symmetry

Article

Solving the Max-Diversity Orthogonal Regrouping Problem by an Integer Linear Programming Model and a GRASP/VND with Path-Relinking Approach

Eduardo Canale [1], Franco Robledo [1], Pablo Sartor [2,*] and Luis Stábile [1]

[1] Facultad de Ingeniería, Universidad de la República, Montevideo 11300, Uruguay; canale@fing.edu.uy (E.C.); frobledo@fing.edu.uy (F.R.); lstabile@fing.edu.uy (L.S.)
[2] IEEM Business School, Universidad de Montevideo, Montevideo 16000, Uruguay
* Correspondence: psartor@um.edu.uy

Abstract: Students from Master of Business Administration (MBA) programs are usually split into teams. In light of the generalistic nature of MBA programs, diversity within every team is desirable in terms of gender, major, age and other criteria. 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, thus training his or her adaptation skills and expanding the peer network. Achieving diverse teams while avoiding–or minimizing—the repetition of student pairs is a complex and time-consuming task for MBA Directors. We introduce the Max-Diversity Orthogonal Regrouping (MDOR) problem to manage the challenge of splitting a group of people into teams several times, pursuing the goals of high diversity and few repetitions. We propose a hybrid Greedy Randomized Adaptive Search Procedure/Variable Neighborhood Descent (GRASP/VND) heuristic combined with tabu search and path relinking for its resolution, as well as an Integer Linear Programming (ILP) formulation. We compare both approaches through a set of real MBA cohorts, and the results show that, in all cases, the heuristic approach significantly outperforms the ILP and manually formed teams in terms of both diversity and repetition levels.

Keywords: MBA teams; orthogonal regrouping; diversity; GRASP; VND; path relinking

check for updates

Citation: Canale, E.; Robledo, F.; Sartor, P.; Stábile, L. Solving the Max-Diversity Orthogonal Regrouping Problem by an Integer Linear Programming Model and a GRASP/VND with Path-Relinking Approach. *Symmetry* **2022**, *14*, 18. https://doi.org/10.3390/sym14010018

Academic Editors: Angelo Sifaleras, Andrei Sleptchenko, Adriana Felicia Gabor and Laszlo T. Koczy

Received: 15 October 2021
Accepted: 14 December 2021
Published: 23 December 2021

1. Motivation

There is empirical evidence showing that diversity among team members plays an important role in the success of organizations; see [1–3] for studies on the impacts of gender diversity on revenue and profits, female executive ratios on profit, and educational and work ratios on innovation. Collaborative, multidisciplinary team-formation and staffing/scheduling problems in workforce management are of paramount importance in project deployment and large-scale corporations. Diversity within teams is also increasingly important in contexts such as squadrons in "agile organizations" and study groups in executive education programs. In fact, the case that motivated the work presented in this article on MBA team formation and rotation is a good example. Experience shows that student skills and the learning process benefit significantly from highly diverse teams with respect to prior experience, age, gender, major, years of work experience and other attributes. The main goal of MBA programs is to develop generalistic professionals able to cope with the complexity of managing organizations. They are generalistic by nature; students benefit much more from addressing their weaknesses regarding skills, experience and discipline knowledge than from further developing those that they have already mastered. An important part of learning achieved by every student comes from his or her peers through teamwork. In addition, developing abilities to work in diverse teams and quickly adapt oneself to them is also a goal in itself of most current MBA programs. The problem of splitting a group of people into a given number of teams in a way that maximizes the intra-team diversity is challenging, as discussed in Section 2. Moreover, in many contexts, it can be desirable to rotate the teams several times so that every team member meets as many peers as possible, trains his or her adaptation skills, etc. Continuing with the case of MBA programs, they are usually split into 4–6 terms. Many business schools rotate the groups every term so that students develop their ability to adapt to different groups, benefit from new points of view and expand their peer network. Creating highly diverse teams while minimizing the repetition of peer pairs between terms is a very challenging problem faced by program directors when launching a new cohort. For instance, at IEEM Business School (Universidad de Montevideo), MBA directors typically spend 6–8 hours coping with this problem, as many times as cohorts are launched every year, yet achieving team partitions has been barely satisfactory in meeting the goals of high diversity and few repetitions.

Given the intrinsic difficulty of the underlying partitioning and clustering problems involved, it is highly convenient to develop algorithms to automate this task as much as possible. In this work, we focused on maximum-diversity regrouping assignments of MBA students; nevertheless, the reader can find potential applications to similar partitioning problems that involve rotating the partitions several times.

