Yugoslav Journal of Operations Research
24 (2014) Number 1, 21-33
DOI: 10.2298/YJOR121223003U

VARIABLE NEIGHBORHOOD SEARCH FOR MAXIMUM DIVERSE GROUPING PROBLEM

Dragan UROŠEVIĆ
Mathematical Institute SANU
draganu@mi.sanu.ac.rs

Received: December 2012 / Accepted: January 2013

Abstract: This paper presents a general variable neighborhood search (GVNS) heuristic for solving the maximum diverse grouping problem. Extensive computational experiments performed on a series of large random graphs as well as on several instances of the maximum diversity problem taken from the literature show that the results obtained by GVNS consistently outperform the best heuristics from the literature.

Keywords: Combinatorial optimization, Maximum Diverse Grouping, Metaheuristics, Variable Neighborhood Search.

MSC: 90C59, 90C27, 90C06, 90C20.

1. INTRODUCTION

The maximum diverse grouping problem (MDGP) consists of finding a way to divide a set of n elements into m mutually disjoint groups so that the total diversity among the elements belonging to the same group is maximized. The diversity among the elements in a group is calculated as the sum of the individual distance between each pair of elements. The objective of the problem is to maximize the overall diversity, i.e., the sum of the diversity of all groups.

Feo and Khellaf [10] proved that the MDGP is NP-hard. The MDGP is also known as the k-partition problem (Feo et al. [9]), and the equitable partition problem (Mingers and O'Brien [17], O'Brien and Mingers [20]) that appears in a wide range of real life situations. The first application is in designing of VLSI circuits (Chen [4]; Feo and Khellaf [10]). Also, it can be applied in storing large programs onto paged memory (Kral, [15]), where the subroutines of a program have to be stored onto pages of available memory. In this case, the objective is to maximize data transfer between subroutines on

the same page (minimizing in this way data transfers between different pages). One of the most popular MDGP applications appears in academic context when forming student groups [20]. Specifically, in business schools, it is nowadays common to create diverse student workgroups or training teams in order to provide students with diverse environment [23]. Note that this type of student grouping problem is different from the one of making MBA student teams [6], where groups are supposed to be as similar as possible. In that way, the members of the same group diverse. Lofti and Cerveny [16] proposed Lofti-Cerveny (LC) method for minimizing (instead of maximizing) diversity in each group, which was also a part of a method for scheduling final exams.

The MDGP can be applied in forming diverse groups of reviewers in scientific publications or project evaluation in scientific funding agencies [13]. Also, workforce diversity is an increasing phenomenon in organizations. Creating diverse groups where people with different background work together, is a way to deal with this heterogeneity and to facilitate their communication (Bhadury et al. [2]).

1.1. Mathematical formulation

There are two variants of the MDGP. The better known is (MDGP1), where all groups are forced to have the same number $k = n/m$ of elements. The second variant, (MDGP2) allows the variable size of groups. So, the number c_g of elements in a group g can be in the interval $\left[a_g, b_g\right]$, where $a_g \le b_g$ for $g = 1,2,\ldots,m$. Obviously, (if we set $a_g = b_g = n/m$ for each g) MDGP1 problem is a special case of MDGP2 (). So, if we developed the method for solving MDGP2, the same method could also be used for solving MDGP1. In the remainder, MDGP will refer to the general case MDGP2. Both variants can be formulated as quadratic integer programs. Denote with x_{ig} ($i = 1,2,\ldots,n$ and $g = 1,2,\ldots,m$) binary variables defined as follows:

$$x_{ig} = \begin{cases} 1, & \text{if element } i \text{ belongs to group } g \\ 0, & \text{otherwise} \end{cases}$$

A quadratic integer programming formulation of MDGP2 is [8]:

$$\max z = \sum_{g=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ig} x_{jg} \tag{1}$$

such that

$$\sum_{g=1}^{m} x_{ig} = 1, \quad i = 1,2,\ldots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ig} \ge a_g, \quad g = 1,2,\ldots,m \tag{3}$$

$$\sum_{i=1}^{n} x_{ig} \le b_g, \quad g = 1,2,\ldots,m \tag{4}$$

Constraint (2) provides that each element i is included (distributed) in exactly one group. Constraints (3) and (4) provide that group g contains, at least a_g, and b_g elements at most.

