ldots, 0.4m\_it]$	$m + \mu + 0.2\mu$

the last iteration of block B_1, we evaluate the quality of the solutions obtained for each K_i. We calculate the mean diversity value $zm_i = \sum_{1 \leq q \leq 0.1m\_it} z(sol_{iq})$ for the solutions $sol_{iq}, i = 1, \ldots 4; q = 1, \ldots, 0.1m\_it$ obtained using each K_i. The values K_i are stored in a list LK ordered by their zm_i values.

Then for the next block of iterations $B_2 = 0.6m\_it$, we divide it into four intervals y_i, each one with different number of iterations, and use the K_i values as shown in Tab. 2. In this way, the values K_i that provide better solutions are used in a larger number of iterations.

Table 2. K values for block B_2

i	y_i	K
1	$[0.4m\_it, \ldots, 0.64m\_it]$	lk_1
2	$(0.64m\_it, \ldots, 0.82m\_it]$	lk_2
3	$(0.82m\_it, \ldots, 0.94m\_it]$	lk_3
4	$(0.94m\_it, \ldots, m\_it]$	lk_4

At each GRASP iteration, we apply the filter technique for this heuristic by constructing 400 solutions and only the best solution is sent to the local search procedure.

The pseudo-code, including the description of the procedure for the construction phase using K larger Distances heuristic, is given in Fig. 1.

```
procedure constr_KLD(it_GRASP, m_it, numsol, n, m)
 1.   best_cost_sol ← 0;
 2.   K ← det_K(it_GRASP, m_it, LK, i);
 3.   num_sol[i] ← num_sol[i] + 1;
 4.   RCL ← Build_RCL(K);
 5.   for j = 1, ..., max_sol_filter do
 6.     sol ← {};
 7.     for k = 1, ..., m do
 8.       Randomly select an individual e* from RCL;
 9.       sol ← sol ∪ {e*};
10.       RCL ← RCL − {e*};
11.     end for;
12.     if (z(sol) > best_cost_sol) then do
13.       sol_constr ← sol;
14.       best_cost_sol ← z(sol);
15.     end if
16.   end for;
17.   sol_eval[i, num_sol[i]] ← z(sol_constr);
18.   if (it_GRASP == 0.4m_it) then do
19.     LK ← Build_LK(sol_eval);
20.   end if;
21.   return sol_constr.
```

Fig. 1. Construction procedure used to implement the KLD heuristic

In line 1, we initialize the cost of the best solution found in the execution of max_sol_filter iterations. The value K to be used to build the Restricted Candidate List (RCL) is calculated by the procedure det_K in line 2. This procedure

defines the value for K implementing the reactive GRASP described before. In line 3, the number of solutions found for a specific K is updated and, in line 4, the RCL is built. From line 5 to line 16, the construction procedure is executed max_sol_filter times and only the best solution is returned to be used as an initial solution by the local search procedure. From line 7 to line 11, a solution is constructed by the random selection of an element from RCL. In lines 12 to 15, we update the best solution found by the construction procedure and the cost of the solution found using the selected K is stored in line 17. When the first block B_1 of iterations ends, the values K_i are evaluated and put in the list LK sorted in descending order, in line 19.

K larger distances heuristic-v2 (KLD-v2). This algorithm is similar to the previously described algorithm, the difference between them is the way that the Restricted Candidate List is built. In the former algorithm, the RCL is computed before the execution of the construction iterations and, for each iteration, the only modification made in the RCL is the removal of the element that is inserted in the solution.

In this algorithm, the RCL is built using an adaptive procedure, where the process to select the first element of the constructed solution is the same as of the KLD heuristic, which means that an element is randomly selected from the RCL built as described in line 4 of Fig. 1.