The contributions of this article can be summarized as follows:

1. We introduce the Max-Diversity Orthogonal Regrouping (MDOR) problem.
2. A feasibility condition and an upper bound for the optimum are derived in Section 3.3 for some ranges of the parameters.
3. An exact Integer Linear Programming (ILP) formulation for the MDOR problem is proposed (Section 3.2).
4. A GRASP/VND methodology combined with tabu search and path relinking is proposed.
5. We compare the performance of both approaches against each other and against a real case of manual team splitting using data from five MBA cohorts of IEEM Business School, Universidad de Montevideo, Uruguay, who graduated between 2017 and 2019.

This document is organized as follows. The related work is presented in Section 2. A mathematical programming formulation for the MDOR is introduced in Section 3. A full GRASP/VND heuristic combined with tabu search and path relinking as a post-optimization technique is presented in Section 4. Computational results based on real-life students are presented in Section 5. Section 6 contains concluding remarks and directions for future work.

2. Related Work

Based on our scientific literature review, the works that are closest to ours are [4–6]. A simplified model with a high similarity in team formation was presented in [4], which considers the dining philosophers problem for the assignment of students into groups. In [5], the problem was modeled using integer linear programming. This work considered a centroid for each cluster. Two approaches were 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 [6] consisted of the assignment of 8 advisors to 235 students. This work applied integer linear programming, and it is equivalent to the min-sum approach given by [5]. The problem belongs to the $\mathcal{NP}$-hard computational complexity class, and heuristics are available to tackle it [7]. A hybrid genetic algorithm was proposed in [8]. There, the authors suggested Tabu Search combined with strategic oscillations. Independently, [9] proposed an artificial bee-workers approach. In [10], a competitive General Variable Neighborhood Search (GVNS) was also proposed. An extension of this GVNS was developed in [11], with a skewed VNS combined with a shaking process to better explore the search space. The goal in the Orthogonal Regrouping Problem is to repeatedly partition a given set in such a way that every pair is included only once in a cluster. Well-known instances have been extensively treated, e.g., the Kirkman's Schoolgirl Problem and the Social Golfer Problem.

This article is an extended version of [12], where we introduced the MDOR problem. It is worth remarking that, although this work was motivated by the assignment of MBA students to teams that are reconstructed every term, it has potential applications to other scenarios, such as staffing and scheduling in workforce management [13], team formation models for collaboration [14] and team-formation algorithms for faultline minimization [15], among others.

3. Problem

In this section, we describe the main features of the problem, and then we present a mathematical programming formulation. A brief discussion covers particular cases, which are 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 represents 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 into percentages of Social, Natural and Exact Sciences content);
- Admission test score;
- Residence (urban or countryside);
- Gender;
- Age.

Career is split into three attributes in $[0,1]$ that account for the relative levels of Social, Natural and Exact Sciences. The score on the admission test and age are natural numbers, while the remaining attributes assume a 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). Note that this normalization implies that $0 \leq d_{ij} \leq 1$ for all 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 students i and j;
- R: the maximum number of terms that any pair of students can share (R = 1 for an SGP instance).

Consider the set of binary decision variables x_{igs}, such that $x_{igs} = 1$ if and only if student i is assigned to group g in term s, and $x_{igs} = 0$ otherwise. Additionally, we consider the set of binary decision variables y_{ij}^{gs} such that $y_{ij}^{gs} = 1$ if and only if $x_{igs} = 1$ and $x_{jgs} = 1$, where i and j are students, g is a group and s is a term. We introduce the MDOR problem as the following integer linear programming model:

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

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

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

$$2y_{ij}^{gs} \le x_{igs} + x_{jgs}, \forall (i,j,g,s) \in \times \{1,\ldots,N\}^2 \times \{1,\ldots,G\} \times \{1,\ldots,S\} \tag{5}$$

$$x_{igs} + x_{jgs} \le y_{ij}^{gs} + 1, \forall (i,j,g,s) \in \{1,\ldots,N\}^2 \times \{1,\ldots,G\} \times \{1,\ldots,S\} \tag{6}$$

$$\sum_{g=1}^{G} \sum_{s=1}^{S} y_{ij}^{gs} \le R, \forall (i,j) \in \{1,\ldots,N\} \times \{1,\ldots,N\} \tag{7}$$

$$x_{igs} \in \{0,1\}, \forall (i,g,s) \in \{1,\ldots,N\} \times \{1,\ldots,G\} \times \{1,\ldots,S\} \tag{8}$$

$$y_{ij}^{gs} \in \{0,1\}, \forall (i,j,g,s) \in \{1,\ldots,N\}^2 \times \{1,\ldots,G\} \times \{1,\ldots,S\} \tag{9}$$

