# Max-Diversity Orthogonal Regrouping of MBA Students Using a GRASP/VND Heuristic

Matías Banchero[1], Franco Robledo[1], Pablo Romero[1](✉), Pablo Sartor[2], and Camilo Servetti[1]

[1] Instituto de Computación, INCO, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay
{matias.banchero,frobledo,promero,camilo.servetti}@fing.edu.uy

[2] IEEM Business School, Universidad de Montevideo, Lord Ponsomby 2542, Montevideo, Uruguay
psartor@um.edu.uy

**Abstract.** Students from Master in Business Administration (MBA) programs are usually split into teams. Many schools rotate the teams at the beginning of every term, so that each student works with a different set of peers during every term. Diversity within every team is desirable regarding gender, major, age and other criteria. Achieving diverse teams while avoiding -or minimizing- the repetition of student pairs is a time-consuming complex task for MBA Directors.

The Max-Diversity Orthogonal Regrouping (MDOR) problem is here introduced, where the goal is to maximize a global notion of diversity, considering multiple stages (i.e., terms) and intra-diversity within the teams. A hybrid GRASP/VND heuristic combined with Tabu Search is developed for its resolution. Its effectiveness has been tested in real-life groups from the MBA program offered at IEEM Business School, Universidad de Montevideo, Uruguay, with a notorious gain regarding team diversity and repetition level.

**Keywords:** MBA teams · Orthogonal regrouping · Diversity · GRASP · VND

## 1 Motivation

The collaborative team-formation and staffing/scheduling problems in workforce management is of paramount importance in projects deployment and large/scale corporations. Given the intrinsic hardness of multidisciplinary team-formation and clustering techniques, it is necessary to develop tools for this task. In this work we are focused on a maximum diversity regrouping assignment of MBA students; nevertheless, the reader can find potential applications in similar clustering problems. Experience shows that the student skills and learning process benefit significantly from highly-diverse teams when regarding prior experience,

N. Mladenovic et al. (Eds.): ICVNS 2021, LNCS 12559, pp. 58–70, 2021.
https://doi.org/10.1007/978-3-030-69625-2_5

age, gender, major and other features. MBA programs are usually split into four to six terms. Many MBA rotate the groups in every term so that students train their ability to adapt to different groups, benefit from new points of view and expand their peer network. Creating highly-diverse teams while keeping at a minimum the repetition of peer-pairs between terms is a very challenging problem faced by program directors at the beginning of every trimester.

The contributions of this paper can be summarized in the following items:

1. A novel combinatorial optimization problem called Max-Diversity Orthogonal Regrouping (MDOR) is here introduced. The goal is to find as many clusterings as terms, maximizing cluster diversity while keeping at a minimum the repetitions of pairs.
2. A GRASP/VND methodology combined with Tabu Search is developed.
3. The effectiveness of our proposal is tested with real-life students from the MBA program offered at IEEM Business School, Universidad de Montevideo, Uruguay.

The document is organized in the following manner. The related work is presented in Sect. 2. A mathematical programming formulation for the MDOR is introduced in Sect. 3. A full GRASP/VND heuristic combined with Tabu Search is presented in Sect. 4. Computational results based on real-life students are presented in Sect. 5. Section 6 contains concluding remarks and trends for future work.

## 2 Related Work

We identify the closest works of ours from the scientific literature in [2,3,7]. A simplified model with a large similarity in the team formation is presented in [3], which considers the dining philosophers problem for the assignment of students into groups. In [7], the problem is modeled using integer linear programming. This work considers a centroid for each cluster. Two approaches are studied: the min-sum approach tries to minimize the distances with respect to the centroid; the second is a min-max approach whose goal is to minimize the maximum (i.e., the worst) distance.

