Filomat 33:9 (2019), 2875–2891
https://doi.org/10.2298/FIL1909875D

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: `http://www.pmf.ni.ac.rs/filomat`

Solving Balanced Multi-Weighted Attribute Set Partitioning Problem with Variable Neighborhood Search

Dušan Džamić[a,b], Bojana Ćendić[c], Miroslav Marić[b], Aleksandar Đenić[b]

[a]*Faculty of Organizational Sciences, University of Belgrade*
[b]*Faculty of Mathematics, University of Belgrade*
[c]*Belgrade Business School - Higher Education Institution for Applied Studies*

Abstract. This paper considers the Balanced Multi-Weighted Attribute Set Partitioning (BMWASP) problem which requires finding a partition of a given set of objects with multiple weighted attributes into a certain number of groups so that each attribute is evenly distributed amongst the groups. Our approach is to define an appropriate criterion allowing to compare the degree of deviation from the "perfect balance" for different partitions and then produce the partition that minimizes this criterion. We have proposed a mathematical model for the BMWASP and its mixed-integer linear reformulation. We evaluated its efficiency through a set of computational experiments. To solve instances of larger problem dimensions, we have developed a heuristic method based on a Variable Neighborhood Search (VNS). A local search procedure with efficient fast swap-based local search is implemented in the proposed VNS-based approach. Presented computational results show that the proposed VNS is computationally efficient and quickly reaches all optimal solutions for smaller dimension instances obtained by exact solver and provide high-quality solutions on large-scale problem instances in short CPU times.

1. Introduction

The motivation for our research stems from Belgrade Business School, where students from the same grade must be divided into groups in such a way that each group provides a good representation of the classroom population. School administration chooses student attributes and determine their relative importance. For example, some of attributes are gender, age, current grades, country of origin, etc. After the attributes and their weights are determined, it is necessary to form groups and measure the quality of their composition and balance. This problem can be seen as general Balanced Multi-Weighted Attribute Set Partitioning (BMWASP) problem which requires finding a partition of a given set of objects with multiple weighted attributes into a certain number of groups so that the groups are as balanced as possible with respect to the number of elements possessing each attribute.

BMWASP problem arises in many real-life applications, ranging from assigning students to study groups [14] to designing level schedules for JIT assembly lines [18]. Specially to properly isolate and nullify

2010 *Mathematics Subject Classification*. Primary 90C27; Secondary 90C59, 65K10, 90C90, 90C11.
Keywords. set partitioning, balanced groups, variable neighborhood search
Received: 20 May 2018; Accepted: 05 July 2018
Communicated by Dragan S. Djordjević
This research was partially supported by Serbian Ministry of Education and Science under the grants No. 174010.

Email addresses: `dzamic@fon.bg.ac.rs` (Dušan Džamić), `bojana.cendic@bbs.edu.rs` (Bojana Ćendić), `maricm@matf.bg.ac.rs` (Miroslav Marić), `djenic@matf.bg.ac.rs` (Aleksandar Đenić)

any nuisance or confounding variables in many true experimental designs BMWASP can be used to divide objects into two balanced rather than randomly groups (test group and a control). One example is education, where researchers want to monitor the effect of a new teaching method upon groups of children. Other areas include evaluating the effects of counseling, testing medical treatments, and measuring psychological constructs. Both groups are pre-tested, and both are post-tested, the ultimate difference being that one group was administered the treatment.

1.1. *Review of the literature related to BMWASP*

Generally, the objective of balanced partitioning (classification) is the reverse of the standard classification problem. Cormack [5] defines classification as the process by which objects are allocated to groups so that objects within a group are similar to one another. Thus the balanced partitioning addresses the question of how to create similar groups containing dissimilar objects. In the literature can be found different variants of balanced partitioning problem, each in turn considering different assumptions, constraints and objective functions.

In [7] Desrosiers et al. use a centroid to represent each group of entities, and propose two different ways of measuring the balance among groups min-sum and min-max objectives. Model with min-sum objective minimizes the sum of weighted distances between team centroids and the target vector of attributes. Model with min-max objective minimizes the maximum weighted distance between team centroids and the target vector of attributes. The authors apply their method to partition 120 MBA students in groups of 5 at HEC, Montreal. In [1] Baker and Benn presented case study which consists of assigning 235 students to eight tutor groups. They used mixed linear programming formulation with min-sum objective. In [13] Krass et al. enforce balance through hard constraints and problem observed as satisfiability (feasibility of perfectly balanced partition), rather than an optimization problem. The authors analyze one representative practical application to design student groups at the Rotman School of Management, University of Toronto. The goals of the analysis in [13] are to understand which classes of balancing problems may contain infeasible instances and how prevalent such instances are within these classes. Another approach that can achieve the balancing between the groups is to maximize the diversity in each group [4, 6, 12]. The maximum diverse grouping problem (MDGP) consists of finding a way to divide a set of elements into mutually disjoint groups so that the total diversity among the elements belonging to the same group is maximized [10]. In [3] Bhadury et al. propose simplified model where the population comes partitioned into 'families' with a high degree of intra-familial similarity and inter-familial dissimilarity. A network flow problem, known as the dining problem, is used to assign students to different projects.

Our problem formulation differs from others in the literature because the data is binary and multidimensional with weight coefficients. By contrast, Behestian et al. [2] use either one-dimensional attributes or composite scores in an attempt to integrate attributes that are not easily commensurable Mingers and O'Brien [16] developed a heuristic algorithm and compared it to a mixed linear programming formulation for the balanced partitioning involving only binary-valued attributes.

1.2. *Main contribution*

Our approach for solving BMWASP is to define an appropriate criterion allowing to compare the degree of deviation from the "perfect balance" for different partitions and then produce the partition that minimizes this criterion. First we propose an mathematical formulation for the BMWASP to minimization the total distance from the ideal number of each attribute in each group including attributes weight. Second, we present mixed-integer linear reformulation and evaluate its efficiency through a set of computational experiments. Third, we propose a simple and fast method for calculating lower bounds using relaxation of proposed formulation.

In order to solve larger problems more efficiently we have developed a heuristic method based on a Variable Neighborhood Search (VNS) for solving BMWASP. A local search procedure with efficient fast swap-based local search is implemented in the proposed VNS-based approach. The proposed algorithm presented in this paper achieved all optimal solutions for smaller dimension instances obtained by exact solver and provide high-quality solutions on large-scale problem instances in short CPU times. Following the constraints and objectives given by administrators at the Belgrade Belgrade Business School, we have successfully applied proposed method to partition 229 students in 10 groups.

2. Problem formulation of BMWASP

In BMWASP, there are n objects to be partitioned in k groups. Each object $s_i, (i = 1, 2 \ldots, n)$ has one or more attributes $c_j, (j = 1, 2, \ldots, m)$ and each attribute c_j has a weight w_j. The goal is to partition set of n objects into k mutually disjoint groups such that each group contains approximately the same number of objects possessing each attribute. In process of partitioning objects into groups attributes with greater weight have a higher priority than the attributes with less weight.