The goal (objective function (2)) is to maximize the diversity-sum of all student peers i, j among all clusters (second summation) and terms (first summation), where the intracluster diversity is precisely the distance-sum among all pairs of that cluster. Constraints (3) state that each student is included in one and just one team for all clusters and terms. Constraints (4) state that the teams have exactly M students for all teams and terms. Constraints (5) and (6) enforce the coherence between the x and y sets of variables as follows. Constraints (5) state that if two students i and j share group g in a certain term s, i.e., $y_{ijgs} = 1$, then both flag variables x_{igs} and x_{jgs} must be equal to one. Conversely, Constraints (6) state that if x_{igs} and x_{jgs} are both one, meaning that they share team g in term s, then y_{ijgs} must be equal to one. Constraints (7) limit the number of times that any pair of students i, j can meet throughout the different terms (outer summation). Finally, Constraints (8) and (9) define the binary domain for the decision variable sets x and y.

3.3. Discussion

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

3.3.1. Number of Students per Group

If $M = \frac{N}{G}$ is not an integer, we can introduce a slight change in Constraints (4). In fact, consider the Euclidean division: $N = G \times M + r$ for some remainder $r : 0 \le r < G$. 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 $\vec{a}$ and $\vec{b}$ representing lower and upper bounds on the number of students per group. Replace Constraints (4) with:

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

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

3.3.2. Avoiding Repetitions

Completely avoiding repetitions is not always possible, depending on the parameters G, M and S of an MDOR instance (see Table 1). Even when it is possible, no polynomial-complexity algorithm is known for the general case; variations such as the SGP-completion problem are known to be NP-complete [16,17], so no feasible solution is guaranteed to be computed in polynomial time unless $S = 1$. In the last case, any distribution is a feasible solution, but the optimization problem is equivalent to the MIN CUT INTO BOUNDED SET problem [18], where sets have bounds equal to M and edge weights equal to $d_{i,j}$, which is NP-complete on N and G, since maximizing the edge weights inside the groups is the same as minimizing the edge weights between these groups. It is then plausible to assume that MDOR is NP-hard, but we were not able to prove it. If so, it would be NP-complete since checking if a solution is feasible and if its cost is greater than a given constant takes polynomial time.

Table 1. GRASP/VND with path relinking versus ILP for *MDOR*.

Instances					**Heuristic**			**ILP**		**GAP**
Name	**N**	**S**	**G**	**R**	$f_h(x)$	**Time (h)**	**GAP$_i$ (%)**	$f_i(x)$	**Time (h)**	**%**
2017-1	31	1	5	1	43.5	11.7	202	42.5	24	2.43
2017-1	31	6	5	3	256.2	24	171	245.2	24	4.45
2017-1	31	6	6	2	198.4	24	123	198.4	24	0
2017-1	31	6	6	3	209.3	24	249	190.0	24	10.17
2017-1	31	6	7	2	170.4	24	176	162.1	24	5.12
2017-1	31	6	7	3	179.7	24	315	160.2	24	12.21
2018-1	47	1	5	1	107.3	24	293	104.6	24	2.58
2018-1	47	6	6	3	514.6	24	234	479.8	24	7.24
2018-1	47	6	7	2	408.1	24	162	408.0	24	0.02
2018-1	47	6	7	3	437.6	24	288	413.1	24	5.92
2018-1	47	6	8	2	351.1	24	209	346.2	24	1.39
2018-1	47	6	8	3	377.5	24	356	352.2	24	7.16
2018-2	35	1	5	1	52.6	21.8	223	52.1	24	0.99
2018-2	35	6	6	2	242.7	24	128	239.4	24	1.40
2018-2	35	6	6	3	260.8	24	230	247.9	24	5.23
2019-1	40	1	5	1	77.8	19.1	267	77.0	24	0.99
2019-1	40	6	5	3	452.2	24	167	447.8	24	0.99
2019-1	40	6	6	2	357.3	24	123	357.2	24	0.02
2019-1	40	6	6	3	381.8	24	237	356.0	24	7.24
2019-1	40	6	7	2	301.7	24	167	298.7	24	0.98
2019-1	40	6	7	3	324.0	24	304	296.8	24	9.16
2019-2	30	1	5	1	41.1	11.3	226	40.9	24	0.50
2019-2	30	6	5	3	241.2	24	188	223.9	24	7.70
2019-2	30	6	6	2			No feasible solution			
2019-2	30	6	6	3	195.9	24	258	180.1	24	8.79
2020-1	42	1	7	1	58.2	24	474	57.3	24	1.57
2020-1	42	6	7	2			Time limit exceeded			
2020-1	42	6	7	3	337.5	24	651	323.62	24	4.29
Average										4.17

Let us consider a certain student. If $R = 1$ in each term s, he/she will meet $M - 1$ different students, so, among the S terms, he/she will meet $S(M - 1)$ students. In order to fulfill this restriction, the total number of students should be greater than or equal to $1 + S(M - 1)$, i.e., $N \geq 1 + S(M - 1)$.