The case-study in [2] consists of the assignment of 235 students to 8 advisors. This work considers integer linear programming, and it is equivalent to the min-sum approach given by [7]. The problem belongs to the $\mathcal{NP}$-Hard class, and heuristics are available to tackle it [10]. A hybrid Genetic Algorithm is proposed in [9]. There, the authors suggest Tabu Search combined with strategic oscilations. Independently, [12] proposed an artificial bee-workers approach. In [8], a competitive General Variable Neighborhood Search (GVNS) is also proposed. An extension of this GVNS is offered in [4], with a Skewed VNS combined with a Shaking process to better explore the search-space. The goal in the Orthogonal Regrouping Problem is to partition a given set repeatedly, in such a way that every pair is included only once in some cluster. Well known instances have been extensively treated, e.g., the Kirkman's Schoolgirl Problem and the Social Golfer Problem.

Here we introduce the MDOR problem, which is suitable to the assignment of MBA students to teams that are re-built in every term. It is worth to remark that our approach has potential applications to other scenarios, such as staffing and scheduling in workforce management [5], team formation models for collaboration [14], and team-formation algorithms for faultline minimization [1], among others.

## 3 Problem

In this section, we describe the main features of our problem, and then we present a mathematical programming formulation. A brief discussion covers particular cases, which will be considered to address the problem heuristically.

### 3.1 Problem Description

Our problem formulation requires a definition of distance between any two items. In the context of grouping MBA students, the distance between two students would represent how different they are in terms of a set of criteria (age, type of major, gender, work experience, admission test score, etc.) that the MBA Director chooses. In the case of the real-life sets used in our test, the criteria are:

- Career (subdivided in percentage of Social Sciences, Natural and Exact Sciences content).
- Score in the Admission Test.
- Residence (urban or countryside).
- Gender.
- Age.

Career is split into three attributes in $[0, 1]$ which account for the relative levels of Social Sciences, Natural and Exact Sciences. The score in the Admission Test and the Age are natural numbers, while the remaining attributes assume binary domain. Once the attributes are selected, a distance function between the different individuals $d_{ij}$ must be specified. In what follows, the normalized-Euclidean distance is considered:

$$d_{ij} = d(x^i, x^j) = \frac{\|x^i - x^j\|_2}{max_{u \neq v}\|u - v\|_2}, \tag{1}$$

where the distance between each pair of students is found by a numerical assignment to the different attributes (i.e., different coordinates). Observe that this normalization implies that $0 \leq d_{ij} \leq 1$ for all the pairs of students $i$ and $j$ with corresponding attributes $x^i$ and $x^j$.

### 3.2 Problem Formulation

Consider the following variables:

- $N$ the number of students.
- $G$ the number of teams (clusters).
- $K$ the number of attributes.
- $M$ the number of students per team: $M = \frac{N}{G}$ (if integer).
- $S$ the number of terms (clusterings).
- $d_{ij}$ the distance between the students $i$ and $j$.
- $R$ is the number of terms that any pair of students can share ($R = 1$ for a SGP instance).

Consider the set of binary decision variables $x_{igs}$, such that $x_{igs} = 1$ if and only if the student $i$ is assigned to the group $g$ in term $s$, and $x_{igs} = 0$ otherwise. We introduce the MDOR problem as the following Integer Quadratic Problem:

$$\max_{x_{igs}} \sum_{s=1}^{S} \sum_{g=1}^{G} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} x_{igs} x_{jgs}, \tag{2}$$

$$s.t. \sum_{g=1}^{G} x_{igs} = 1,\ \forall (i,s) \in \{1,\ldots,N\} \times \{1,\ldots,S\} \tag{3}$$

$$\sum_{i=1}^{N} x_{igs} = M,\ \forall (g,s) \in \{1,\ldots,G\} \times \{1,\ldots,S\} \tag{4}$$

$$\sum_{s=1}^{S} \sum_{g=1}^{G} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} x_{igs} x_{jgs} \le R,\ \forall (g,s) \in \{1,\ldots,G\} \times \{1,\ldots,S\} \tag{5}$$

$$x_{igs} \in \{0,1\}, \forall (i,g,s) \in \{1,\ldots,N\} \times \{1,\ldots,G\} \times \{1,\ldots,S\} \tag{6}$$

The goal is to maximize the diversity-sum among all clusters and clusterings, where the intra-cluster diversity is precisely the distance-sum among all the pairs of that cluster. Constraint 3 states that each student is included in a single team. Constraint 4 states that the teams have precisely $M$ students. Constraint 5 limits the number of times any pair of students can meet in different terms. Finally, Constraint 6 defines the binary domain for the decision variables.