1.2. Previous work

Arani and Lotfi [1] developed multistart algorithm. This procedure consists of a random construction followed by an improving phase that partially deconstructs the random solution, and examines all possible reconstructions in order to determine the best one. This process is repeated until no reconstructions giving a better solution are found (i.e., when the current solution cannot be improved). Feo and Khellaf [10] proposed several heuristics based on graph theory. They considered the special case of even-sized groups, odd-sized groups, and groups having 2^i elements. The authors prove that this way obtained values are within a bounded percentage of the optimal solution.

Weitz and Jelassi [21] developed a basic constructive heuristic (WJ). The idea is to avoid the assignment of very similar elements to belong to the same group. WJ first randomly assigns an element to the first group. The heuristic then selects the element with the smallest distance to the previously considered element and assigns it to the next group. When a sweep of all groups has been completed, the procedure goes back to the first group. The construction finishes when all the elements are assigned.

Weitz and Lakshminarayanan [22] discovered and corrected the errors in the LC method. The modification of the LC method developed for the MDGP (LCW method) is presented by Weitz and Lakshminarayanan [23].

Weitz and Lakshminarayanan in [23] made detailed experimentation to compare all heuristics for the MDGP known at that time. They conclude that the LCW heuristic is the best. The LCW is an improvement method that can start from a random solution or a solution generated with the Weitz-Jelassi. The authors did not find significant differences in solution quality when LCW was started from a random solution and when the WJ construction method was used to determine starting solution. They however concluded that there was significant difference in computational time.

Fan et al. [8] presented a hybrid genetic algorithm (LSGA) for the solution of the MDGP. LSGA combines a genetic algorithm and a local search procedure. The genetic aspect of LSGA is based on the encoding scheme for grouping problems proposed by Falkenauer [7]. The local search within LSGA implements a best improvement strategy based on exchanging elements between groups. Fan et al.'s is the first publication that describes a method for the general version of the MDGP applicable to different group sizes.

Gallego et al. [11] proposed Tabu Search for solving MDGP. The local search is based on exchanging elements between groups and moving one element from one to the other group. In order to avoid cycling, elements moved within any iteration can not be moved in the next *TabuTenure* (a parameter of method) iterations. Also, they allowed visiting cardinality-infeasible solution during the search. More precisely, they apply previously described Tabu Search in order to divide elements in a group so that each group g has at least $a_g - k$ and at most $b_g + k$ elements ($k = 1,2,\ldots,k_{\max}$). After finishing the search, they repair the obtained local optima by moving elements from the groups

that have more than b_g elements to the groups that have less than a_g elements. They call their method Tabu Search with Strategic Oscillation and denote it with SO. In this paper for the first time is suggested VNS based heuristic for maximum diversity grouping problem.

The rest of the paper is organized as follows: In Section 2, we provide the detailed rules of General VNS (GVNS) heuristics for solving the MDGP. This includes a description of the Neighborhoods used within Variable neighborhood descent (VND). Section 3 gives a summary of computational results on several problem instances from the literature, and Section 4 concludes the paper.

2. GENERAL VARIABLE NEIGHBORHOOD SEARCH FOR MDGP

In this section, we describe the components of our GVNS heuristic: the solution space, the three local search neighborhoods used (Insertion, 2-opt and 3-opt), the shaking step, and our new GVNS algorithm.