Let us also upper bound the value of the objective function given in Equation (2), assuming that N/G is an integer and $R = 1$, by sorting the distances d_{ij}. More concretely, let $j_{i,h}$ be a matrix, where every row i is a vector that contains the indexes of $N - 1$ students (excluding i) sorted by descending distance to i, i.e,

$$d_{i,j_{i,h}} \geq d_{i,j_{i,h+1}} \qquad h = 1, \ldots, N - 1.$$

Then,

$$F = \sum_{s=1}^{S}\sum_{g=1}^{G}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}y_{ij}^{gs} = \frac{1}{2}\sum_{s=1}^{S}\sum_{g=1}^{G}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}y_{ij}^{gs} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j:y_{ij}^{gs}=1} d_{ij}$$

$$\leq \frac{1}{2}\sum_{i=1}^{N}\sum_{h=1}^{S(M-1)} d_{ij_{i,h}},$$

which is an upper bound that can be easily precomputed, which can help to estimate the gap to the optimum. If $R > 1$ and $S(M-1) = qR + R'$ with $0 \leq R' < R$, then

$$F \leq \frac{1}{2}\sum_{i=1}^{N}\left(R'd_{ij_{i,q+1}} + R\sum_{h=1}^{q} d_{ij_{i,h}}\right).$$

Two possible heuristic approaches arise to cope with the repetition problem. One might build high-diversity solutions while controlling the repetition level. Alternatively, one might generate repetition-free solutions and then choose and/or modify them to seek for improved diversity. In this paper, we introduce an algorithm that follows the first approach. A parameter $GLOBAL\_REP$ is set; if the algorithm generates solutions that include a repetition for a certain pair more than $GLOBAL\_REP$ times, then it accepts the algorithm.

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 multistart metaheuristic that operates in two phases. A feasible solution is generated in the 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 neighborhood structures. Additionally, the metaheuristic is enriched with path relinking, which is an enhancement of the GRASP procedure, leading to significant improvements in solution quality. Path relinking was first introduced in the context of tabu sarch [19]. It was suggested as an approach to integrate intensification and diversification into the search for solutions. In the context of GRASP, path relinking was first introduced by [20]. This approach generates new solutions by exploring trajectories between high-quality solutions; it starts from one of these solutions, called the *initiating solution*, and generates a path in the neighborhood space that leads towards other solutions, the *guiding solutions*. The reader is invited to consult the comprehensive *Handbook of Heuristics* [21] and *GRASP with Path-Relinking* [22] for further information. Here, we develop a GRASP/VND with path-relinking methodology.

4.1. GRASP/VND with Path-Relinking Methodology for the MDOR

We followed a traditional VND flow diagram, which 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 is run first.

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

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

Note 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 virtue of the Handshaking Lemma, we obtain the following objective:

$$f(x^c) = \frac{1}{2}\sum_{i=1}^{N} sd^c[i][x^c[i]]. \tag{10}$$

Next, the details of the construction and local searches are presented.

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 satisfies 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 realize feasibility, all students are assigned to some group, satisfying 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 of the previous terms is used, and if two assignment have an identical number of repetitions, the assignment with maximum diversity is chosen. During the process, the diversity per group g for some student x is computed using the following expression:

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

Note the division by the cardinality $|g|$; this is meant to avoid preferring groups with larger numbers of students. Algorithm 1 illustrates the pseudocode of the construction phase.

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

```
1: $studentVector \leftarrow \{1,2,\ldots,N\}$
2: $groupVector \leftarrow \{1,2,\ldots,N\}$
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
```

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 student i;
- $groupCount[g]$: the number of students in group $g \in \{1,\ldots,G\}$.

The following functions are also considered:

- $assignOneRandomStudentToEachGroup()$: it assigns one random student uniformly picked at random to each group.
- $assignGroupToStudForMinRepetitions()$: it picks a random student and assigns him/her to the group that leads to the least number of repetitions. Ties are resolved based on maximum diversity.

4.3. Insertion

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

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

where x^n is the new solution, and x^c is the current solution.