### 3.3 Discussion

Observe that the previous MDOR model is adequate when $M = \frac{N}{G}$ is an integer. Next we comment on how to overcome this limitation and to minimize the number of repetitions as well.

**Number of Students per Group.** If $M = \frac{N}{G}$ is not an integer, we can replace Constraints 4 with a minimal variation. In fact, consider the Euclidean division: $N = G \times M + r$ for some remainder $r : 0 \leq r < G$. We can arrange $M + 1$ students in $r$ groups, and $M$ students in the remaining $G - r$ groups.

As a more general setting, pick two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ representing lower and upper-bounds on the number of students per group. Replace Constraints 4 with:

$$\sum_{g=1}^{G} x_{igs} \geq a_g, \, \forall (g, s) \in \{1, \ldots, G\} \times \{1, \ldots, S\}$$
$$\sum_{g=1}^{G} x_{igs} \leq b_g, \, \forall (g, s) \in \{1, \ldots, G\} \times \{1, \ldots, S\}.$$

**Avoiding Repetitions.** Avoiding repetitions is not always possible, depending on the parameters $G, M, S$ of a MDOR instance. Even when it is possible, no polynomial-complexity algorithm is known for the general case; variations like the SGP-completion problem are known to be NP-complete [6,13].

Let us consider a certain student, and let $w_s$ be the number of feasible peer students for him/her during the term $s$. The sequence $w_s$ satisfies the following recurrence:

$$w_1 = N - 1;$$
$$w_{i+1} = w_i - (M - 1),$$

since $M - 1$ new students are met in the last term $s = i$. A straight solution of the recurrence leads to $w_s = N - 1 - (s-1)(M-1)$. When the courses are finished we get $s = S$ and $w_S = N - 1 - (S-1)(M-1)$. Hence, if $N < (S-1)(M-1)+1$, it is impossible to avoid repetitions.

Two possible heuristic approaches arise to cope with the repetition problem. One might build high-diversity solutions while controlling the repetition level. Alternative, one might generate repetition-free solutions and then choose and/or modify them seeking for improved diversity. In this paper we introduce an algorithm that follows the first approach. A parameter *GLOBAL_REP* is set; once more than *GLOBAL_REP* times a solution is generated including a repetition for a certain pair, the algorithm accepts the repetition.

## 4 Solution

GRASP and VND are well known metaheuristics that have been successfully used to solve many hard combinatorial optimization problems. GRASP is a powerful multi-start process which operates in two phases. A feasible solution is built in a first phase, whose neighborhood is then explored in the Local Search Phase. The second phase is usually enriched by means of different variable neighborhood structures. For instance, VND explores several neighborhood structures in a deterministic order. Its success is based on the simple fact that different neighborhood structures do not usually have the same local minimum. Thus, the resulting solution is simultaneously a locally optimum solution under all the neighborhood structures. The reader is invited to consult the comprehensive Handbook of Heuristics for further information [11]. Here, we develop a GRASP/VND methodology.

## 4.1 GRASP/VND Methodology for the MDOR

We followed a traditional VND flow diagram, that consists of three local searches:

- $Insert$: moves a student to another group.
- $Swap$: swaps two students from different groups.
- $3 - Chain$: exchanges three students from three different groups.

The most simple local searches appear at the beginning. Therefore, the order is respectively $Insert$, $Swap$ and $3 - Chain$. A greedy randomized $Construction$ phase takes effect first.