2.1. Solution space

Solution space consists of all possible feasible divisions of elements into groups. A division is feasible if and only if each created group g contains at least a_g and at most b_g elements. Solution is represented with an array x^c of length n such that $x^c[i]$ is the label of the group containing element i ($i = 1, 2, \ldots, n$). In order to speedup local search, we also maintain matrix sd^c such that $sd^c[i][g]$ is the sum of diversities between element i and all elements assigned to the group g in the current solution:

$$sd^c[i][g] = \sum_{\substack{x^c[j]=g \\ j=1,2,\ldots,n}} d_{ij} .$$

Note that for the current solution, matrix sd can be computed in $O\left(n^2\right)$.

2.2. VND Local Search

The Local Search is organized as Variable Neighborhood Descent. The following neighborhoods are designed: *Inseriont*, *Swap* and 3-*Chain*.
Neighborhood *Insertion* contains solutions obtained by moving only one element from the current group to the other group. By using previously described matrix sd^c, it is possible to efficiently compute, for each feasible move, the change of the objective function value. Denote with x^n the solution obtained form a solution x^c by moving element i from a group g_1 to a group g_2. In this case, the sum of diversities in all groups except the groups g_1 and g_2, are unchanged. The element i is removed from the group g_1, and because of that, the sum of diversities in the group g_1 decreases for the sum of diversities between i and all other elements belonging to the group g_1. Element i is inserted into g_2, hence, the sum of diversities in g_2 increases for all diversities between i

and elements belonging to the group g_2. It is easy to conclude that the difference between objective function values for solutions x^c and x^n is

$$\Delta f = f(x^n) - f(x^c) = sd^c[i][g_2] - sd^c[i][g_1].$$

If we perform *Insertion* step, we change the current solution and then it is necessary to update matrix *sd*. If the element *i* moves from the group g_1 to the group g_2, then groups g_1 and g_2 are modified and values $sd^c[j][g_1]$ and $sd^c[j][g_2]$ must be updated in the following way

$$sd^c[j][g_1] = sd^c[j][g_1] - d[j][i]$$

and

$$sd^c[j][g_2] = sd^c[j][g_2] + d[j][i].$$

Since this updating must be performed for each element *j*, updating of matrix *sd* after performing *Insertion* move has the complexity $O(n)$. On the other hand, the cardinality of *Insertion* neighborhood is $O(nm)$.

Neighborhood *Swap* contains solutions obtained by swapping pair of elements belonging to different groups (elements exchange the group currently assigned to). Let an element *i* be in a group g_i and an element *j* in a group g_j in the current solution x^c. Denote with x^n solution obtained after moving element *i* into group g_j and element *j* into group g_i. Since the element *i* is removed from the group g_i, the diversities between element *i* and elements remaining in the group g_i do not contribute to objective function value of a new solution. But, because the element *i* is inserted in the group g_j, all diversities between *i* and the elements belonging to the group g_j contribute to objective function value of a new solution. Similar facts are true for the element *j*. So, we can finally calculate difference between objective value for a new and for the old solutions:

$$\Delta f = f(x^n) - f(x^c) = \left(sd^c[i][g_j] - sd^c[i][g_i]\right) + \left(sd^c[j][g_i] - sd^c[j][g_j]\right) - 2d[i][j]$$

It is obvious that the change in objective value for each solution from neighborhood *Swap* is done in $O(1)$. Cardinality of *Swap* is $O\left(n^2\right)$. After performing *Seap* move, it is necessary to update the matrix *sd,* and complexity of this update is $O(n)$ (we can consider 2-opt as two successive Insertions).

One 3-*Chain* move is determined by three elements *i*, *j* and *k* belonging to three different groups: g_i, g_j and g_k, respectively, and by moving the element *i* to the group g_j, the element *j* to the group g_k and the element *k* to the group g_i. Neighborhood 3-*Chain* consists of solution obtained by performing 3-*Chain* move.