Let M_c be a partial solution with $c, 1 \leq c < m$ elements and $i \in N \backslash M_c$ a candidate to be inserted in the next partial solution M_{c+1}. For each i, we select the $(K - c - 1)$ elements $j \in N \backslash (M_c \bigcup \{i\})$,that present larger values of d_{ij} and calculate the sum of the $(K - c - 1)$ values of d_{ij} obtaining s_i. To select the next element to be inserted, an initial candidate list is created based on the greedy function $gf(i)$ shown in (1), where the first term corresponds to the sum of distances from the candidate i to the elements $j \in M_c$, and the second term stands for the sum of distances from element i to the $(K - c - 1)$ elements that are not in the solution M_c and present larger distances to i. The initial candidate list is formed by the elements i, sorted in descending order with respect to $gf(i)$, and the first K elements are selected from this list to build the RCL.

$$gf(i) = \sum_{j \in M_c} d_{ij} + s_i \tag{1}$$

The Reactive GRASP and the construction filter are implemented in the same way as in KLD. Once this construction algorithm demands much more execution time than KLD algorithm, only 2 solutions, instead of 400, are generated to be filtered.

Most distant insertion heuristic (MDI). Let M_c be a partial solution with $c, (1 \leq c < m)$ elements, the partial solution M_1 is obtained by randomly selecting an element from all elements $i \in N$.

The second element m_2 is the element j, which presents the larger distance $d_{ij}, i \in M_1, j \in N \backslash M_1$. To obtain $M_c (c \geq 3)$ from M_{c-1}, the element to be

inserted in the solution is randomly selected from a RCL. The RCL is built based on the function $dsum(j)$ showed in (2), where the first term of this function corresponds to the sum of distances between all elements $i \in M_{c-1}$. The second term is the sum between all elements $i \in M_{c-1}$ to a candidate j that is not in the partial solution M_{c-1}.

$$dsum(j) = \sum_{1 \leq y \leq c-2} \sum_{y+1 \leq w \leq c-1} d_{yw} + \sum_{1 \leq v \leq c-1} d_{vj} \tag{2}$$

An initial candidate list (ICL) is created containing the elements $j \in N \backslash M_{c-1}$, sorted in descending order by their $dsum(j)$ values. The first $\alpha \times n$ elements of ICL are selected to form the RCL.

For this algorithm, the reactive GRASP is implemented in the same way done for the K larger distances heuristic. The first block $B_1 = 0.4m\_it$ is divided into four intervals of the same size and four values for $\alpha \in \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ are evaluated. Table 3 shows the values of α used for each interval. The values $\alpha_i, i = 1, \ldots, 4$ are evaluated by calculating the mean diversity value $zm_i = \sum_{1 \leq q \leq 0.1m\_it} z(sol_{iq})$ for the solutions $sol_{iq}, i = 1, \ldots 4; q = 1, \ldots, 0.1m\_it$ obtained using each α_i. The values α_i are stored in a list $L\alpha$ ordered by their zm_i values.

Table 3. α values for block B_1

i	c_i	α
1	$[1, \ldots, 0.1m\_it]$	0.03
2	$(0.1m\_it, \ldots, 0.2m\_it]$	0.05
3	$(0.2m\_it, \ldots, 0.3m\_it]$	0.07
4	$(0.3m\_it, \ldots, 0.4m\_it]$	0.1

The next block of iterations $B_2 = 0.6m\_it$ is also divided into four intervals y_i, each one with distinct number of iterations and, for each one, a value of α is associated, as shown in Tab. 4.

Table 4. α values for block B_2

i	y_i	α
1	$[0.4m\_it, \ldots, 0.64m\_it]$	$l\alpha_1$
2	$(0.64m\_it, \ldots, 0.82m\_it]$	$l\alpha_2$
3	$(0.82m\_it, \ldots, 0.94m\_it]$	$l\alpha_3$
4	$(0.94m\_it, \ldots, m\_it]$	$l\alpha_4$

We have also implemented the same procedure described above for filtering the constructed solutions. In this case, the number of solutions generated is n, so it depends on the population size of each instance.