To speed-up the evaluation of the objective function, the internal structures in the main algorithm consider two vectors:

- $x^c[i]$: current group for student $i$, and
- $sd^c[i][g]$: current sum-diversity between the student $i$ and his/her peers in group $g$.

Observe that $sd^c[i][g] = \sum_{j:x[j]=g} d_{i,j}$, and if we link the students in a graph with link-weights $d_{i,j}$, by Handshaking Lemma we get that the objective is:

$$f(x^c) = \frac{1}{2}\sum_{i=1}^{N} sd^c[i][x^c[i]]. \tag{7}$$

In the following, the details of the construction and local searches are presented, in the respective order.

## 4.2 Construction Phase

The search space is the set of all student assignments to the groups, where each student belongs to exactly one group. A feasible solution also meets the respective lower and upper bounds $a_g$ and $b_g$. In our $Construction$ phase, an iterative student insertion into groups takes effect, meeting the lower bounds $a_g$. Finally, in order to fulfill feasibility, all the students are assigned in some group, meeting the upper-bound $b_g$. Two factors are considered for these group-insertions: diversity and repetitions. In this construction phase, the priority is given to repetitions. Therefore, a memory with the previous terms is used, and if two assignment have identical number of repetitions, the assignment with maximum diversity is chosen. During the process, the diversity per group $g$ for some student $x$ is found using the following expression:

$$d'(x,g) = \sum_{y \in g} \frac{d(x,y)}{|g|}.$$

Observe the relation with the cardinality $|g|$; otherwise, groups with larger number of students are always preferred (Fig. 1).

**Algorithm 1** $Construction(studentGroup, a, b, atrsStandard, repMatrix)$

```
1: $studentVector \leftarrow \{1, 2, .., N\}$
2: $groupVector \leftarrow \{1, 2, .., N\}$
3: $assignOneRandomStudentToEachGroup(studentGroup, repMatrix)$
4: while $groupVector \neq \{\}$ do
5:     $selGroup \leftarrow assignGroupToStudForMinRepetitions($
6:     $studentGroup, repMatrix)$
7:     if $groupCount[selGroup] = a[selGroup]$ then
8:         $groupVector \leftarrow groupVector - selGroup$
9:     end if
10: end while
11: for $g \leftarrow 1$ to $G$ do
12:     if $groupCount[g] = b[g]$ then
13:         $groupVector \leftarrow groupVector - g$
14:     end if
15: end for
16: while $groupVector \neq \{\}$ do
17:     $selGroup \leftarrow assignGroupToStudentForMinRepetitions($
18:     $studentGroup, repMatrix)$
19:     if $groupCount[selGroup] = b[selGroup]$ then
20:         $groupVector \leftarrow groupVector - selGroup$
21:     end if
22: end while
```

**Fig. 1.** Construction phase

The following variables are considered during the *Construction* phase:

– $studentGroup[s]$: the group assigned to student $s \in \{1, \ldots, N\}$.
– $atrsStandard[i, j]$: the value of attribute $j \in \{1, \ldots, K\}$ for the student $i$.
– $groupCount[g]$: the number of students in the group $g \in \{1, \ldots, G\}$.

The following functions are also considered:

– $assignOneRandomStudentToEachGroup()$: assigns, in each group, one random student uniformly picked at random.
– $assignGroupToStudForMinRepetitions()$: picks a random student, and assigns him/her to the group that leads to the least number of repetitions. Ties are solved using the maximum diversity.