If we denote with x^c and x^n solutions before and after the move, we can calculate the difference between objective function values as follows:

$$\Delta f = f(x^n) - f(x^c) = \left(sd^c[i][g_j] - sd^c[i][g_i]\right) +$$
$$+\left(sd^c[j][g_k] - sd^c[j][g_j]\right) +$$
$$+\left(sd^c[j][g_k] - sd^c[j][g_j]\right) -$$
$$+\left(d[i][j] + d[j][k] + d[k][i]\right)$$

So, complexity of solution checking is $O(1)$, but complexity (cardinality) of the whole neighborhood is $O(n^3)$.

We develop two variants of variable neighborhood descent:

- VND-2 – use neighborhoods Insertion and Swap, in this order
- VND-3 – use all three developed neighborhoods in the following order: Insertion, Swap, 3-Chain.

2.3. Initial solution

Calculation of an initial solution is done in two phases. First, we ensure that each group g contains at least a_g elements by inserting the chosen elements. In the second phase, we distribute the remaining elements so that each group g contains at most b_g elements. At the beginning, we select m elements at random and insert them into different groups.

Denote with E_g a set of elements currently assigned to a group g. During the first phase, we maintain the set of groups that have fewer elements than the desired minimum:

$$G' = \left\{g : \left|E_g\right| < a_g\right\}$$

Each iteration of the first phase consists of selecting at random one unassigned element, denoted with i, and assigning it to a selected group. The selected element must be inserted into one of the groups from the set G'. So, for each group $g \in G'$, we calculate average distance between the selected element i and all the elements belonging to the group g, and select the group whose average distance value is maximal. The selected group g is one for which expression D_{ig} defined as

$$D_{ig} = \frac{\sum_{j \in E_g} d_{ij}}{\left|E_g\right|}$$

is maximized.

The first phase ends when the set G' becomes empty. During the second phase, we maintain a set of groups that have fewer elements than the desired maximum:

$$G' = \left\{g : \left|E_g\right| < b_g\right\}.$$

All other steps of the second phase are the same. The second phase finishes when all elements have been assigned.

The initial solution can also be created at random: instead of assigning a selected element to a group with maximal average distance, we assign it to a randomly selected group.

2.4. Shaking

In order to perform perturbation of the incumbent solution, we define k^{th} neighborhood as a set of solutions obtained by k consecutive Swap moves. Thus, in the Shaking step, we generate a random solution from the k^{th} neighborhood of the current solution by performing k random Swap moves. In each move i, $i = 1, \ldots, k$, we randomly select two elements that belong to two different groups.

2.5. GVNS for MDGP

Variable Neighborhood Search combines previously described VND and the Shaking step. We propose small modification of the basic VNS: during execution, we periodically restart VNS search starting from the new initial solution. The new initial solution created each time when the neighborhood counter k reaches the maximal allowed value (k_{max}) n_{rest} times. The number n_{rest} is a parameter of the method and in our experiments is set to 2. Similar modification of the basic VNS may be seen in [19]. Pseudocode for GVNS is given in Figure 1.

```
program GVNS(xopt, kmin, kmax, kstep, tmax, nrest)
  x := InitialSoution;
  x := VND(x);
  xopt := x;
  k := kmin;
  niter := 0;
  while RunningTime < tmax do
    x' := Shake(x, k);
    x'' := VND(x')
    if f(x'') > f(x) then
      x := x''; k := kmin;
    else
      k := k + kstep;
      if k > kmax then
        niter := niter+1;
        if niter = nrest then
          if f(x) > f(xopt) then xopt := x;
          x := IniitalSolution;
          niter := 0;
        endif
        k := kmin;
      endif
    endif
  endwhile
```

3. COMPUTATIONAL RESULTS

In this section the computational results of the proposed GVNS heuristics and a comparison with existing algorithms are given. The proposed method is implemented in C++ programming language. All computations are performed on computer based on Intel Pentium Core Duo (2.66GHz) with 6GB RAM, running the Linux operating system.