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 function is:

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

4.5. Three-Chain

Consider three different students i, j and 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:

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

4.6. Shake

In order to increase the diversity in the search space, a shake process takes place. Consider a k-neighborhood of a Swap operation, i.e., an arbitrary application of k swaps. *Shake* picks a k-neighbor, and the VND phase is restarted with the obtained solution, provided that the tabu list allows for the shake to be carried out (i.e., controlling the repetition threshold). In the general algorithm, k starts as equal to a parameter K_MIN and is increased by a second parameter K_STEP until the solution is improved or until it equals a third parameter K_MAX.

4.7. Path Relinking

To perform the post-optimization path-relinking process, a neighborhood structure is defined. Two solutions x_1 and x_2 are neighbors if there exists a sequence of swaps between students that keeps the number of global repetitions under the threshold R. Which students to take in each swap is determined based on the symmetric difference between the two solutions. The paths are developed using this neighborhood structure. Starting from one or more elite solutions, paths in the solution space leading toward other elite solutions are generated and explored in the search for better solutions.

Algorithm 2 illustrates the pseudocode of the path-relinking procedure applied to a pair of solutions x_s (starting solution) and x_t (target solution).

Algorithm 2 $Path\_Relinking(x_s, x_t)$

```
1:  Compute symmetric difference $\Delta(x_s; x_t)$;
2:  $f^* \leftarrow \max\{f(x_s), f(x_t)\}$;
3:  $x^* \leftarrow \text{argmax}\{f(x_s), f(x_t)\}$;
4:  $x \leftarrow x_s$;
5:  while $\Delta(x; x_t) \neq \emptyset$ do
6:    $m^* \leftarrow argmax\{f(x \oplus m) : m \in \Delta(x, x_t)\}$;
7:    while $maxNumberOfRepetitions(x \oplus m^*) > R$ do
8:      $m \leftarrow getRandom(\Delta(x, x_t))$
9:      $\Delta(x \oplus m, x_t) \leftarrow \Delta(x, x_t) \setminus \{m\}$
10:     $x \leftarrow x \oplus m$
11:     $m^* \leftarrow argmax\{f(x \oplus m) : m \in \Delta(x, x_t)\}$;
12:   end while
13:   $\Delta(x \oplus m^*, x_t) \leftarrow \Delta(x, x_t) \setminus \{m^*\}$
14:   $x \leftarrow x \oplus m^*$
15:   if $f(x) > f^*$ then
16:     $f^* \leftarrow f(x)$
17:     $x^* \leftarrow x$
18:   end if
19: end while
20: return $x^*$
```

The procedure starts by computing the symmetric difference $\Delta(x_s, x_t)$ between the two solutions; in this case, this represents the set of moves (swaps) needed to reach x_t (target solution) from x_s (initial solution). A path of solutions is generated linking x_s and x_t. The best solution x^* in this path is returned by the algorithm. At each step, the procedure examines all moves $m \in \Delta(x; x_t)$ from the current solution x and selects the one that results in the maximum diversity under the threshold R. If there is no swap that keeps the number of repetitions within the threshold, it applies one swap from $\Delta(x, x_t)$ at random and repeats the procedure. If necessary, the best solution x^* is updated. The procedure terminates when x_t is reached, which is when $\Delta(x, x_t) = \emptyset$.

4.8. Main Algorithm

The main algorithm iterates over all terms. For each one, it starts by invoking Construction for a number of times equal to MAX_TRIES, which acts as a parameter. The most diverse solution is passed to the following step, where the following cycle is repeated for a number of times equal to T_MAX (another parameter): *Shake*, Insertion, Swap and 3-Chain. Then, the algorithm selects a set $\mathcal{Y}$ of instances, chosen randomly from the pool of elite solutions, for the post-optimization path-relinking process [19]. The best solution found is chosen for the term, and the process moves to the next one.

Algorithm 3 illustrates the pseudocode of the main algorithm.

Algorithm 3 $Heuristic(MAX\_TRIES, T\_MAX, K\_MIN, K\_STEP, K\_MAX)$

```
 1: $f_h^* \leftarrow 0$
 2: for $iter \leftarrow 1$ to $MAX\_TRIES$ do
 3:   $P \leftarrow \emptyset$ {Elite solution set}
 4:   $\mathcal{T} \leftarrow \emptyset$ {Tabu List}
 5:   for $i \leftarrow 1$ to $T\_MAX$ do
 6:     $x \leftarrow GreedyRandomizedConstructionPhase()$
 7:     $x \leftarrow LocalSearchPhase(x, \mathcal{T}, K\_MIN, K\_STEP, K\_MAX)$
 8:     $\mathcal{T} \leftarrow UpdateTabuList(x, \mathcal{T})$
 9:     if $i \geq 2$ then
10:       Choose, at random, pool solutions $\mathcal{Y} \subseteq P$ to relink with $x$
11:       for $y \in \mathcal{Y}$ do
12:         Determine which ($x$ or $y$) is the initial $x_s$ and which is the target $x_t$
13:         $x_p \leftarrow PathRelinking(x_s, x_t)$
14:         Update the elite set $P$ with $x_p$
15:         if $f(x_p) > f_h^*$ then
16:           $f_h^* \leftarrow f(x_p)$
17:           $x^* \leftarrow x_p$
18:         end if
19:       end for
20:     end if
21:   end for
22: end for
23: return $x^*$
```