Figure 2 shows the construction phase procedure using the MDI heuristic. In line 1, we initialize the value of the best solution found. The value α to

```
procedure constr_MDI(it_GRASP, m_it, numsol, n, m)
 1.   best_cost_sol ← 0;
 2.   α ← det_α(it_GRASP, m_it, Lα, i);
 3.   num_sol [i] ← num_sol [i] + 1;
 4.   N_RCL ← N;
 5.   for j = 1, . . . , max_sol_filter do
 6.     sol ← {};
 7.     Randomly select an individual m1 from N;sol ← sol ∪ {m1};
 8.     for all j ∈ N\M1 do
 9.       Compute d_{m1 j}
10.       m2 ← l, |d_{m1 l} = max(d_{m1 j}), j ∈ N\M1;
11.       sol ← sol ∪ {m2};
12.     end for all;
13.     N_RCL ← N − M2;
14.     for k = 3, . . . , m do
15.       RCL ← Build_RCL_α(N_RCL, α);
16.       Randomly select an individual e* from RCL;
17.       sol ← sol ∪ {e*};
18.       N_RCL ← N_RCL − {e*};
19.     end for;
20.     if (z(sol) > best_cost_sol) then do
21.       sol_constr ← sol;
22.       best_cost_sol ← z(sol);
23.     end if
24.   end for;
25.   sol_eval [i, num_sol[ i ]] ← z(sol_constr);
26.   if (it_GRASP == 0.4m_it) then do
27.     Lα ← Build_Lα(sol_eval);
28.   end if;
29.   return sol_constr;
```

Fig. 2. Construction procedure used to implement the MDI heuristic

be used to build the RCL is calculated by the procedure det_α in line 2. This procedure selects α based on the reactive GRASP discussed before. In line 3, the number of solutions found for a specific α is updated and in line 4, the set that contains the candidates to be inserted in the solution is initialized to contain all elements belonging to N. From line 5 to line 24, the construction procedure is executed max_sol_filter times and only the best solution is returned to be used as an initial solution by the local search procedure. In line 7, the first element is selected and from line 8 to line 12, we determine the second element of the solution. From line 14 to line 19, the insertion of the other elements is performed. For each iteration, in line 15, a RCL is built and, in line 16, an element is randomly selected from it. In line 18, we update the candidates to be inserted in the next iteration. In lines 20 to 23, we update the best solution found by the construction procedure. The cost of the best solution found using the selected α is stored in line 25. When the first block B_1 of iterations finishes, the values α_i are evaluated and put in the list $L\alpha$ in line 27.

2.2 Local Search Phase

After a solution is constructed, a local search phase should be executed to attempt to improve the initial solution. In this paper, we use two different local search algorithms. The first one was developed by Ghosh [5] and the second one by us using the Variable Neighborhood Search (VNS) [8] heuristic.

Ghosh Algorithm (GhA) The neighborhood of a solution defined by Ghosh [5] is the set of all solutions obtained by replacing an element in the solution by other that does not belong to the set associated with the solution. The incumbent solution M is initialized with the solution obtained by the construction phase. For each $i \in M$ and $j \in N \backslash M$, the improvement due to exchanging i by j, $\Delta z(i,j) = \sum_{u \in M \backslash \{i\}} (d_{ju} - d_{iu})$ is computed. If for all i and j, $\Delta z(i,j) < 0$, the local search is terminated, as no exchange will lead to a better solution. Otherwise, the elements of the pair (i,j) that provides the maximum $\Delta z(i,j)$ are interchanged creating a new incumbent solution M and the local search is performed again.

SOM Algorithm (SOMA) We have also implemented a local search using a VNS heuristic. In this case, we use the GhA algorithm until there is no more improvement in the solution. After that, we execute a local search based on a new neighborhood, which is defined as the set of all solutions obtained by replacing two elements in the solution by another two that are not in the solution. The incumbent solution M is initialized with the solution obtained by the first phase of the local search. For each $(i,j) \in M$ and $(v,w) \in N \backslash M$, the improvement due to exchanging (i,j) by (v,w), $\Delta z((i,j),(v,w)) = \sum_{u \in M \backslash \{i,j\}} (d_{vu} + d_{wu} - d_{iu} - d_{ju})$ is computed. If for all pairs (i,j) and (v,w), $\Delta z((i,j),(v,w)) < 0$, as no exchange will improve the solution, the local search is terminated. Otherwise, the pairs (i,j) and (v,w) that provides the maximum $\Delta z((i,j),(v,w))$ are interchanged, a new incumbent solution M is created and the local search is performed again.

We developed several GRASP heuristics combining the construction procedures with the local search strategies described above and the computational experiments implemented to evaluate the performance of these heuristics are presented in next section.

3 Computational Results

We tested nine GRASP procedures that are shown in Tab. 5.