3.1. Test instances

We used 480 instances in our experimentation. This benchmark set of instances, referred to as MDGPLIB, is available at http://www.optsicom.es/mdgp. The set is divided into three subsets:

1. *RanReal* — This set consists of 160 instances in which the distance values d_{ij} are real numbers generated by using a Uniform distribution $U(0,100)$. The number of elements n, the number of groups m and the limits of each group size a_g and b_g are shown in Table 1. There are 20 instances for each combination of parameters (i.e., each row in Table 1): 10 for instances with equal group size (EGS), and 10 for instances with different group size (DGS). For the 10 instances in EGS, the group size is equal for all instances and is calculated as $\lfloor n/m \rfloor$. For the 10 instances in DGS, the limits of each group (a_g and b_g) for each instance are generated randomly in the predefined interval. That is, the value of a_g is generated in the interval $\left[a_g^{\min}, a_g^{\max}\right]$ and the value of b_g is generated in the interval $\left[b_g^{\min}, b_g^{\max}\right]$. This data set was introduced by Fan et al. [8] with n ranging from 10 to 240. Gallego at al. [11] have generated larger instances with $n = 480$ and $n = 960$.

Table 1. Summary of value of parameters used to generate instances

n	m	$a_g = b_g$	$a_g^{\min}$	$a_g^{\max}$	$b_g^{\min}$	$b_g^{\max}$
10	2	5	3	5	5	7
12	4	3	2	3	3	5
30	5	6	5	6	6	10
60	6	10	7	10	10	14
120	10	12	8	12	12	16
240	12	20	15	20	20	25
480	20	24	18	24	24	30
960	24	40	32	40	40	48

2. *RanInt* — This set has the same structure and size as *RanReal* (shown in Table 1) but distances are generated with an integer Uniform distribution $U(0,100)$.

3. *Geo* — This set follows the same structure and size as the previous two, however, distance values are calculated as Euclidean distances between a pair of points with coordinates randomly generated in [0,10]. The number of coordinates for each instance is generated randomly in the range 2 to 21. Glover et al. [12] introduced this generator for the Maximum Diversity Problem.

3.2. Local Search

In the first series of experiments, we compare local searches in the three defined neighborhood structures together with the two VND procedures. First, we examine the test case with n=960 and m=24 (different group sizes variant). One thousand initial points are generated by using greedy procedure described above, and a local search is performed from each one in each type of neighborhood. For each local optima obtained by local search, percentual deviation over the best known solution (obtained by GVNS) is calculated. In Table, 2 we give the summarized results. For each neighborhood (i.e. corresponding Local search) minimal, maximal, and average deviation is shown. Also, the average computational time is given.

Table 2. Comparison of local searches on RanReal instance with n=960 and m=24

Local Search	Percentual deviation over best			Time
	Min	Max	Avg	
Insertion	6.64	9.59	8.45	0.03
Swap	2.12	3.67	2.70	0.12
3-Chain	1.64	3.35	2.21	58.58
VND-2	1.01	2.49	1.47	0.15
VND-3	0.39	2.28	0.87	23.20

We observe that

- Local search in neighborhood 3-Chain produce better solutions than the local search in Swap, but average execution time is significantly longer (0.12 sec vs 58.58 sec).
- VND-2 as a local search gives better results than Local Search in any single neighborhood explored. As in the other problems (see e.g. [14]), VND-2 gives better results than local search in 3-Chain; however VND-2 is significantly faster.
- VND-3 local search gives the best results, but the execution times are very long.

Based on this analysis, we decided to use VND-2 as Local Search within GVNS.

3.3. Initial Solution

The second experiment is dedicated to comparing methods for generating the initial solution. We propose two methods for computing an initial solution: greedy and random. One thousand random initial solutions were generated and local search was performed on each of them. Similarly, one thousand greedy initial solutions were produced and then the local search performed. Table 3 summarizes the results obtained. For each neighborhood minimal, maximal and average percentual deviation of local optima over the best known is given. From this Table, we conclude that the results obtained by local search from greedy initial solution in each type of neighborhood is better than the results produced by local search from the random initial solution. Also, the execution times for local search that start from random solutions are longer. Based on

these results, we decided to use greedy initial solution as the initial solution within our GVNS.