2.1. *Mathematical formulation*

This subsection will introduce the mathematical formulation of the BMWASP problem, which requires the following notation:

- $S = \{s_1, s_2, s_3, \ldots, s_n\}$ - set of n objects;
- $G = \{g_1, g_2, g_3, \ldots, g_k\}$ - set of k groups;
- $C = \{c_1, c_2, c_3, \ldots, c_m\}$ - set of m attributes;
- w_j - weight of attribute c_j for $j = 1, 2, \ldots\ldots, m$;
- $a_{ij} = \begin{cases} 1 & \text{, if object } s_i \text{ has attribute } c_j \\ 0 & \text{, otherwise} \end{cases}$

  for $i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m$;
- $c_j^{avg} = \frac{1}{k} \sum_{i=1}^{n} a_{ij}$ - ideal number of attribute $j = 1, 2, \ldots m$ in group
- $x_{il} = \begin{cases} 1 & \text{, if object } s_i \text{ is assigned to group } g_l \\ 0 & \text{, otherwise} \end{cases}$

  for $i = 1, 2, \ldots, n, \quad l = 1, 2, \ldots, k$;

Using the above notation BMWASP problem are defined as follows:

$$\text{Minimize} \sum_{j=1}^{m} \sum_{l=1}^{k} w_j \left| y_{jl} - c_j^{avg} \right| \tag{1}$$

subject to:

$$y_{jl} = \sum_{i=1}^{n} a_{ij} x_{il}, \quad j = 1, 2, \ldots m, \ l = 1, 2, \ldots, k; \tag{2}$$

$$\sum_{l=1}^{k} x_{il} = 1, \quad i = 1, 2, \ldots n; \tag{3}$$

$$\sum_{i=1}^{n} x_{il} \leq \left\lceil \frac{n}{k} \right\rceil, \quad l = 1, 2, \ldots k; \tag{4}$$

$$\sum_{i=1}^{n} x_{il} \geq \left\lfloor \frac{n}{k} \right\rfloor, \quad l = 1, 2, \ldots k; \tag{5}$$

$$x_{ij} \in \{0, 1\} \text{ for } i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m; \tag{6}$$

We will denote this formulation as BMWASP-O throughout the article. The objective function (1) minimizes the total distance from the ideal number of each attribute in each group including attributes weight. Constraint (2) is used to define the number of each attribute in each group. The constraits (3) ensure that each object is assigned to exactly one group. Constraints (4) and (5) impose maximum and minimum group sizes, respectively. The above model allows groups to differ in size if n is not divisible by k. Else constraints (4) and (5) in BMWASP-O can be replaced with

$$\sum_{i=1}^{n} x_{il} = \frac{n}{k}, \quad l = 1, 2, \ldots k. \tag{7}$$

2.2. *MILP reformulation*

The proposed BMWASP-O is Non Linear Programming (NLP) formulation, because of the non-linear absolute value function. Absolute value functions are very difficult to perform standard optimization procedures on. They are not continuously differentiable functions, nonlinear, and relatively difficult to operate on. However, through simple manipulation of the absolute value expression, these difficulties can be avoided and the problem can be reformulated as Mixed-Integer Linear Programming (MILP) [15]. The motivation for our reformulation, denoted as the BMWASP-I stems from the desire to solve larger problems to optimality more efficiently.

Let $d_{jl} = y_{jl} - c_j^{avg}$ for $j = 1, 2, \ldots m,\ l = 1, 2, \ldots, k$. For each d_{jl} we begin by defining two new non-negative variables d_{jl}^{+} and d_{jl}^{-}. Then substitute $d_{jl}^{+} - d_{jl}^{-}$ for d_{jl} in each constraint and in objective function. Also add sign restrictions $d_{jl}^{+} \geq 0$ and $d_{jl}^{-} \geq 0$. The absolute value expression $\left|d_{jl}\right| = \left|d_{jl}^{+} - d_{jl}^{-}\right|$ can be simplified whenever either $d_{jl}^{+} = 0$ or $d_{jl}^{-} = 0$. Algebraically, if the product of the variables is zero $d_{jl}^{+} \cdot d_{jl}^{-} = 0$ then the above absolute value expression can be written as the sum of the two variables $\left|d_{jl}^{+} - d_{jl}^{-}\right| = \left|d_{jl}^{+}\right| + \left|d_{jl}^{-}\right| = d_{jl}^{+} + d_{jl}^{-}$. Imposing the restriction that one or the other variable is zero, the formulation becomes:

$$\text{Minimize} \sum_{j=1}^{m} \sum_{l=1}^{k} (w_j \cdot d_{jl}^{+} + w_j \cdot d_{jl}^{-}) \tag{8}$$

subject to:

$$y_{jl} = \sum_{i=1}^{n} a_{ij} x_{il}, \quad j = 1, 2, \ldots m,\ l = 1, 2, \ldots, k; \tag{9}$$

$$\sum_{l=1}^{k} x_{il} = 1, \quad i = 1, 2, \ldots n; \tag{10}$$

$$\sum_{i=1}^{n} x_{il} \leq \left\lceil \frac{n}{k} \right\rceil, \quad l = 1, 2, \ldots k; \tag{11}$$

$$\sum_{i=1}^{n} x_{il} \geq \left\lfloor \frac{n}{k} \right\rfloor, \quad l = 1, 2, \ldots k; \tag{12}$$

$$d_{jl}^{+} - d_{jl}^{-} = y_{jl} - c_j^{avg}, \quad j = 1, 2, \ldots m, \; l = 1, 2, \ldots, k; \tag{13}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m; \tag{14}$$

$$d_{jl}^{+} \geq 0, \quad d_{jl}^{-} \geq 0 \quad j = 1, 2, \ldots m, \; l = 1, 2, \ldots, k. \tag{15}$$

Constraint ($d_{jl}^{+} \cdot d_{jl}^{-} = 0 \quad j = 1, 2, \ldots m, \; l = 1, 2, \ldots, k$) is not necessary and can be eliminated. Consider a simple problem:

Minimize $(d^{+} + d^{-})$ subject to: $d^{+} - d^{-} = C; \quad d^{+} \geq 0$ and $d^{-} \geq 0$.

If $C \geq 0$ then $\min(d^{+} + d^{-}) = C$ at $(d^{+}, d^{-}) = (C, 0)$, else if $C < 0$ then $\min(d^{+} + d^{-}) = -C$ at $(d^{+}, d^{-}) = (0, -C)$. In both cases minimization will automatically cause $d^{+} \cdot d^{-} = 0$ and this non-linear constraint is not necessary. Some of decision variables are integers, while other variables are allowed to be non-integers. So the model is an Mixed Integer Linear Programming (MILP) formulation.

2.3. *Relaxation*

Lower bound is a value which is guaranteed less than or equal to the optimal solution, so if a solution obtained by some algorithm, reaches the lower bound, the solution must be optimal. Using relaxation of the problem that is easier to solve we propose a simple and fast method for calculating lower bounds.

We relax the problem by breaking the connection between objects and attributes. In this case the arrangement of objects in groups was irrelevant and problem can be easy solved as a assignment problem of available attributes to groups. So, we obtain one lower bound by calculating value of objective function for ideal assignment of each attribute.