### 4.3 Insertion

In this local search, a student $i$ is moved from a different group. We remark that a local search takes place whenever the resulting solution is both better and feasible. To test feasibility, we just check the lower and upper bounds for the old and the new group, respectively. The difference in the objective is the change in the diversity:

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

being $x^n$ the new solution and $x^c$ the current solution (Fig. 2).

```
Algorithm 2 $Insertion(studentGroup, sd, solCurrent, atrsStandard, tabuMatrix)$
1: $res \leftarrow false$
2: for $i \leftarrow 1$ to $N$ do
3:     for $g \leftarrow 1$ to $G$ do
4:         if $studentGroup[i] \neq g$
5:         and $groupCount[g] < b[g]$ and
6:         $groupCount[studentGroup[i]] > a[g]$ then
7:             $diffSol \leftarrow sd[i][g] - sd[i][studentGroup[i]]$
8:             if $diffSol > 0$ and $updateTabuSearchMatrix($
9:             $i, g, studentGroup, tabuMatrix)$ then
10:                $studentGroup[i] \leftarrow g$
11:                $solCurrent \leftarrow solCurrent + diffSol$
12:                $updateSD(studentGroup, sd, i, g)$
13:                $res \leftarrow true$
14:            end if
15:        end if
16:    end for
17: end for
18: return $res$
```

**Fig. 2.** Local Search I: *Insertion*

### 4.4 Swap

In this local search, two students $i$ and $j$, originally belonging to different groups $g_i \neq g_j$, are exchanged, and the difference in the objective is:

$$f(x^n) - f(x^c) = (sd^c[i][g_j] - sd^c[i][g_i]) + (sd^c[j][g_j] - sd^c[j][g_i]) - 2d_{ij}$$

A pseudocode for *Swap* is presented in Fig. 3.

**Algorithm 3** $Swap(studentGroup, sd, solCurrent, atrsStandard, tabuMatrix)$

```
 1: $res \leftarrow false$
 2: for $i \leftarrow 1$ to $N$ do
 3:     for $j \leftarrow 1$ to $N$ do
 4:         if $studentGroup[i] \neq studentGroup[j]$ then
 5:             $diffSol \leftarrow sd[i][studentGroup[j]] + sd[j][studentGroup[i]]$
 6:             $-sd[i][studentGroup[i]] - sd[j][studentGroup[j]] - 2d_{i,j}$
 7:             if $diffSol > 0$
 8:             and $updateTabuSearchMatrix($
 9:             $i, studentGroup[j], studentGroup, tabuMatrix)$
10:             and $updateTabuSearchMatrix($
11:             $j, studentGroup[i], studentGroup, tabuMatrix)$ then
12:                 $oldI \leftarrow studentGroup[i]$
13:                 $oldJ \leftarrow studentGroup[j]$
14:                 $studentGroup[i] \leftarrow oldJ$
15:                 $studentGroup[j] \leftarrow oldI$
16:                 $updateSD(studentGroup, sd, i, studentGroup[i])$
17:                 $updateSD(studentGroup, sd, j, studentGroup[j])$
18:                 $solCurrent \leftarrow solCurrent + diffSol$
19:                 $res \leftarrow true$
20:             end if
21:         end if
22:     end for
23: end for
24: return $res$
```

**Fig. 3.** Local Search II: *Swap*

## 4.5 3-Chain

Consider three different students $i$, $j$ y $k$ belonging to three different groups $g_i$, $g_j$ and $g_k$. Student $i$ is moved to $g_j$, $j$ is moved to $g_k$ and $k$ is moved to $g_i$ (Fig. 4):

$$f(x^n) - f(x^c) = (sd^c[i][g_j] - sd^c[i][g_i]) + (sd^c[j][g_k] - sd^c[j][g_j]) + (sd^c[k][g_i] - sd^c[k][g_k]) \\ - (d_{ij} + d_{jk} + d_{ki})$$

## 4.6 Shake

In order to increase the diversity in the search-space, a shake process takes place. Consider a $k$-neighborhood of *Swap* operation, this is, an arbitrary application of $k$ swaps. *Shake* picks a $k$-neighbor, and the VND phase is re-started with the obtained solution, provided that the Tabu List allows for the shake to be done (i.e., controlling the repetitions threshold). Figure 5 presents a full pseudocode for *Shake*. In the general algorithm, $k$ starts equal to a parameter $K\_MIN$ and is increased by a second parameter $K\_STEP$ until the solution is improved or up to a third parameter $K\_MAX$.