Table 3. Influence of the initial solution to local search. All tests are performed on RanReal instance with *n*=960 and *m*=24

	Random Initial				Greedy Initial			
	Percentual deviation			Time	Percentual deviation			Time
	Min	Max	Avg		Min	Max	Avg	
Insertion	9.55	16.05	13.29	0.04	6.64	9.59	8.45	0.03
Swap	3.19	4.55	3.92	0.21	2.12	3.67	2.70	0.12
3-Chain	2.78	3.86	3.37	110.34	1.64	3.35	2.21	58.58
VND-2	2.02	3.18	2.60	0.24	1.01	2.49	1.47	0.15
VND3	1.33	2.66	2.00	33.47	0.39	2.28	0.87	23.20

3.4. Importance of "restarting" search within GVNS

As we have said in the previous sections, we modify the basic VNS, and during the complete GVNS, we periodically restart search from new greedy initial solution. In the next experiment, we compare classical GVNS and the variant with periodical restart. We tested these two variants on 10 DFS (different groups sizes) instances with *n*=960 elements and *m*=24 groups. Both methods were executed 20 times for each instance. In both methods, the same values for the parameters were used: $k_{\min} = 2, k_{\max} = 60, k_{\text{step}} = 2$ and $n_{\text{rest}} = 2$ .Table 4 reports the following: the best results obtained by the corresponding method, the worst results obtained by the corresponding method, and the average results.

Table 4. Comparing GVNS without periodical restart and GVNS with periodical restart. All tests are performed on RanReal instance with *n*=960 and *m*=24

Inst.	GVNS without restart				GVNS with restart			
	Best	Worst	Average	Time	Best	Worst	Average	Time
1	1227100.66	1219482.15	1223016.96	487.41	1234250.65	1231618.31	1233060.24	598.63
2	1225827.52	1222346.79	1223591.46	506.97	1234270.24	1229926.25	1232227.12	598.08
3	1222255.02	1217975.21	1219958.91	441.99	1233449.04	1228500.63	1230290.44	596.60
4	1224430.75	1221272.56	1223042.71	520.18	1234128.75	1229089.85	1231650.62	598.59
5	1223264.49	1216268.87	1220081.21	522.75	1231449.40	1228681.93	1230390.78	598.34
6	1220233.04	1213424.31	1216473.92	434.20	1228508.07	1225002.23	1227408.62	599.22
7	1224728.65	1217371.11	1221595.41	530.00	1233944.50	1229588.32	1231708.07	598.73
8	1219956.60	1213306.64	1216684.82	508.56	1227800.50	1223997.87	1225626.77	598.05
9	1224978.46	1219785.72	1222278.05	413.52	1232149.20	1228155.81	1229949.28	598.79
10	1227047.80	1221932.72	1225050.13	538.63	1235071.88	1231252.82	1233442.31	596.32
Avg.	1223982.30	1218316.61	1221177.36	490.42	1232502.22	1228581.40	1230575.43	598.13

From the table, we conclude that restarting allows us to reach significantly better results. In total, the average of the best results obtained by restarting is 0.69% better than the average of the bests without restart (compare 1223982.30 with 1232502.22 of this maximization problem).

3.5. Main computational results

Extensive testing of VNS was carried out on the Geo, RanReal and RanInt instances with $n \geq 120$. As before, 20 restarts were performed on each test instance. In Tables 5 and 6, results on DGS and EGS instances are given, respectively. There VNS results are compared with the state of the art heuristic methods: Genetic Algorithm (LSGA), (Fan et al. [8]), and Tabu search based heuristic with strategic oscillation (SO) (Gallego et al [11]). In order to save the space, we give average of the best obtained solutions by corresponding methods on 10 test instances that have the same parameters. Execution times are not reported, but for all methods the same time limits are used: $t_{\max} = 3$ seconds for instances with n=120, 20 seconds for instances with n=240, 120 seconds if n=480 and 600 seconds for instances with n=960. VNS parameters had the following values: $k_{\min} = 2, k_{\max} = 60, k_{\text{step}} = 2$ and $n_{\text{rest}} = 2$.