Table 5. GRASP procedures

GRASP procedure	Construction heuristic	Local search heuristic
G1	Ghosh	GhA
G2	Ghosh	SOMA
G3	MDI	GhA
G4	MDI	SOMA
G5	KLD	GhA
G6	KLD	SOMA
G7	KLD-v2	GhA
G8	KLD-v2	SOMA
G9	Andrade	Andrade

The first GRASP procedure G1 is an implementation of the GRASP heuristic developed by Ghosh and the second one is a procedure that implements Ghosh construction heuristic but uses the new local search SOMA. G9 is the GRASP heuristic implemented by Andrade et al. [2]. Except for G9, which code was kindly provided to us by the authors, all other algorithms were implemented by us.

The algorithms were implemented in C++, compiled with g++ compiler version 3.2.2 and were tested on a PC AMD Athlon 1.3GHz with 256 Mbytes of RAM. Twenty instances for the problem were created with populations of sizes $n = 100$, $n = 200$, $n = 300$, $n = 400$ and $n = 500$, and subsets of sizes $m = 10\%n$, $m = 20\%n$, m=$30\%n$ and $m = 40\%n$. The diversities in the set $\{d_{ij}; i < j; i, j \in N\}$ for each set of instances that have the same population size were randomly selected from a uniform distribution over $[0 \ldots 9]$.

In Tab. 6, we show the results of computing 500 iterations for each GRASP heuristic. The first and second columns identify two parameters of each instance: the size of the population and the number m of elements to be selected. Each procedure was executed three times and for each one we show the average value of the solution cost and the best value found. We can see that the proposed GRASP heuristics found better solutions than GRASP algorithms found in literature [2, 5]. Algorithm G7, which implements the KLD-v2 for construction phase and GhA for local search, was the one that found better solutions for larger number of instances.

Table 7 reports the CPU times observed for the execution of the same instances. The first and second columns identify the two parameters of each instance. For each GRASP heuristic, the average time for three executions and the time obtained when the best solution was found are reported. Among the proposed heuristics, algorithm G5 is the most efficient related to execution time. Heuristic G7, for which we have the best quality solutions, demands more time than G5 but is not the worst one, showing that this algorithm works very well for this problem.

We made a deeper analysis for the results obtained for the GRASP heuristics G1, G5, G6, G7 and G8, which present better solutions and/or shorter execution times. We selected two instances: the first one has parameters $n = 200$ and $m = 40$, and the second one, $n = 300$ and $m = 90$. We executed each GRASP heuristic until a solution was found with a greater or equal cost compared to a target value. Two target values were used for each instance: the worst value obtained by these heuristics and an average of the values generated by them. Empirical probability distributions for the time to achieve a target value are plotted in Fig(s). 3 and 4. To plot the empirical distribution for each variant, we executed each GRASP heuristic 100 times using 100 different random seeds. In each execution, we measured the time to achieve a solution whose cost was greater or equal to the target cost. The execution times were sorted in ascending order and a probability $p_i = (i - 0.5)/100$ was associated for each time t_i and the points $z_i = (t_i, p_i)$ were plotted for $i = 1, \ldots, 100$ [1].