**Algorithm 4** $3{-}Chain(studentGroup, sd, solCurrent, atrsStandard, tabuMatrix)$

```
1: $res \leftarrow false$
2: for $i \leftarrow 1$ to $N$ do
3:     for $j \leftarrow 1$ to $N$ do
4:         for $k \leftarrow 1$ to $N$ do
5:             if $studentGroup[i] \neq studentGroup[j]$
6:             and $studentGroup[j] \neq studentGroup[k]$ then
7:                 $diffSol \leftarrow sd[i][studentGroup[j]] + sd[][studentGroup[k]]$
8:                 $+sd[k][studentGroup[i]] - sd[i][studentGroup[i]]$
9:                 $-sd[j][studentGroup[j]] - sd[k][studentGroup[k]]$
10:                $-2d_{i,j} - 2d_{j,k} - 2d_{k,i}$
11:                if $diffSol > 0$
12:                and $updateTabuSearchMatrix($
13:                $i, studentGroup[j], studentGroup, tabuMatrix)$
14:                and $updateTabuSearchMatrix($
15:                $j, studentGroup[k], studentGroup, tabuMatrix)$
16:                and $updateTabuSearchMatrix($
17:                $k, studentGroup[i], studentGroup, tabuMatrix)$ then
18:                    $oldI \leftarrow studentGroup[i]$
19:                    $oldJ \leftarrow studentGroup[j]$
20:                    $oldK \leftarrow studentGroup[k]$
21:                    $studentGroup[i] \leftarrow oldJ$
22:                    $studentGroup[j] \leftarrow oldK$
23:                    $studentGroup[k] \leftarrow oldI$
24:                    $updateSD(studentGroup, sd, i, studentGroup[i])$
25:                    $updateSD(studentGroup, sd, j, studentGroup[j])$
26:                    $updateSD(studentGroup, sd, k, studentGroup[k])$
27:                    $solCurrent \leftarrow solCurrent + diffSol$
28:                    $res \leftarrow true$
29:                end if
30:            end if
31:        end for
32:    end for
33: end for
34: return $res$
```

**Fig. 4.** Local Search III: $3 - Chain$

## 4.7 Main Algorithm

The main algorithm iterates over all terms. For each one, it starts by invoking *Construction* a number of times $MAX\_TRIES$ that acts as a parameter. The most diverse solution is passed to the following step, where the following cycle is repeated a number of times $T\_MAX$ (another parameter): *Shake* - *Insertion* - *Swap* - $3 - Chain$. The best solution found (the most diverse clustering) is chosen for the term, moving on to the next one.

**Algorithm 5** $Shake(studentGroup, k, sd, solCurrent, atrsStandard, tabuMatrix)$

```
1: while $k > 0$ do
2:     $randomI \leftarrow getRandom(N)$
3:     $randomJ \leftarrow getRandom(N)$
4:     if $studentGroup[randomI] <> studentGroup[randomJ]$ then
5:         if $updateTabuSearchMatrix(randomI,$
6:         $studentGroup[randomJ], studentGroup, tabuMatrix)$and
7:         $updateTabuSearchMatrix(randomJ,$
8:         $studentGroup[randomI], studentGroup, tabuMatrix)$ then
9:             $oldI \leftarrow studentGroup[i]$
10:            $oldJ \leftarrow studentGroup[j]$
11:            $studentGroup[i] \leftarrow oldJ$
12:            $studentGroup[j] \leftarrow oldI$
13:            $updateSD(studentGroup, sd, i, studentGroup[i])$
14:            $updateSD(studentGroup, sd, j, studentGroup[j])$
15:            $k \leftarrow k - 1$
16:        end if
17:    end if
18: end while
19: $updateSolCurrent(solCurrent, sd, studentGroup$
20: return $res$
```

**Fig. 5.** Perturbation Step: *Shake*

## 5 Computational Results

We carried out a comparison between the algorithm here introduced and the manual team assignment that was done in real-life with two IEEM Business School MBA cohorts from 2014 and 2015: "MBA1314" (34 students, 6 teams) and "MBA1415" (45 students, 8 teams).