Table 5. Comparison of the results obtained by LSGA, SO and VNS on DGS instances.

Type	n	Objective function value			Percentual improvement of GVNS	
		LSGA	SO	GVNS	LSGA	SO
GEO	120	97999.08	98346.30	98863.27	0.87	0.52
	240	316772.10	318601.53	319226.91	0.77	0.20
	480	805860.40	810454.88	813046.48	0.88	0.32
	960	2120184.24	2129246.04	2136374.79	0.76	0.33
RanInt	120	48578.80	49673.00	50101.40	3.04	0.86
	240	154854.40	158005.20	159445.00	2.88	0.90
	480	375525.20	384059.80	388167.20	3.26	1.06
	960	1197463.80	1217303.60	1234104.80	2.97	1.36
RanReal	120	48510.03	49596.01	50028.62	3.04	0.86
	240	155350.50	157670.80	159393.36	2.54	1.08
	480	374379.24	382799.76	387802.26	3.46	1.29
	960	1193044.24	1214489.86	1233366.77	3.27	1.53

Table 6. Comparison of the results obtained by LSGA, SO and VNS on EGS instances.

Type	n	Objective function value			Percentual deviation over GVNS	
		LSGA	SO	GVNS	LSGA	SO
GEO	120	88105.63	90229.35	90232.59	2.36	0.00
	240	304291.82	304394.92	304415.73	0.04	0.01
	480	773657.22	773877.11	773922.65	0.03	0.01
	960	2072420.16	2072854.19	2072906.73	0.02	0.00
RanInt	120	46711.60	46963.20	47270.80	1.18	0.65
	240	152203.40	153819.00	155387.80	2.05	1.01
	480	368996.40	373689.20	377942.00	2.37	1.13
	960	1184873.80	1202655.80	1214692.00	2.45	0.99
RanReal	120	46845.00	47035.06	47326.24	1.02	0.62
	240	152519.75	153762.52	155188.43	1.72	0.92
	480	367860.81	372950.14	377173.78	2.47	1.12
	960	1183029.83	1199536.45	1213396.01	2.50	1.14

From these results, we can conclude that our VNS outperforms the best known methods for solving Maximum Diversity Group Problem. For example, on the largest instances VNS improves solutions obtained with Tabu search based heuristic (SO) for more than 1% on average. The improvement increases with the increase of the size of instances. So, we can say that VNS is a robust method for solving MDGP. Note that our VNS uses relatively small number of neighborhoods. Our future research will be devoted to developing new neighborhood structures and involve them in both shaking (diversification) and local search (intensification).

4. CONCLUSION

Here we develop a General VNS (GVNS) heuristic for solving the Maximum Diversity Group Problem (MDGP). The Variable neighborhood descent (VND) based local search uses two different neighborhood structures. Local search in these neighborhoods is optimized by using appropriate data structures. Also, periodical restart from the new solution is proposed and implemented. It is shown that this approach leads to improvement of performances of the method and allows us to reach solutions for about 0.5% better than the solutions found by classical VNS.

Acknowledgement This work is supported by the Serbian Ministry for Education and Science, Grants ON174010 and III44006.

REFERENCES

[1] Arani, T. and Lotfi V., "A three phased approach to final exam scheduling", *IIE Transactions,* 21 (1989) 86-96.

[2] Bhadury, J., Mighty E. J. and Damar, H., "Maximizing workforce diversity in project teams: A network flow approach", *Omega*, 28 (2000) 143–153.