Let $I = (n, m, k, w_j, A)$ be an instances of BMWASP problem with k groups, n objects where each object is characterized by a vector of m attributes, through attribute matrix A, such that $a_{ij} = 1$ if object i possesses attribute j, and $a_{ij} = 0$ otherwise. Then

$$LB = \sum_{j=1}^{m} w_j \left(\theta_j \cdot \lambda_j + (k - \theta_j) \cdot (1 - \lambda_j) \right) \tag{16}$$

where is

$$\theta_j = \sum_{i=1}^{n} a_{ij} \mod k; \tag{17}$$

$$\lambda_j = \begin{cases} \max(\Delta_j, 1 - \Delta_j) & , \Delta_j \leq \frac{1}{2} \\ \min(\Delta_j, 1 - \Delta_j) & , \Delta_j \geq \frac{1}{2} \end{cases} ; \quad \Delta_j = \sum_{i=1}^{n} \frac{a_{ij}}{k} - \left\lfloor \sum_{i=1}^{n} \frac{a_{ij}}{k} \right\rfloor. \tag{18}$$

If a perfectly balanced partition exists, value of objective function is equal to LB.

3. Variable neighborhood search

VNS metaheuristic was introduced by Mladenovic and Hansen [17] for solving the traveling salesman problem (TSP). After that, VNS has been applied by more and more researchers to solve continuous and discrete optimization problems [11].

VNS metaheuristic combines local search with systematic changes of neighborhood in the descent and escape from local optimum. The basic VNS employs a set of predefined neighborhoods. By sequentially exploring these neighborhoods, local optima solutions in different neighborhood structure can be obtained, and thus, better solutions can be reached through this process. In VNS, one solution in the current neighborhood is selected as the incumbent solution. Then, local search is performed on the selected solution to generate several neighboring solutions. By comparing the neighboring solutions with the incumbent one, the current solution will be replaced by the best solution found thus far or remain its state if no better solution is found. Then, the neighborhood will turn to the first one if a better solution is found or turn to the next one if the incumbent solution remains itself. By systematically changing the current neighborhood, VNS directs the search to a promising field, and thus, global optimal solutions will be found.

The main steps of the basic VNS are given as follows:

1. **Initialization:** Select a set of neighborhood structures $N_p(p = 1, \dots, p_{max})$, find an initial solution x, set $p = 1$, choose a stopping condition.

2. **Shaking:** Generate a solution $x' \in N_p(x)$ at random.

3. **Local search:** Apply a local search method starting with x' to find local optimum x''.

4. **Move or not**: If x'' is better than the incumbent, then set $x = x''$ and $p = 1$, otherwise set $p = p + 1$ (or if $p = p_{max}$ set $p = 1$).

5. **Test stop condition:** If stop condition is not satisfied then go to step 2. Otherwise return the best solution x.

Inspired by successful application of VNS for forming four member heterogeneous groups within CSCL (Computer supporting collaborative learning) [20] and in other numerous fields [8, 9], we have developed a variant of VNS for solving the BMWASP. The advantage of the proposed VNS is in the implemented fast interchange method used within local search in order to obtain improvements in an efficient manner. In the following subsections, all aspects of the proposed VNS-based method will be explained in detail.

3.1. *Solution space of BMWASP*

The solution space in BMWASP includes all possible partitions of n objects into k groups where each created group g_l contains at least $\left\lfloor \frac{n}{k} \right\rfloor$ and at most $\left\lceil \frac{n}{k} \right\rceil$ objects. The number of ways to arrange n objects in k groups is:

$$C_{n,k} = \frac{n!}{\left(\left\lceil \frac{n}{k} \right\rceil !\right)^{\hat{k}} \cdot \left(\left\lfloor \frac{n}{k} \right\rfloor !\right)^{(k-\hat{k})} \cdot \hat{k}! \cdot (k-\hat{k})!},$$

where $\hat{k}$ is the number of groups which contain exactly $\left\lceil \frac{n}{k} \right\rceil$ objects, i.e., $\hat{k} = k$, if n is divisible by k, else $\hat{k} = (n \bmod k)$. For example, the number of possible ways to arrange 20 objects into 4 where each group contains 5 objects is $C_{20,4} = 488\,864\,376$. In Appendix A we described procedure to derive number $C_{n,k}$.

3.2. *The structure of the VNS method for BMWASP*

For that purpose VNS algorithm receives six input parameters: n - number of objects, m - number of possible attributes for each object, k - number of groups for partitioning, p_{max} - maximum size of the neighborhood, w_j - weights of attributes and a_{ij} - matrix with associated attributes for each object. The pseudo-code of the VNS based heuristic is given in Algorithm 1.

```
    input  : $n, m, k, p_{max}, w_j, a_{ij}$
    output: solution $x_{best}$
 1  $x_{current} \leftarrow$ RandomSolution();
 2  $x_{current} \leftarrow$ RVNS($x_{current}$);
 3  $p \leftarrow 1$;
 4  while stop condition is not satisfied do
 5      $x_{temp} \leftarrow$ Shaking($x_{current}, p$);
 6      $x_{temp} \leftarrow$ LocalSearch($x_{temp}$);
 7      if $x_{temp}$ is better than $x_{current}$ then
 8          $x_{current} \leftarrow x_{temp}$;
 9          $p \leftarrow 1$;
10      else
11          $p \leftarrow p + 1$ ;
12          if $p > p_{max}$ then
13              $p \leftarrow 1$;
14          end
15      end
16  end
17  $x_{best} \leftarrow x_{current}$;
```

Algorithm 1: VNS scheme

Initial feasible solution $x_{current}$ is obtained in the first two steps from a random solution followed by Reduced VNS algorithm (RVNS), which consists of repetition of shake phase. The idea is to use the RVNS method to quickly find a good initial solution for the basic VNS part. The stopping condition of the RVNS procedure is reaching a maximum number of iterations without improving the current solution. The maximum number of iterations is 30000. The initial value of variable p (size of the neighborhood) are defined at step 3. The central part of the VNS algorithm consists of the loop executed in lines 4-16. Loop is repeated until a stopping condition is met, which is 30 iterations without an improvement. In step 5, shaking routine is performed on the solution $x_{current}$. The shaking phase moves the current best solution to a random p-neighborhood which contains solutions obtained by p swapping a single pair of objects belonging to different groups. After shaking, local search is applied over x_{temp} in line 6. In steps 7-9, the current solution is updated if x_{temp} is better then $x_{current}$ and value of parameter p is set to 1. Otherwise, if x_{temp} is not accepted the size of the neighborhood p is increased by one. The value of parameter p is reset to 1 in steps 12-14 if it exceeds value of given parameter p_{max}. Finally, after the main loop terminates, in step 17 the best found solution by VNS is claimed to be x_{best}.

For local search we use best-improvement strategy and swap neighborhood which contains solutions obtained by swapping a single pair of objects belonging to different groups. The pseudo-code of local search part is given in Algorithm 2. In the first line value of the objective function for the initial solution is calculated. This step is not necessary if the value of the objective function is updated in shaking phase similar as described below. Initial solution is claimed to be the best (x_{best}) in step 2 and objective function for the best solution is initialized in step 3. The initial value of variable *improvement* is defined at step 4. The central part of local search consists of the loop executed in lines 5-36 while it is possible to improve the solution by swapping. For each single pair of objects belonging to different groups in steps 11-14 is obtained x_{temp} solutions by swapping them and then updates the objective function value.