Table 6. Solutions for GRASP heuristics

		G1		G2		G3		G4		G5		G6		G7		G8		G9	
n	m	average	best	average	best	average	best	average	best	average	best	average	best	average	best	average	best	average	best
100	10	318,0	318,0	318,0	318,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0	333,0
100	20	1178,0	1178,0	1178,0	1178,0	1195,0	1195,0	1195,0	1195,0	1195,0	1195,0	1195,0	1195,0	1195,0	1195,0	1195,0	1195,0	1191,7	1195,0
100	30	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2457,0	2454,7	2457,0
100	40	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4142,0	4140,3	4142,0
200	20	1233,0	1233,0	1233,0	1233,0	1245,7	1247,0	1247,0	1247,0	1247,0	1247,0	1247,0	1247,0	1247,0	1247,0	1247,0	1247,0	1243,0	1245,0
200	40	4443,0	4443,0	4442,3	4443,0	4446,7	4448,0	4446,0	4448,0	4446,3	4448,0	4449,3	4450,0	4447,0	4448,0	4448,0	4448,0	4443,3	4445,0
200	60	9437,0	9437,0	9437,0	9437,0	9434,3	9437,0	9437,0	9437,0	9437,0	9437,0	9437,0	9437,0	9437,0	9437,0	9437,0	9437,0	9424,0	9425,0
200	80	16225,0	16225,0	16225,0	16225,0	16224,7	16225,0	16225,0	16225,0	16182,3	16207,0	16170,0	16171,0	16225,0	16225,0	16225,0	16225,0	16200,0	16211,0
300	30	2666,0	2666,0	2666,0	2666,0	2684,0	2694,0	2679,0	2686,0	2686,0	2691,0	2683,0	2684,0	2688,5	2691,0	2692,0	2694,0	2681,3	2691,0
300	60	9652,0	9677,0	9644,3	9654,0	9678,0	9681,0	9678,3	9681,0	9667,7	9689,0	9658,0	9688,0	9682,5	9684,0	9676,0	9679,0	9658,0	9676,0
300	90	20725,0	20725,0	20725,0	20725,0	20721,3	20728,0	20722,3	20733,0	20640,0	20640,0	20640,0	20640,0	20726,0	20727,0	20727,0	20734,0	20673,7	20685,0
300	120	35880,0	35881,0	35880,3	35881,0	35880,0	35881,3	35877,0	35881,0	35871,0	35871,0	35871,0	35871,0	35878,0	35881,0	35881,0	35881,0	35853,5	35855,0
400	40	4615,0	4626,0	4611,0	4626,0	4648,0	4648,0	4654,7	4658,0	4635,3	4653,0	4634,7	4654,0	4647,0	4649,0	4654,0	4655,0	4629,7	4635,0
400	80	16900,5	16903,0	16905,3	16918,0	16946,7	16956,0	16944,7	16948,0	16916,3	16925,0	16902,0	16902,0	16930,0	16937,0	16938,0	16940,0	16873,3	16895,0
400	120	36298,7	36304,0	36301,0	36301,0	36301,0	36315,0	36301,7	36306,0	36175,0	36175,0	36175,0	36175,0	36272,5	36283,0	36290,0	36301,0	36187,7	36191,0
400	160	62442,3	62445,0	62432,0	62433,0	62450,0	62470,0	62439,3	62457,0	62313,0	62313,0	62313,0	62313,0	62478,0	62483,0	62451,5	62454,0	62345,0	62359,0
500	50	7079,3	7082,0	7079,0	7079,0	7110,7	7131,0	7105,3	7130,0	7124,0	7130,0	7116,0	7127,0	7112,5	7115,0	7107,0	7116,0	7082,7	7085,0
500	100	26219,0	26220,0	26219,0	26219,0	26213,0	26224,0	26220,0	26222,0	26197,0	26201,0	26221,0	26254,0	26229,0	26237,0	26229,5	26236,0	26129,0	26144,0
500	150	56571,0	56572,0	56571,5	56572,0	56549,0	56563,0	56554,7	56571,0	57055,7	58605,0	56571,7	56572,0	56572,0	56572,0	56571,0	56572,0	56360,0	56365,0
500	200	97255,0	97274,0	97322,3	97326,0	97316,3	97327,0	97325,5	97327,0	97213,0	97213,0	97333,0	97344,0	97316,5	97319,0	97310,0	97320,0	97166,7	97200,0

Table 7. CPU time for GRASP heuristics (seconds)