The algorithm was coded in C++ and executed in a home-PC (Intel-core i7 2.2GHz, 8GB RAM). One hundred independent iterations were run (since GRASP is a multi-start metaheuristic) and the best solution was finally returned. As a preliminary stage, an adjustment of all the parameters was performed running several experiments. $MAX\_TRIES$ and $T\_MAX$ were set to 100 and 500 respectively. The *Shake* parameters were finally set to $K\_MIN = K\_STEP = 1$ and $K\_MAX = 3$. There is a trade-off between diversity and number of repetitions. A larger freezing-factor $GLOBAL\_REP$ in the Tabu List implies a lower level of diversity as one test with MBA1415 shows in Table 1. All results next reported were obtained with Tabu-list parameter to a freezing factor of 285.000 to keep repetitions at a minimum level.

Table 2 compares the diversity achieved by our algorithm vs the manual team assignment for the two cohorts and the five terms that the program spans; Table 3 does a similar comparison for repetitions per term. Our algorithm consistently outperformed the manual assignment when considering diversity and repetitions. It also took less time, since the longest execution took 50 min, while the manual assignment was reported to take more than 4 hours for each cohort.

**Table 1.** Diversity and repetitions per term, MBA1415: manual vs algorithm.

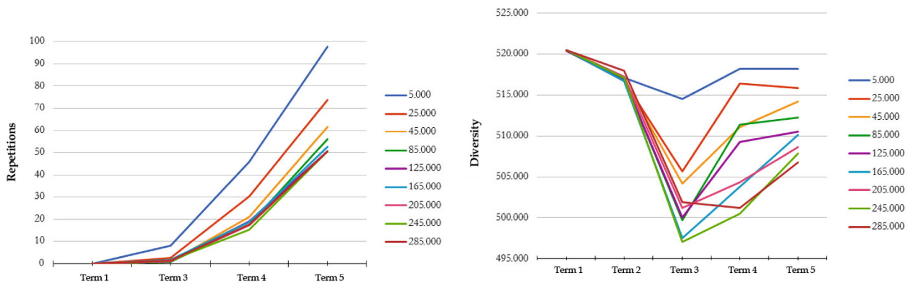


**Table 2.** Diversity per term, MBA1314 and MBA1415: manual vs algorithm.

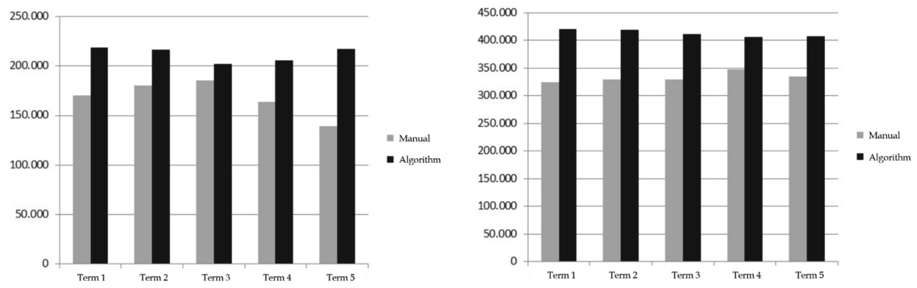
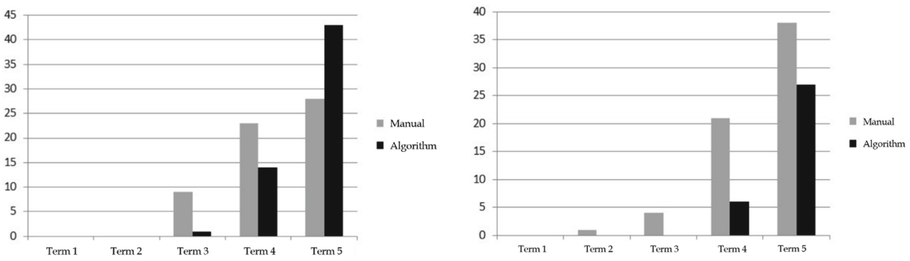