Let object s_{i_1} be in group g_{l_1} and object s_{i_2} in group g_{l_2} of the current solution $x_{current}$. Denote with x_{temp} the solution obtained after moving the object s_{i_1} into group g_{l_2} and the object s_{i_2} into group g_{l_1}. Since the object s_{i_1} is removed from the group g_{l_1} and inserted in the group g_{l_2}, new number of attributes in group g_{l_1} and g_{l_2} contribute to the objective function value of the new solution. But, because the others groups g_l, where $l = 1, 2, \ldots k$ and $l \neq l_1$, $l \neq l_2$ are unchanged, number of attributes in these groups do not contribute to the objective function value of the new solution. In order to use the previous fact and speed up local search we maintain matrix y such that y_{jl} is the number of objects in group g_l which contain attribute c_j.

We have also initialized array c_j^{avg} (for example, after loading the data) which define the ideal number of attribute c_j in groups . So, in one iteration through all attributes in steps 15-27 we can finally calculate the objective values of the neighboring solution. Condition in line 16 skips attributes which not contribute to the objective function value. Before update number of attributes for groups involved in swapping in step 17 objective function value is reduced. Then after matrix y is updated (in steps 18-24) objective function value is increased for new contribution of attribute c_j in these groups. In steps 28-31, the best solution is updated if x_{temp} is better then current x_{best} and value of parameter *improvement* is set to *true*. It is obvious that the change of the objective value for each solution from swap neighborhood is done in $O(m)$.

```
   input : solution $x_{initial}$
   output: solution $x_{best}$
 1 $f_{initial} \leftarrow$ CalculateCost($x_{initial}$);
 2 $x_{best} \leftarrow x_{initial}$;
 3 $f_{best} \leftarrow f_{initial}$;
 4 $improvement \leftarrow true$ ;
 5 while $improvement$ do
 6   $improvement \leftarrow$ false;
 7   for $l_1 = 1$ to $k - 1$ do
 8     foreach object $s_{i_1}$ in group $g_{l_1}$ do
 9       for $l_2 = l_1 + 1$ to $k$ do
10         foreach object $s_{i_2}$ in group $g_{l_2}$ do
11           $x_{temp} \leftarrow x_{initial}$;
12           $f_{temp} \leftarrow f_{initial}$;
13           $x_{temp}^{g_{l_1}} \leftarrow \left(x_{temp}^{g_{l_1}} \setminus \{s_{i_1}\}\right) \cup \{s_{i_2}\}$;
14           $x_{temp}^{g_{l_2}} \leftarrow \left(x_{temp}^{g_{l_2}} \setminus \{s_{i_2}\}\right) \cup \{s_{i_1}\}$;
15           foreach attribute $c_j$ do
16             if $a_{i_1,j} \neq a_{i_2,j}$ then
17               $f_{temp} \leftarrow f_{temp} - w_j \cdot |y_{jl_1} - c_j^{avg}| - w_j \cdot |y_{jl_2} - c_j^{avg}|$;
18               if $a_{i_1,j} = 1$ and $a_{i_2,j} = 0$ then
19                 $y_{j,l_1} \leftarrow y_{j,l_1} - 1$;
20                 $y_{j,l_2} \leftarrow y_{j,l_2} + 1$;
21               else
22                 $y_{j,l_1} \leftarrow y_{j,l_1} + 1$;
23                 $y_{j,l_2} \leftarrow y_{j,l_2} - 1$;
24               end
25               $f_{temp} \leftarrow f_{temp} + w_j \cdot |y_{jl_1} - c_j^{avg}| + w_j \cdot |y_{jl_2} - c_j^{avg}|$;
26             end
27           end
28           if $f_{temp} < f_{best}$ then
29             $x_{best} \leftarrow x_{temp}$;
30             $improvement \leftarrow true$ ;
31           end
32         end
33       end
34     end
35   end
36 end
```

Algorithm 2: Local search scheme

4. Experimental results

In this section, we evaluate efficiency of BMWASP-O model and MILP reformulation BMWASP-I through a set of computational experiments. The results of computational experiments conducted to verify the performance of the proposed VNS algorithm are presented as follows.

Computational experiments were first performed on problem instances that were generated randomly by varying the values of objects from 10 to 250, included 12 levels of attributes value and 7 levels of groups partitions. We have also performed experiments on a real-life problem instance with 229 objects and 116 attributes for each of them that was derived from a real situation in Belgrade Business School.

To the best of our knowledge, the public test instances available for this problem does not exist. In order to evaluate the proposed algorithm and models, 300 problems are designed in various conditions. The following parameters are considered to design and generate these problems:

– numbers of objects: 10, 25, 50, 100, 250.
– numbers of attributes for each object: 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100.
– numbers of groups: 2, 5, 10, 15, 20, 25, 30.
– attributes for each object: discrete uniform distribution $\mathcal{U}\{0,1\}$.
– weight for each attribute: discrete uniform distribution $\mathcal{U}\{1,3\}$.

This distribution of parameters provides a good range of problems and the proposed algorithm can be tested and evaluated in various conditions. For example, the notation o50a20g5 means a 50-object, 20-attribute, 5 group problem. The letters o, a and g are abbreviations for object, attribute and group, respectively.

Mathematical programming and constraint programming are two technologies for solving complex planning and optimization problems. A constraint programming solver proves optimality by showing that no better solution than the current one can be found, while an mathematical programming solver uses a lower bound proof provided by cuts and linear relaxation. ILOG CPLEX provides CP Optimizer for solving a constraint programming problems with a large set of arithmetic-logical constraints and high-performance MIP optimizer for solving a mixed-integer programming problems using a very general and robust algorithm based on branch & cut. Computational experiments with proposed mathematical formulations were carried out by using CPLEX CP 12.5 solver for BMWASP-O formulation and CPLEX MIP 12.5 solver for BMWASP-I formulation. For all instances time limit is set to 14400s (4 hours). The proposed VNS was implemented on .NET Framework platform (Mono open source implementation) using C# programming language. As noted earlier, VNS algorithm runs until the limit of 30 iterations without an improvement is reached. The values of parameter p_{max} is set to $min(\lceil n/6 \rceil, 20)$. All computational tests were performed on an Intel Core i5-2400 CPU 3.10GHz with 4GB of RAM memory under Ubuntu operating system.

Table 1 present complete computational results obtained with CPLEX optimizers and proposed VNS algorithm for small instances with up to 25 objects. First column refers to the name of the instance. The second, third, fourth and fifth column present, for the optimal solution obtained with CPLEX CP and MIP Optimizer, the value of the objective function and time in seconds required by CPLEX Optimizer to calculate it. Sixth column present the value of the objective function for the best solution obtained by VNS in the 20 independent runs. Seventh and eighth column refers to average execution time, and standard deviation of objective function value. Finally, last column present lower bound value obtained with proposed method in section 2. The optimal values of objective function obtained with VNS algorithm are in bold.

CP optimizer was able to solve 31 small instance, while MIP optimizer solved 42 instance. VNS found all known optimal solutions very quickly, and have similar execution time as CPLEX on small instances, but significantly smaller execution time on the larger ones.