5. Computational Results

We carried out a comparison between the algorithm (coded in Python 3.9) introduced herein and the ILP model implemented using the optimization engine IBM CPLEX version 12.8. Both were executed on a home PC (Intel-core i7 2.2 GHz, 8 GB RAM). One hundred independent iterations were run (since GRASP is a multistart metaheuristic), and the best solution was finally returned. As a preliminary stage, an adjustment of all parameters was performed by running several experiments. Some instances were included even though there were no feasible solutions in order to show that the existence depends on the parameters, as stated in Section 3.3.

Table 1 reports the performance of the GRASP/VND with path relinking and ILP for each instance. The heuristic was tested with the following parameters: $MAX\_TRIES = 100$, $T\_MAX = 10^6$, $K\_MIN = K\_STEP = 1$ and $K\_MAX = 3$.

Following the terminology, $f_h(x)$ and $f_i(x)$ represent the values of the best solutions found using the heuristic and the integer linear programming model, respectively. The column *Time* indicates the CPU time in seconds that the algorithms runs. The column *Time* under *ILP* gives the time required to reach either the optimum value or the best lower bound when the optimum is not attained. The elapsed time is 24 h. Column GAP_i indicates the gap calculated by CPLEX. In essence, it is the best bound, as the best objective value, that an integer solution could potentially have based on information that the solver has discovered so far. The best bound is the best relaxed-but-region-constrained solution for any region that has not yet been eliminated from the search space. The column under GAP shows the result of computing the gap with respect to the best solution found with ILP ($\text{GAP} = \frac{f_h(x) - f_i(x)}{f_i(x)} \times 100$).

In the GAP column, the reader can observe that our heuristic finds better solutions than the ILP in almost all cases. Even though a global optimum is not formally proved for some instances, the null gap between ILP and our solution reinforces the evidence of optimality. The average GAP obtained is 4.28%.

To understand the global effectiveness of our VND scheme with path relinking, a *midpoint test* was performed. Table 2 shows the performance of the algorithm for the *MDOR*. The columns Insertion, Swap, 3-Chain and Path Relinking show the percentage of each kind of movement applied over 100 executions of the VND local search phase and path relinking, respectively. The column #Moves states the average number of moves applied during these iterations. The column entitled mp shows the relative improvement in the objective function between the best solution found in each local search phase with the post-optimization path-relinking process compared to the feasible solution obtained from the construction phase over 100 iterations. The reader can see that the VND with the path-relinking effect is noticeable, since the diversity is roughly half the optimum in most cases using only the construction phase. Note that each movement is applied only if an improvement is achieved. It is clear that Insertion, Swap and Path Relinking are the most effective movements, while 3-Chain seldom has an effect. The effectiveness of path relinking is evident and reaches over 25% of the total number of applied movements in the post-optimization process.

The six test cohorts were manually split into teams by the corresponding program directors at IEEM Business School when each one was launched (between 2017 and 2020). They claim to have spent between 6 and 10 h working on a spreadsheet for every cohort. Table 3 compares what they achieved to the output of the heuristic algorithm. Columns f and R stand for the objective function and the maximum repetition parameter for both the heuristic and manually developed solutions. The comparison shows that, in all cases, the heuristic generated superior solutions in terms of objective function (the global diversity score) and equal or better solutions in terms of repetition levels; furthermore, by using the algorithm, directors can save almost all of the mentioned time that they spent on this task.

Table 2. Performance of the local search phase and path relinking. Abbreviations: I = Insertion, S = Swap, 3C = 3-Chain, PR = Path Relinking, M = #Moves.