[3] Carrizosa, E., Mladenović, N. and Todosijević, R., "Sum-of-Squares Clustering on Networks", *Yugoslav Journal Of Operations Research*, 21 (2011) 157-161.

[4] Chen, C. C., "Placement and partitioning methods for integrated circuit layout", Ph.D. Dissertation, EECS Department, University of California, Berkeley (1986).

[5] Chen, Y., Fan, Z. P., Ma, J. and Zeng S., "A hybrid grouping genetic algorithm for reviewer group construction problem", *Expert Systems with Applications*, 38 (2011) 2401-2411.

[6] Desrosiers J, Mladenovic N and Villeneuve D., "Design of balanced MBA student teams", *Journal of Operational Research Society*, 56 (2005) 60-66.

[7] Falkenauer, E., Genetic Algorithms for Grouping Problems, Wiley: New York (1998).

[8] Fan, Z. P., Chen, Y., Ma, J. and Zeng, S., "A hybrid genetic algorithmic approach to the maximally diverse grouping problem", *Journal of the Operational Research Society*, 62 (2011) 92-99.

[9] Feo, T., Goldschmidt, O. and Khellaf, M. "One-half approximation algorithms for the k-partition problem", *Operations Research*, 40 (1992) S170-S173.

[10] Feo, T. and Khellaf, M., "A class of bounded approximation algorithms for graph partitioning", *Networks*, 20 (1990) 181-195.

[11] Gallego M., Laguna, M., Martí, R., Duarte, A., "Tabu search with strategic oscillation for the maximally diverse grouping problem", *Journal of the Operational Research Society,* 64 (5) 724-734.

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

[13] Hettich S. and Pazzani, M. J., "Mining for element reviewers: Lessons learned at the national science foundation", in: *Proceedings of the KDD'06*, ACM: New York, NY, 862–871.
[14] Ilić, A., Urošević, D., Brimberg, J. and Mladenović N., "A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem", *European Journal of Operational Research,* 206 (2010) 289-300.
[15] Kral, J., "To the problem of segmentation of a program", *Information Processing Machines*, 2 (1965) 116-127.
[16] Lotfi V. and Cerveny, R., "A final exam scheduling package", *Journal of the Operational Research Society*, 42 (1991) 205-216.
[17] Mingers, J. and O'Brien, F. A., "Creating students groups with similar characteristics: a heuristic approach", *Omega*, 23 (1995) 313-321.
[18] Mladenović, N., Todosijević, R. and Urošević, D., "An Efficient General Variable Neighborhood Search for Large Traveling Salesman Problem With Time Windows", *Yugoslav Journal Of Operations Research*, 22 (2012) .
[19] Mladenović, N., Urošević, D., Perez-Brito, D. and Garcia-Gonzalez C.G., "Variable neighbourhood search for bandwidth reduction", *European Journal of Operational Research,* 200 (2010) 14-27.
[20] O'Brien, F. A. and Mingers, J., "The equitable partitioning problem: a heuristic algorithm applied to the allocation of university student accommodation", Warwick Business School, Research Paper no. 187 (1995).
[21] Weitz, R. R. and Jelassi, M. T., "Assigning students to groups: a multi-criteria decision support system approach", *Decision Sciences*, 23(3) (1992) 746-757.
[22] Weitz, R. R. and Lakshminarayanan, S., "On a heuristic for the final exam scheduling problem", *Journal of the Operational Research Society*, 47 (4) (1996) 599-600.
[23] Weitz, R. R. and Lakshminarayanan, S. "An empirical comparison of heuristic and graph theoretic methods for creating maximally diverse groups, VLSI design, and exam scheduling", *Omega*, 25 (4) (1997) 473-482.
[24] Weitz, R. R. and Lakshminarayanan, S., "An empirical comparison of heuristic methods for creating maximally diverse groups", *Journal of the Operational Research Society*, 49 (6) (1998) 635-646.