The results of conducted experimental study on 252 larger instances showed that the BMWASP-I formulation is superior over the BMWASP-O. CPLEX CP solver has not been able to solve any instances optimality with the specified time limit. CPLEX MIP solver was capable of solving 57 of 252 instances optimality with the specified time limit. A very common difficulty with MIP solver is running out of memory when the branch and cut tree becomes so large that insufficient memory.

Table 1: Comparison of computational results on small instances

Instance	CP (BMWASP-O)		MIP (BMWASP-I)		VNS algorithm			LB
name	f	t	f	t	f_{best}	t_{avg}	σ	
o10a5g2	3.00	0.26	3.00	0.08	**3.00**	0.07	0.00	3.00
o10a10g2	11.00	0.56	11.00	0.03	**11.00**	0.14	0.00	11.00
o10a15g2	26.00	0.48	26.00	0.08	**26.00**	0.20	0.00	22.00
o10a20g2	26.00	0.55	26.00	0.09	**26.00**	0.24	0.00	18.00
o10a30g2	48.00	0.57	48.00	0.06	**48.00**	0.36	0.00	20.00
o10a40g2	52.00	0.61	52.00	0.08	**52.00**	0.47	0.00	26.00
o10a50g2	94.00	0.75	94.00	0.11	**94.00**	0.59	0.00	50.00
o10a60g2	104.00	0.94	104.00	0.09	**104.00**	0.69	0.00	72.00
o10a70g2	134.00	0.98	134.00	0.12	**134.00**	0.80	0.00	68.00
o10a80g2	144.00	1.06	144.00	0.13	**144.00**	0.90	0.00	84.00
o10a90g2	158.00	1.16	158.00	0.17	**158.00**	1.01	0.00	88.00
o10a100g2	195.00	1.20	195.00	0.15	**195.00**	1.15	0.00	89.00
Average time:		*0.76*		*0.10*		*0.60*		
o10a5g5	13.60	6.45	13.60	0.05	**13.60**	0.08	0.00	13.60
o10a10g5	33.20	48.57	33.20	0.10	**33.20**	0.13	0.00	31.20
o10a15g5	56.40	71.27	56.40	0.16	**56.40**	0.20	0.00	40.80
o10a20g5	59.20	78.42	59.20	0.15	**59.20**	0.26	0.00	37.60
o10a30g5	134.00	180.10	134.00	0.55	**134.00**	0.37	0.00	87.20
o10a40g5	172.00	293.40	172.00	0.77	**172.00**	0.49	0.00	118.40
o10a50g5	210.80	335.90	210.80	0.50	**210.80**	0.59	0.00	144.80
o10a60g5	261.60	493.00	261.60	1.01	**261.60**	0.70	0.00	173.60
o10a70g5	306.40	516.66	306.40	1.70	**306.40**	0.83	0.00	195.20
o10a80g5	340.40	590.85	340.40	0.85	**340.40**	0.93	0.00	258.40
o10a90g5	419.20	811.50	419.20	2.18	**419.20**	1.05	0.00	282.40
o10a100g5	452.40	901.74	452.40	2.39	**452.40**	1.20	0.00	286.40
Average time:		*360.65*		*0.87*		*0.60*		
o25a5g2	1.00	5.60	1.00	0.03	**1.00**	0.11	0.00	1.00
o25a10g2	14.00	180.92	14.00	0.06	**14.00**	0.19	0.00	14.00
o25a15g2	15.00	208.28	15.00	0.04	**15.00**	0.34	0.89	15.00
o25a20g2	27.00	670.43	27.00	0.08	**27.00**	0.41	1.73	25.00
o25a30g2	48.00	2620.94	48.00	0.11	**48.00**	0.64	3.69	34.00
o25a40g2	66.00	3647.57	66.00	0.22	**66.00**	0.85	3.51	34.00
o25a50g2	94.00	5565.21	94.00	0.36	**94.00**	1.01	3.86	48.00
o25a60g2	-	-	130.00	0.52	**130.00**	1.18	4.26	72.00
o25a70g2	-	-	146.00	1.15	**146.00**	1.40	2.89	64.00
o25a80g2	-	-	194.00	1.70	**194.00**	1.57	3.28	72.00
o25a90g2	-	-	209.00	1.52	**209.00**	1.94	8.57	85.00
o25a100g2	-	-	252.00	2.50	**252.00**	1.90	5.71	110.00
Average time:		-		*0.70*		*1.00*		
o25a5g5	-	-	20.80	0.16	**20.80**	0.12	0.00	20.80
o25a10g5	-	-	23.20	0.22	**23.20**	0.27	0.93	23.20
o25a15g5	-	-	61.20	14.46	**61.20**	0.42	0.96	59.20
o25a20g5	-	-	88.00	74.08	**88.00**	0.55	1.75	79.20
o25a30g5	-	-	130.00	713.30	**130.00**	0.82	1.51	100.00
o25a40g5	-	-	176.80	2121.98	**176.80**	1.10	2.19	116.80
o25a50g5	-	-	-	-	220.00	1.27	5.38	139.20
o25a60g5	-	-	-	-	345.60	1.61	3.32	187.20
o25a70g5	-	-	-	-	359.60	1.78	5.88	206.40
o25a80g5	-	-	-	-	433.60	1.86	3.43	242.40
o25a90g5	-	-	-	-	528.00	2.36	4.47	284.80
o25a100g5	-	-	-	-	586.80	2.84	6.66	312.80
Average time:		-		-		*1.30*		

Proposed VNS algorithm has reached 54 of 57 known optimal solutions in a quite short amount of computational time. Also for three more instances (o250a15g10, o250a15g15 and o250a15g20) solutions obtained by VNS algorithm reaches the lower bound. As noted earlier, lower bound is a value which is guaranteed less than or equal to the optimal solution, so if a objective function of solution obtained by VNS reaches the lower bound, the solution must be optimal. The proposed method also provides solutions on large-scale instances in reasonable amount of CPU time. Detailed results obtained with VNS algorithm for all instances are given in Table 4, Table 5 and Table 6 in Appendix B.

To perform an objective comparison of running times, we compare average CPU time of the proposed VNS algorithm with the average CPU time of the CP solver, but only for the instances for which both algorithms reached the same solution. In the same way, we compare the average CPU times of the proposed VNS and the MIP solver. The summary of comparison of CPU times is presented in Table 2. On average, the proposed VNS algorithm has shorter average CPU time compared to both CP solver and MIP solver on the subset of test instances on which the same solution is reached.

Table 2: Comparison of the average CPU times

Comparison	No. of instances with the same solution	VNS (t_{avg})	CP/MIP solver (t_{avg})
VNS vs CP	31 (31 + 0)	0.50	556.00
VNS vs MIP	96 (42 + 54)	5.98	50.81

4.1. *Application of BMWASP in Belgrade Belgrade Business School*

In Belgrade Belgrade Business School, 229 students from the same grade must be divided into 10 groups in such way that each group provides a good representation of the classroom population. School administration chooses 116 student attributes and determine their relative importance.

Table 3 shows comparison of computational results obtained with CPLEX CP solver, CPLEX MIP solver and VNS algorithm. CPLEX CP Optimizer was interrupted after 28800 seconds (8 hours) and CPLEX MIP optimizer was interrupted after 86400 seconds (24 hours). Figure 1 presents the improvement of the objective function by this three methods during the execution time (in seconds).