		G1		G2		G3		G4		G5		G6		G7		G8		G9	
n	m	average	best	average	best	average	best	average	best	average	best	average	best	average	best	average	best	average	best
100	10	18,0	16,9	19,3	18,5	22,3	21,4	23,1	22,3	10,4	9,6	11,8	11,0	28,0	27,1	30,2	29,3	0,4	0,4
100	20	47,4	45,9	64,7	62,9	53,3	52,3	60,5	59,7	28,2	27,4	37,0	36,0	64,8	63,7	79,5	78,4	2,8	2,7
100	30	88,3	87,2	123,4	123,2	91,0	90,3	113,6	112,3	57,4	56,9	79,1	77,9	111,7	111,0	148,3	147,8	7,5	7,4
100	40	130,94	132,6	198,7	196,9	138,8	137,6	187,1	187,0	89,2	88,3	126,2	125,6	165,6	164,9	234,4	234,0	14,6	14,8
200	20	147,5	145,3	203,4	201,9	224,3	224,2	247,6	247,9	71,7	71,0	101,0	99,6	263,6	262,8	301,1	300,3	6,3	6,2
200	40	561,9	561,6	825,3	827,5	635,6	633,9	835,6	833,1	318,3	317,4	510,3	507,3	784,7	781,1	986,3	982,4	39,3	39,2
200	60	1075,3	1067,3	1665,9	1675,4	1224,0	1215,7	1715,6	1723,6	626,6	628,6	1006,4	998,1	1958,8	1961,0	2807,0	2792,1	113,5	114,3
200	80	1647,0	1638,6	2629,7	2630,0	1850,0	1845,8	2673,7	2678,7	804,3	802,6	1329,0	1327,3	3051,8	3030,9	4502,5	4487,6	225,2	225,8
300	30	712,7	712,7	908,1	904,8	979,5	978,3	1189,9	1190,6	412,7	413,2	644,8	645,5	1159,4	1152,6	1405,3	1402,3	32,3	32,3
300	60	2477,2	2450,2	3557,2	3523,6	2954,4	2949,9	3998,7	4002,4	1690,5	1687,5	2800,1	2807,3	3450,4	3613,17	4774,9	4919,3	215,3	216,7
300	90	5603,2	5606,2	8151,1	8107,9	5966,6	5906,6	8582,0	8593,9	3260,6	3276,4	5164,7	5145,8	6829,5	6841,3	10514,6	10420,9	605,3	605,7
300	120	9740,2	9675,0	15672,3	15587,3	10329,5	10278,8	15842,3	15835,8	4453,0	4452,8	8216,8	8215,7	11458,5	11423,4	17517,72	17562,01	1184,6	1195,2
400	40	2469,3	2457,8	3323,3	3341,2	2790,0	2781,5	3371,0	3370,2	1208,7	1208,3	2044,7	2044,1	3202,7	3195,9	4066,2	4077,4	106,1	112,5
400	80	9651,9	9724,3	14833,6	14849,0	10048,6	10161,6	13772,0	13751,5	5266,5	5258,0	8077,1	8129,3	10464,3	10451,3	15275,1	15283,4	682,5	681,1
400	120	20210,3	20264,5	32205,1	32286,0	19201,3	19220,9	28881,7	28940,3	10097,5	10097,9	16620,7	16553,0	22038,36	22184,8	33594,6	33481,1	1346,0	1891,8
400	160	32217,8	32273,2	51497,7	51819,2	30010,7	30096,4	45161,1	45471,9	15750,3	15684,3	26943,4	27012,2	32843,6	32550,5	52042,6	52056,2	5064,3	5109,6
500	50	5122,4	5176,2	7304,3	7253,1	5625,8	5661,6	7452,6	7340,8	2934,9	2935,6	4555,8	4540,8	6988,6	7005,0	9171,3	9187,4	254,5	254,5
500	100	26128,9	26048,1	31542,5	31546,0	22781,3	22822,1	32888,1	32855,2	11921,7	11916,2	19200,8	19239,3	26513,3	26473,1	39343,5	39416,2	1611,3	1608,2
500	150	54392,6	54342,9	76068,63	75762,54	49617,7	49686,0	76443,5	76249,0	29051,4	27546,0	65082,3	65173,3	55898,1	55594,0	88158,7	88217,9	7922,9	8087,6
500	200	80252,09	80245,14	115137,3	115200,5	75998,8	75810,0	117932,6	117675,1	39043,3	38944,42	86247,0	86513,8	84678,4	84840,8	137503,3	138031,3	28842,6	29133,5