Instances					**Heuristic, all in %**					
Name	**N**	**S**	**G**	**R**	**I**	**S**	**3C**	**PR**	**M**	mp
2017-1	31	1	5	1	53	15	0	32	21	73
2017-1	31	6	5	3	25	29	2	44	137	43
2017-1	31	6	6	2	61	23	0	16	119	52
2017-1	31	6	6	3	29	13	0	68	91	36
2017-1	31	6	7	2	13	46	1	40	97	58
2017-1	31	6	7	3	45	32	0	23	114	42
2018-1	47	1	5	1	76	12	0	12	51	23
2018-1	47	6	6	3	61	22	0	17	276	74
2018-1	47	6	7	2	22	65	1	12	204	32
2018-1	47	6	7	3	21	14	4	61	197	88
2018-1	47	6	8	2	12	52	1	35	186	21
2018-1	47	6	8	3	11	57	0	32	202	62
2018-2	35	1	5	1	61	22	0	27	31	52
2018-2	35	6	6	2	19	57	1	23	131	78
2018-2	35	6	6	3	29	58	2	11	169	85
2019-1	40	1	5	1	63	32	0	5	32	17
2019-1	40	6	5	3	68	9	3	20	178	46
2019-1	40	6	6	2	61	35	0	4	177	59
2019-1	40	6	6	3	45	27	0	28	132	87
2019-1	40	6	7	2	12	79	0	9	253	86
2019-1	40	6	7	3	31	45	0	24	101	49
2019-2	30	1	5	1	78	19	0	3	24	62
2019-2	30	6	5	3	23	61	1	15	117	59
2019-2	30	6	6	3	46	12	0	52	99	64
2020-1	42	1	7	1	23	38	2	37	44	64
2020-1	42	6	7	3	57	16	0	27	208	46
Average					40.1	34.2	0.7	26	130.4	56.1

Table 3. Performance comparison: heuristic vs. manually generated solutions.

Instances			**Manually Generated**		**Heuristic**	
Name	**G**	**N**	$f(x)$	R	$f(x)$	R
2017-1	5	31	229.8	4	256.2	3
2018-1	8	47	361.1	3	377.5	3
2018-2	6	35	243.7	3	260.8	3
2019-1	6	40	354.9	4	381.8	3
2019-2	5	30	227.0	3	241.2	3
2020-1	7	42	301.2	3	337.5	3

6. Conclusions and Directions for Future Work

A combinatorial optimization problem named Max-Diversity Orthogonal Regrouping (MDOR) is introduced. It was conceived to cope with the problem of partitioning MBA cohorts into high-diversity teams, rotating the teams every term and keeping repetitions under a given (low) threshold. Nevertheless, the MDOR has potential applications in workforce management or team formation models for collaboration.

An exact integer linear programming method and a GRASP/VND methodology enriched with path relinking are proposed in order to address the MDOR. The tests presented, based on six real MBA cohorts, show that the heuristic algorithm produces partitionings faster and with fewer repetitions and higher diversities than the best solution found for the ILP exact method in almost all cases. The heuristic algorithm produced solutions that were significantly better than those manually generated by the program directors in terms of maximum repetitions and global diversity score. In fact, the partitionings obtained via the GRASP algorithm were presented to the MBA Directors, who agreed that they would have been far superior to those that they had manually generated, which led them to start applying the GRASP/VND algorithm from 2021 onward.

Future work includes:

- Weighing the diversity differently in different terms. In the MBA example, diversity is more important in earlier terms than in later ones, since as long as the program advances, students tend to increase their field-specific knowledge and experience; towards the end of the program, it should be much more difficult to guess what their major or previous experience is based only on the attendance of a class;
- Considering the fact that a student can join the MBA in a term other than the first one (e.g., deferrals from previous cohorts) or that a student can leave the program before completing it;
- Allowing the algorithm to run starting at a term other than the first one, taking into account the previous teams for further repetitions and allowing a different number of teams to be set;
- Allowing rules such as "student A and B cannot (or must) be together in terms X, Y, Z";
- Testing a different approach to define the problem from the one presented herein (maximizing diversity while controlling for repetitions); this could be (a) minimizing repetitions while controlling for diversity or (b) maximizing an objective function that simultaneously considers diversity and repetitions through a well-tuned relative weighting for both factors;
- Exploring the potential benefits of other heuristic approaches, e.g., adaptive VND [23].

Author Contributions: Conceptualization, P.S.; methodology, F.R. and E.C.; software, L.S.; validation, P.S.; formal analysis, F.R. and E.C.; data curation, P.S.; writing, P.S. and L.S.; visualization, L.S.; project administration, F.R.; funding acquisition, P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The MBA cohort datasets used in our tests are available at: http://www2.um.edu.uy/ieem-papers/mba-groups.zip (accessed on 17 December 2021).