Table 3: Comparasion of computational results - Belgrade Belgrade Business School

	Property	Value
CPLEX CP Optimizer	objective value after 8h	1151.20
	number of improved solutions	75
	memory usage	149.20 MB
	explored branches	7 326 914
CPLEX MIP Optimizer	lower bound	390.40
	upper bound	709.80
	number of improved solutions	19
	memory usage	658.20 MB
	iterations	118 828 066
LB	lower bound	390.40
VNS	best objective value	653.00
	average objective value	671.84
	average execution time	567.60s
	memory usage	36.50 MB
	standard deviation	11.81
	relative standard deviation	1.8

The obtained results showed that the average value of the objective function obtained by VNS algorithm in less than 10 minutes is much better than that achieved by the parallel CPLEX CP Optimizer with 4 workers after 8 hours. The results also showed that CPLEX MIP Optimizer can not find better upper and lower bound then proposed VNS algorithm and method for calculating lower bounds.

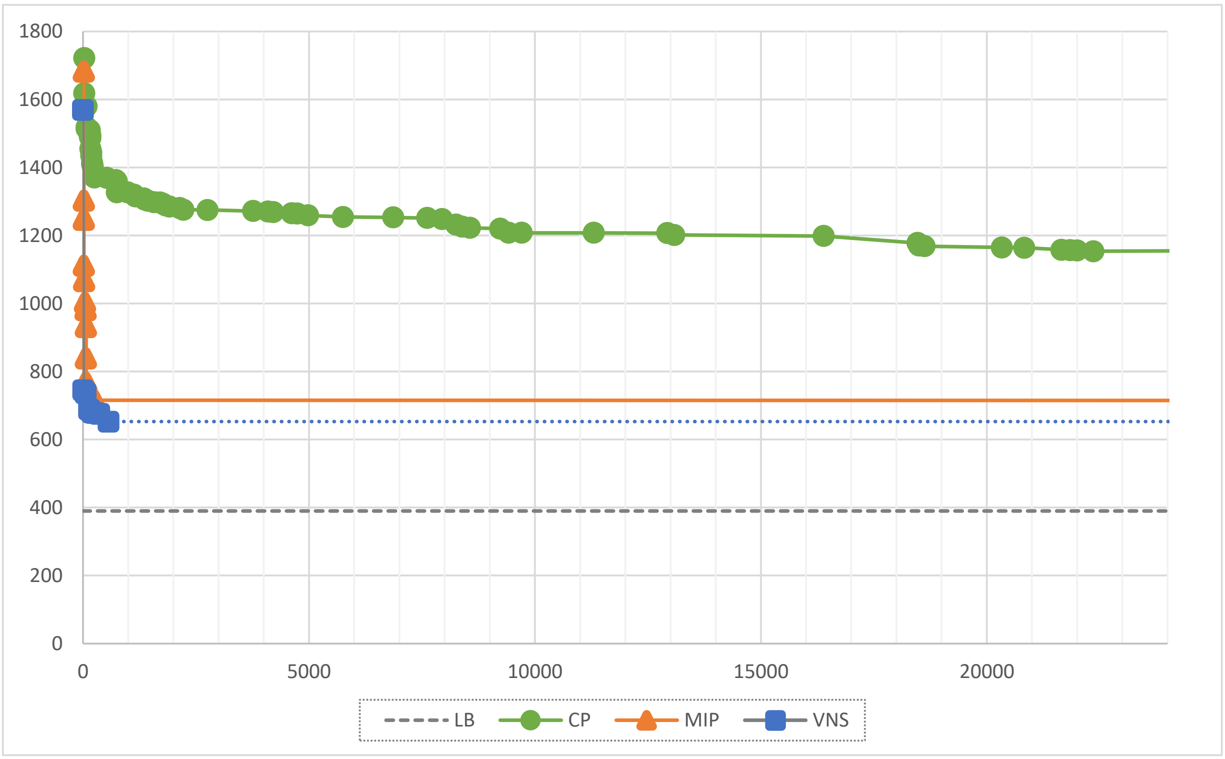

Figure 1: Improvement of the objective function value during execution

5. Conclusion

The Balanced Multi-Weighted Attribute Set Partitioning (BMWASP) problem requires finding a partition of a given set of objects with multiple weighted attributes into a certain number of groups so that the groups are as balanced as possible with respect to the number of elements possessing each attribute. Our approach is to define an appropriate criterion allowing to compare the degree of deviation from the ”perfect balance” for different partitions and then produce the partition that minimizes this criterion.

We have proposed an mathematical formulation BMWASP-O and its mixed-integer linear reformulation BMWASP-I in order to solve larger problems to optimality more efficiently. For testing purposes we have created 300 new instances. In order to solve large-scale instances we suggested a new variant of Variable Neighborhood Search (VNS) with efficient fast swap-based local search. Results of experimental study showed that the BMWASP-I formulation is superior over the BMWASP-O. Based on extensive computational tests, we have showed that our new heuristic quickly reaches all known optimal solutions and provides high-quality solutions on large-scale problem instances in short CPU times.

Finally we have successfully applied BMWASP and proposed VNS algorithm at the Belgrade Belgrade Business School for forming study groups.

References

[1] B. Baker and C. Benn. Assigning pupils to tutor groups in a comprehensive school. *Journal of the Operational Research Society*, 52(6):623–629, 2001.
[2] M. Beheshtian-Ardekani and M. A. Mahmood. Education development and validation of a tool for assigning students to groups for class projects. *Decision Sciences*, 17(1):92–113, 1986.
[3] J. Bhadury, E. J. Mighty, and H. Damar. Maximizing workforce diversity in project teams: A network flow approach. *Omega*, 28(2):143–153, 2000.

[4] J. Brimberg, N. Mladenović, and D. Urošević. Solving the maximally diverse grouping problem by skewed general variable neighborhood search. *Information Sciences*, 295:650–675, 2015.

[5] R. M. Cormack. A review of classification. *Journal of the Royal Statistical Society. Series A (General)*, pages 321–367, 1971.

[6] M.-I. Dascalu, C.-N. Bodea, M. Lytras, P. O. De Pablos, and A. Burlacu. Improving e-learning communities through optimal composition of multidisciplinary learning groups. *Computers in Human Behavior*, 30:362–371, 2014.

[7] J. Desrosiers, N. Mladenović, and D. Villeneuve. Design of balanced mba student teams. *Journal of the Operational Research Society*, 56(1):60–66, 2005.

[8] A. Djenić, M. Marić, Z. Stanimirović, and P. Stanojević. A variable neighbourhood search method for solving the long-term care facility location problem. *IMA Journal of Management Mathematics*, page dpw008, 2016.

[9] A. Djenić, N. Radojičić, M. Marić, and M. Mladenović. Parallel vns for bus terminal location problem. *Applied Soft Computing*, 42:448–458, 2016.

[10] T. A. Feo and M. Khellaf. A class of bounded approximation algorithms for graph partitioning. *Networks*, 20(2):181–195, 1990.