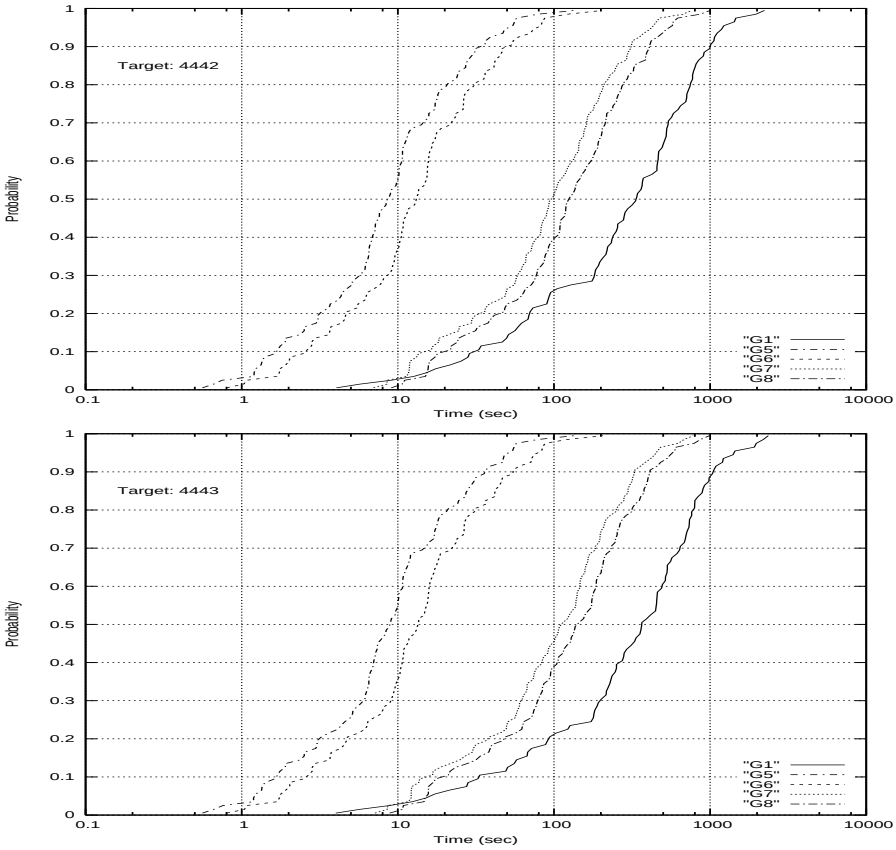

Fig. 3. Comparison of GRASP heuristics for the instance $n = 200, m = 40$ with targets values 4442 and 4443

We compared the proposed GRASP algorithms with Ghosh algorithm (G1) by evaluating the average probability that G1 presents when we have the probability values equal to 0.9 and 1.0 for the proposed GRASP heuristics. We obtain these values from Fig(s). 3 and 4. For example, we can obtain the probability values for G1, when we have a probability value equal to 0.9 for G5. In this case, we have a value of 0.12 for both target values 4442 and 4443, 0.83 for target 20640, and 0.7 for target 20693. The average of these values is 0.44. So we have evaluated these average values for G5, G6, G7 and G8 and the results are presented in Tab. 8.

We can see that although the algorithm G1 presents a good convergence to the target values, the proposed algorithms G5, G6, G7 and G8 were able to improve this convergence.