**Table 3.** Repetitions per term, MBA1314 and MBA1415: manual vs algorithm.



## 6 Conclusions and Trends for Future Work

A novel combinatorial optimization problem is introduced named Max-Diversity Orthogonal Regrouping (MDOR). It was conceived to cope with the problem of partitioning MBA cohorts into high-diversity teams, rotating the teams in every term and keeping under a given (low) threshold the repetitions. Nevertheless, the MDOR has potential applications in workforce management or team formation models for collaboration. The mathematical programming formulation is similar to a quadratic assignment problem, and the MDOR is presumably hard, even though a formal proof is not available in the literature.

A GRASP/VND methodology enriched with Tabu Search is here proposed in order to address the MDOR. A Shaking process in order to further explore the search-space is also included. The tests presented show that this algorithm produces clusterings faster, with fewer repetitions and higher diversities than the manually-built clusters applied to the real-life cohorts of the test cases. Future work includes formally establishing the computational complexity of the MDOR, and comparing our GRASP/VND methodology with alternative heuristics.

**Acknowledgements.** This work is partially supported by Project ANII FCE_1_2019 _1_156693 *Teoría y Construcción de Redes de Máxima Confiabilidad*, MATHAMSUD 19-MATH-03 *Rare events analysis in multi-component systems with dependent components* and STIC-AMSUD ACCON *Algorithms for the capacity crunch problem in optical networks*.

## References

1. Bahargam, S., Golshan, B., Lappas, T., Terzi, E.: A team-formation algorithm for faultline minimization. Expert Syst. Appl. **119**, 441–455 (2019)
2. Baker, B.M., Benn, C.: Assigning pupils to tutor groups in a comprehensive school. J. Oper. Res. Soc. **52**, 623–629 (2001)
3. Bhadurya, J., Mightyb, E.J., Damar, H.: Maximizing workforce diversity in project teams: a network flow approach. Omega **28**, 143–153 (2000)
4. Brimberg, Jack, Mladenovic, Nenad, Uroševic, Dragan: Solving the maximally diverse grouping problem by skewed general variable neighborhood search. Inf. Sci. **295**, 650–675 (2015)
5. De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E.: Workforce planning incorporating skills: state of the art. Eur. J. Oper. Res. **243**(1), 1–16 (2015)
6. Colbourn, C.J.: The complexity of completing partial Latin squares. Discret. Appl. Math. **8**(1), 25–30 (1984)
7. Desrosiers, J., Mladenović, N., Villeneuve, D.: Design of balanced MBA student teams. J. Oper. Res. Soc. **56**, 60–66 (2005)
8. Dragan, Uroševic: Variable neighborhood search for maximum diverse grouping problem. Yugoslav J. Oper. Res. **24**, 21–33 (2014)
9. Fan, Z.P., Chen, Y., Ma, J., Zeng, S.: A hybrid genetic algorithmic approach to the maximally diverse grouping problem. J. Oper. Res. Soc. **62**, 1423–1430 (2011)
10. Feo, T.A., Khellaf, M.: A class of bounded approximation algorithms for graph partitioning. Networks **20**, 181–195 (1990)
11. Mart, R., Pardalos, P.M., Mauricio, G., Resende, C.: Handbook of Heuristics, 1st edn. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-319-07124-4
12. Rodriguez, F.J., Lozano, M., García-Martínez, C., González, J.D.: An artificial bee colony algorithm for the maximally diverse grouping problem. Inf. Sci. **230**, 183–196 (2013)
13. Triska, M.: Solution methods for the social golfer problem. Inf. Sci. **295** (2008)
14. Wi, H., Seungjin, O., Mun, J., Jung, M.: A team formation model based on knowledge and collaboration. Expert Syst. Appl. **36**(5), 9121–9134 (2009)