[11] P. Hansen, N. Mladenović, and J. A. M. Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[12] R. Hubscher. Assigning students to groups using general and context-specific criteria. *IEEE Transactions on Learning Technologies*, 3(3):178–189, 2010.

[13] D. Krass and A. Ovchinnikov. Constrained group balancing: Why does it work. *European Journal of Operational Research*, 206(1):144–154, 2010.

[14] B. Krauss, J. Lee, and D. Newman. Optimizing the assignment of students to classes in an elementary school. *INFORMS Transactions on Education*, 14(1):39–44, 2013.

[15] B. A. McCarl and T. H. Spreen. Applied mathematical programming using algebraic systems. *Cambridge, MA*, 1997.

[16] J. Mingers and F. A. O'Brien. Creating student groups with similar characteristics: a heuristic approach. *Omega*, 23(3):313–321, 1995.

[17] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[18] G. Steiner and J. S. Yeomans. Optimal level schedules in mixed-model, multi-level jit assembly systems with pegging. *European Journal of Operational Research*, 95(1):38–52, 1996.

[19] M. Taboga. *Lectures on probability theory and mathematical statistics*. CreateSpace Independent Pub., 2012.

[20] D. Takači, M. Marić, G. Stankov, and A. Djenić. Efficiency of using vns algorithm for forming heterogeneous groups for cscl learning. *Computers and Education*, 109:98 – 108, 2017.

Appendix A

Number of partitions

Denote by $F_{n,n_1,n_2,\ldots,n_k}$ the number of ways to arrange n objects in k groups, where group g_l contain n_l objects ($n = \sum_{l=1}^{k} n_l$). In next sequential procedure [19] we derive the number $F_{n,n_1,n_2,\ldots,n_k}$.

1. First, we assign n_1 objects to the g_1 group. The number of possible ways to choose n_1 of the n objects is equal to the number of combinations of n_1 elements from n. So, number of possible ways to form group g_1 is
$$\binom{n}{n_1} = \frac{n!}{n_1!(n-n_1)!}.$$

2. Then, we assign n_2 objects to the second group g_2. There were n objects, but n_1 have already been assigned to the first group g_1. So, there are $n-n_1$ objects left, that can be assigned to the second group. The number of possible ways to choose n_2 of the remaining $n-n_1$ objects is equal to the number of combinations of n_2 elements from $n-n_1$. So, number of possible ways to form group g_2 is
$$\binom{n-n_1}{n_2} = \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!},$$
and number of possible ways to form the first two groups g_1, g_2 is
$$\binom{n}{n_1}\cdot\binom{n-n_1}{n_2} = \frac{n!}{n_1!(n-n_1)!}\cdot\frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} = \frac{n!}{n_1!n_2!(n-n_1-n_2)!}.$$

3. An so on, until we are left with n_k objects and the last group g_k. There is only one way to form the last group, which can also be written as
$$\binom{n-n_1-n_2-\cdots-n_{k-1}}{n_k} = \frac{(n-n_1-n_2-\cdots-n_{k-1})!}{n_k!(n-n_1-n_2-\cdots-n_k)!}.$$

Therefore, multiplying these factors we get

$$\begin{aligned}
&\binom{n}{n_1}\cdot\binom{n-n_1}{n_2}\cdot\ \ldots\ \cdot\binom{n-n_1-n_2-\cdots-n_{k-1}}{n_k}\\
&=\frac{n!}{n_1!n_2!\cdot\ \ldots\ \cdot n_{k-1}!(n-n_1-n_2-\cdots-n_{k-1})!}\cdot\frac{(n-n_1-n_2-\cdots-n_{k-1})!}{n_k!(n-n_1-n_2-\cdots-n_k)!}\\
&=\frac{n!}{n_1!n_2!\cdot\ \ldots\ \cdot n_k!(n-n_1-n_2-\cdots-n_k)!}\\
&=\frac{n!}{n_1!n_2!\cdot\ \ldots\ \cdot n_k!0!}\\
&=\frac{n!}{n_1!n_2!\cdot\ \ldots\ \cdot n_k!}.
\end{aligned}\tag{19}$$

In BMWASP each group contains at least $\left\lfloor \frac{n}{k} \right\rfloor$ and at most $\left\lceil \frac{n}{k} \right\rceil$ objects. If some groups contain the same number of objects then the labeling of these groups does not matter. Then we can label the group 1 to be group 2 and group 2 to be group 3 and so on. Denote by $\hat{k}$ the number of groups which contain $\left\lceil \frac{n}{k} \right\rceil$ objects, then $k-\hat{k}$ groups contain $\left\lfloor \frac{n}{k} \right\rfloor$ objects (if n is divisible by k then $\hat{k}=k$). We have total $\hat{k}!\cdot(k-\hat{k})!$ ways to label the groups. Thus, the final number of ways to arrange n objects in k groups with these conditions is:

$$F_{n,n_1,n_2,\ldots,n_k}=C_{n,k}=\frac{n!}{\left(\left\lceil \frac{n}{k} \right\rceil!\right)^{\hat{k}}\cdot\left(\left\lfloor \frac{n}{k} \right\rfloor!\right)^{(k-\hat{k})}\cdot\hat{k}!\ \cdot(k-\hat{k})!}.$$