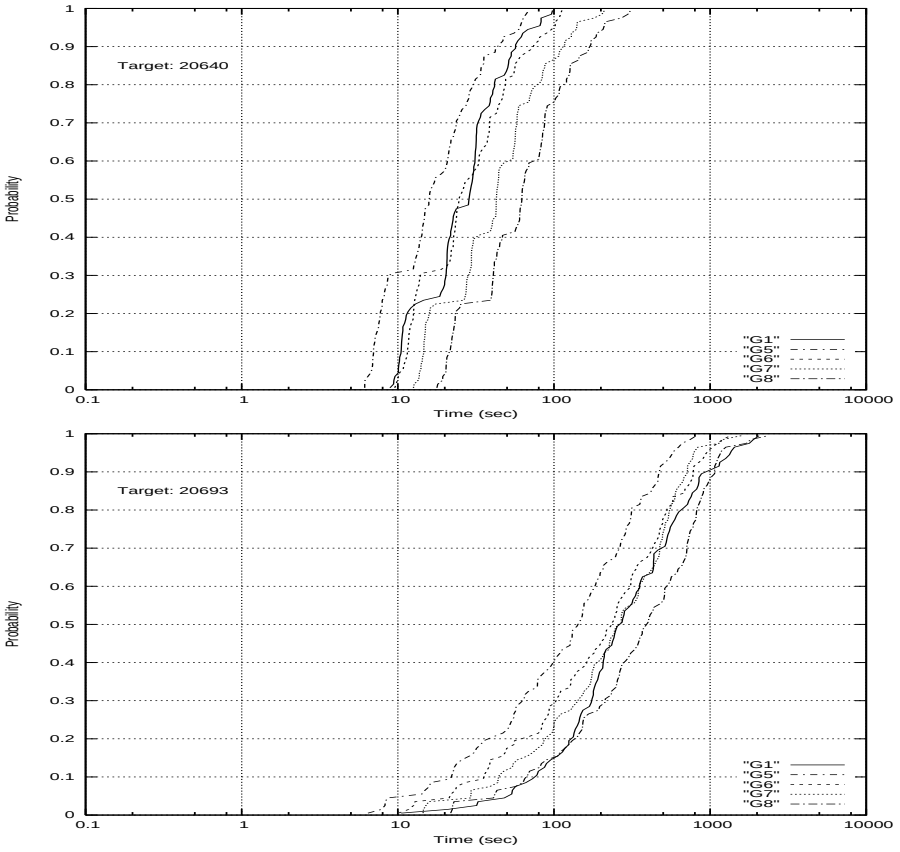

Fig. 4. Comparison of GRASP heuristics for the instance $n = 300, m = 90$ with targets values 20640 and 20693

4 Concluding Remarks

This paper presented some versions of GRASP heuristic to solve the maximum diversity problem (MDP). The main goal of this work was to analyse the influence of the construction and local search heuristics on the performance of GRASP techniques.

Experimental results show that the versions that use KLD or KLD-v2 construction algorithms and Gha or SOMA local search algorithms (G5, G6, G7 and G8) significantly improve the average performance of the best GRASP approaches proposed in the literature (G1 and G9).

Our experiments also show that if the execution time is restricted (limited to smaller value), version G5 is a good choice since it obtains reasonable results faster (see Fig(s). 3 and 4). On the other hand, if the execution time is not an

Table 8. Comparison of convergence of solutions

probability	G1-G5	G1-G6	G1-G7	G1-G8
0.9	0.44	0.5	0.7	0.75
1.0	0.6	0.67	0.91	0.95

issue, versions G7 and G8 tend to produce the best solutions (see Tabs. 6 and 7).

Acknowledgments. The authors acknowledge *LabPar* of PUC-RIO (Rio de Janeiro, Brazil) for making available their computational facilities on which some computational experiments were performed. We thank Paulo Andrade for allowing us to use the code developed by him for algorithm G9. We also acknowledge Coordination of Improvement of Personnel of Superior Level (CAPES) support for providing a master fellowship to Geiza C. Silva.

References

1. Aiex, R. M., Resende, M. G. C., Ribeiro, C. C.: Probability distribuition of solution time in GRASP: an experimental investigation, Journal of Heuristics, **8** (2002), 343–373
2. Andrade, P. M. F., Plastino, A., Ochi, L. S., Martins, S. L.: GRASP for the Maximum Diversity Problem, Proceedings of MIC 2003, (2003)
3. Bhadury J., Joy Mighty E., Damar, H.: Maximing workforce diversity in project teams: a network flow approach, Omega, **28** (2000), 143–153
4. Feo T. A., Resende, M. G. C.: Greedy randomized adaptive search procedures, Journal of Global Optimization **6** (1995), 109–133
5. Ghosh, J. B.: Computational aspects of maximum diversity problem, Operations Research Letters **19** (1996), 175–181
6. Glover, F., Hersh, G., McMillan C.: Selecting subsets of maximum diversity, MS/IS Report No. 77-9, University of Colorado at Boulder, (1977)
7. Glover, F., Kuo, C-C., Dhir,K. S.: Integer programming and heuristic approaches to the minimum diversity problem, Journal of Business and Management **4(1)**, (1996), 93–111
8. Hansen, P., Mladenović, N.: An introduction to variable neighborhood search, Metaheuristics, Advances and Trends in Local Search, Paradigms for Optimization, S. Voss et al. editors, (1999) 433–458
9. Katayama, K., Naribisa, H.: An evolutionary approach for the maximum diversity problem, Working Paper, Department of Information and Computer Engineering, Okayama University of Science, (2003)
10. Kochenberger, G., Glover, F.: Diversity data mining, Working Paper, The University of Mississipi, (1999)
11. Prais, M., Ribeiro, C. C.: Reactive GRASP: an aplication to a matrix decomposition problem in TDMA traffic assignment, INFORMS Journal on Computing **12** (2000), 164–176
12. Weitz, R., Lakshminarayanan, S.: An empirical comparison of heuristic methods for creating maximally diverse group, Journal of the operational Research Society **49** (1998), 635–646