Acknowledgments: 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*, STIC-AMSUD ACCON *Algorithms for the capacity crunch problem in optical networks and the Grant-Thornton Research Center of IEEM Business School*.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Herring, C. Does Diversity Pay?: Race, Gender, and the Business Case for Diversity. *Am. Sociol. Rev.* **2009**, *74*, 208–224. [CrossRef]
2. Ellemers, N.; Rink, F. Diversity in work groups. *Curr. Opin. Psychol.* **2016**, *11*, 49–53. [CrossRef]
3. Talke, K.; Salomo, S.; Kock, A. Top Management Team Diversity and Strategic Innovation Orientation: The Relationship and Consequences for Innovativeness and Performance. *J. Prod. Innov. Manag.* **2011**, *28*, 819–832. [CrossRef]
4. Bhadurya, J.; Mightyb, E.J.; Damar, H. Maximizing workforce diversity in project teams: a network flow approach. *Omega* **2000**, *28*, 143–153. [CrossRef]
5. Desrosiers, J.; Mladenović, N.; Villeneuve, D. Design of balanced MBA student teams. *J. Oper. Res. Soc.* **2005**, *56*, 60–66. [CrossRef]
6. Baker, B.M.; Benn, C. Assigning pupils to tutor groups in a comprehensive school. *J. Oper. Res. Soc.* **2001**, *62*, 623–629. [CrossRef]
7. Feo, T.A.; Khellaf, M. A class of bounded approximation algorithms for graph partitioning. *Networks* **1990**, *20*, 181–195. [CrossRef]
8. Fan, Z.P.; Chen, Y.; Ma, J.; Zeng, S. A hybrid genetic algorithmic approach to the maximally diverse grouping problem. *J. Oper. Res. Soc.* **2011**, *62*, 1423–1430. [CrossRef]
9. Rodriguez, F.J.; Lozano, M.; García-Martínez, C.; González, J.D. An artificial bee colony algorithm for the maximally diverse grouping problem. *Inf. Sci.* **2013**, *230*, 183–196. [CrossRef]
10. Dragan, U. Variable neighborhood search for maximum diverse grouping problem. *Yugosl. J. Oper. Res.* **2014**, *24*, 21–23.
11. Brimberg, J.; Mladenovic, N.; Uroševic, D. Solving the maximally diverse grouping problem by skewed general variable neighborhood search. *Inf. Sci.* **2015**, *295*, 650–675. [CrossRef]
12. Banchero, M.; Robledo, F.; Romero, P.; Sartor, P.; Servetti, C. Max-Diversity Orthogonal Regrouping of MBA Students Using a GRASP/VND Heuristic. In Proceedings of the ICVNS 2021, Abu Dhabi, United Arab Emirates, 21–25 March 2021; pp. 58–70.
13. Bruecker, P.D.; den Bergh, J.V.; Beliën, J.; Demeulemeester, E. Workforce planning incorporating skills: State of the art. *Eur. J. Oper. Res.* **2015**, *243*, 1–16. [CrossRef]
14. Wi, H.; Oh, S.; Mun, J.; Jung, M. A team formation model based on knowledge and collaboration. *Expert Syst. Appl.* **2009**, *36*, 9121–9134. [CrossRef]
15. Bahargam, S.; Golshan, B.; Lappas, T.; Terzi, E. A team-formation algorithm for faultline minimization. *Expert Syst. Appl.* **2019**, *119*, 441–455. [CrossRef]
16. Triska, M. Solution Methods for the Social Golfer Problem. Master's Thesis, Technische Universität Wien, Vienna, Austria, 2008.
17. Colbourn, C.J. The complexity of completing partial Latin squares. *Discret. Appl. Math.* **1984**, *8*, 25–30. [CrossRef]
18. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*, 1st ed.; W. H. Freeman: New York, NY, USA, 1979.
19. Glover, F. Tabu search and adaptive memory programming advances, applications and challenges. In *Interfaces in Computer Science and Operations Research*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 1–75.
20. Laguna, M.; Marti, R. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. Comput.* **1999**, *11*, 44–52. [CrossRef]
21. Mart, R.; Pardalos, P.M.; Resende, M.G.C. *Handbook of Heuristics*, 1st ed.; Springer Publishing Company: Berlin/Heidelberg, Germany, 2018.
22. Resendel, M.G.; Ribeiro, C.C. GRASP with path-relinking: Recent advances and applications. In *Metaheuristics: Progress as Real Problem Solvers*; Ibaraki, T., Nonobe, K., Yagiura, M., Eds.; Operations Research/Computer Science Interfaces Series; Springer: Boston, MA, USA, 2005; Volume 32, pp. 29–63.
23. Karakostas, P.; Sifaleras, A.; Georgiadis, M.C. Variable neighborhood search-based solution methods for the pollution location-inventory-routing problem. *Optim. Lett.* **2020**. [CrossRef]