Appendix B

Table 4: Computational results obtained with VNS algorithm

Instance	f_{best}	t_{avg}	σ	LB	Instance	f_{best}	t_{avg}	σ	LB
o25a5g10	**24.40**	0.10	0.00	24.40	o50a5g2	**5.00**	0.20	0.00	5.00
o25a10g10	**76.60**	0.30	0.43	76.60	o50a10g2	**4.00**	0.50	0.00	4.00
o25a15g10	111.00	0.50	1.32	103.80	o50a15g2	**8.00**	0.90	1.01	8.00
o25a20g10	184.20	0.60	1.47	154.60	o50a20g2	**15.00**	1.30	1.75	15.00
o25a30g10	262.00	0.90	1.93	199.20	o50a30g2	34.00	1.80	2.75	26.00
o25a40g10	403.40	1.20	1.93	300.80	o50a40g2	61.00	2.70	4.68	39.00
o25a50g10	521.20	1.30	1.52	371.80	o50a50g2	93.00	3.00	5.45	53.00
o25a60g10	615.20	1.70	2.83	416.00	o50a60g2	133.00	3.70	7.27	77.00
o25a70g10	767.20	2.00	3.15	445.60	o50a70g2	144.00	4.60	7.38	60.00
o25a80g10	744.60	2.30	3.47	466.60	o50a80g2	193.00	5.20	8.78	79.00
o25a90g10	991.20	2.20	1.13	628.60	o50a90g2	233.00	5.80	10.90	99.00
o25a100g10	1073.60	2.80	4.69	684.40	o50a100g2	250.00	5.90	13.02	108.00
o50a5g5	**19.20**	0.30	0.00	19.20	o50a5g10	**31.00**	0.30	0.00	31.00
o50a10g5	**37.60**	0.70	0.00	37.60	o50a10g10	**62.60**	0.80	0.00	62.60
o50a15g5	**44.80**	1.80	1.65	44.80	o50a15g10	**108.00**	2.30	1.66	108.00
o50a20g5	**75.20**	2.30	1.63	75.20	o50a20g10	155.80	2.80	3.11	146.00
o50a30g5	147.20	3.30	3.95	111.20	o50a30g10	267.80	4.20	5.24	196.60
o50a40g5	190.00	4.40	5.13	138.40	o50a40g10	397.40	5.70	8.03	293.20
o50a50g5	260.00	5.80	7.66	153.60	o50a50g10	548.00	7.20	6.32	379.20
o50a60g5	369.20	6.60	7.43	200.80	o50a60g10	622.00	7.70	6.74	393.20
o50a70g5	458.40	7.90	8.43	240.80	o50a70g10	816.60	8.90	8.21	471.40
o50a80g5	503.20	8.60	11.04	252.80	o50a80g10	961.80	11.20	7.30	589.20
o50a90g5	572.80	11.50	12.48	272.80	o50a90g10	1115.20	12.40	9.57	651.40
o50a100g5	644.00	11.50	13.14	321.60	o50a100g10	1222.80	13.00	9.33	704.40
o50a5g15	**72.27**	0.30	0.00	72.27	o50a5g20	**81.90**	0.30	0.00	81.90
o50a10g15	**70.40**	1.30	1.34	70.40	o50a10g20	**141.30**	0.70	0.00	141.30
o50a15g15	175.47	2.50	3.12	169.60	o50a15g20	246.20	2.20	1.24	227.20
o50a20g15	203.33	3.00	3.11	174.67	o50a20g20	294.10	3.00	2.87	262.00
o50a30g15	360.00	4.10	8.03	286.13	o50a30g20	536.40	4.00	3.87	410.60
o50a40g15	654.67	6.00	10.17	485.07	o50a40g20	682.40	6.70	3.74	493.30
o50a50g15	802.67	8.80	7.83	570.67	o50a50g20	914.20	7.60	3.95	594.50
o50a60g15	905.47	8.60	7.32	614.40	o50a60g20	1237.90	9.90	7.96	879.50
o50a70g15	1161.07	9.40	9.61	792.53	o50a70g20	1293.60	8.80	4.80	895.60
o50a80g15	1285.20	11.80	9.91	830.67	o50a80g20	1809.20	9.80	7.93	1152.00
o50a90g15	1540.67	12.40	10.87	978.13	o50a90g20	1878.90	13.50	7.06	1184.90
o50a100g15	1765.73	14.10	7.31	1101.33	o50a100g20	2341.10	13.40	10.66	1499.60
o50a5g25	**57.12**	0.40	0.00	49.44					
o50a10g25	97.36	1.20	0.82	70.24					
o50a15g25	195.76	2.20	1.60	98.56					
o50a20g25	312.00	2.80	1.49	162.24					
o50a30g25	578.24	3.50	3.64	226.88					
o50a40g25	677.12	4.70	3.06	319.68					
o50a50g25	1016.24	6.00	2.57	545.76					
o50a60g25	1252.24	6.80	4.37	600.64					
o50a70g25	1303.28	8.80	3.74	488.80					
o50a80g25	1605.84	10.50	2.52	706.88					
o50a90g25	2028.08	11.00	3.70	900.32					
o50a100g25	2286.08	11.70	4.23	1016.00					

Table 5: Computational results obtained with VNS algorithm

Instance	f_{best}	t_{avg}	σ	LB	Instance	f_{best}	t_{avg}	σ	LB
o100a5g2	**6.00**	0.70	0.00	6.00	o100a5g5	**16.80**	1.20	0.00	16.80
o100a10g2	**10.00**	1.50	0.00	10.00	o100a10g5	**32.80**	3.30	0.00	32.80
o100a15g2	**4.00**	3.30	0.00	4.00	o100a15g5	**50.40**	7.70	0.39	50.40
o100a20g2	**31.00**	4.10	0.00	31.00	o100a20g5	54.80	13.80	1.83	45.60
o100a30g2	34.00	9.70	3.02	32.00	o100a30g5	108.40	24.00	3.70	77.60
o100a40g2	56.00	13.20	5.24	38.00	o100a40g5	207.60	30.90	6.81	144.00
o100a50g2	82.00	18.50	4.32	52.00	o100a50g5	271.20	43.70	7.12	160.80
o100a60g2	124.00	21.90	6.52	66.00	o100a60g5	340.40	49.40	9.09	149.60
o100a70g2	156.00	23.60	7.18	76.00	o100a70g5	493.20	52.30	15.45	220.80
o100a80g2	184.00	34.30	9.42	64.00	o100a80g5	597.60	78.40	16.73	281.60
o100a90g2	227.00	37.80	10.18	101.00	o100a90g5	647.20	80.40	17.91	305.60
o100a100g2	273.00	41.90	15.47	111.00	o100a100g5	714.40	86.90	16.34	276.80
o100a5g10	**17.00**	1.60	0.00	17.00	o100a5g15	**66.13**	1.50	0.00	66.13
o100a10g10	**57.20**	5.10	0.00	57.20	o100a10g15	**82.67**	7.10	0.80	82.67
o100a15g10	**107.00**	12.10	1.25	107.00	o100a15g15	145.07	16.40	1.94	141.87
o100a20g10	140.40	17.70	4.88	122.80	o100a20g15	195.73	24.90	5.03	176.27
o100a30g10	255.20	28.80	6.31	192.20	o100a30g15	386.53	35.10	6.50	303.47
o100a40g10	397.60	43.70	8.71	252.20	o100a40g15	628.40	51.60	14.61	427.20
o100a50g10	519.80	51.90	16.10	324.40	o100a50g15	795.33	60.60	18.09	516.53
o100a60g10	783.80	55.70	16.62	428.40	o100a60g15	1040.67	79.20	19.79	624.00
o100a70g10	937.40	77.00	12.56	511.00	o100a70g15	1208.67	78.40	18.39	720.53
o100a80g10	1077.20	100.10	22.55	507.60	o100a80g15	1280.13	111.10	18.20	670.67
o100a90g10	1239.80	86.20	18.95	626.20	o100a90g15	1668.00	116.00	29.06	830.40
o100a100g10	1395.00	111.60	17.58	687.80	o100a100g15	2096.67	126.90	25.68	1085.87
o100a5g20	**82.60**	1.50	0.00	82.60	o100a5g25	**45.12**	1.70	0.00	45.12
o100a10g20	**152.20**	4.70	0.00	152.20	o100a10g25	**133.28**	7.80	0.85	133.28
o100a15g20	**275.20**	12.60	0.51	275.20	o100a15g25	198.72	22.40	4.56	177.92
o100a20g20	348.60	25.40	3.77	337.00	o100a20g25	343.04	28.10	9.49	276.48
o100a30g20	492.60	38.90	7.73	415.50	o100a30g25	555.68	42.70	13.00	362.40
o100a40g20	802.40	47.70	11.37	621.70	o100a40g25	824.80	54.90	11.71	468.64
o100a50g20	901.80	62.40	9.44	646.30	o100a50g25	989.68	66.70	9.19	543.68
o100a60g20	1303.70	76.70	17.36	908.30	o100a60g25	1502.96	78.60	13.07	692.16
o100a70g20	1583.30	96.70	10.80	1063.00	o100a70g25	1756.72	107.50	19.05	886.56
o100a80g20	1837.50	98.40	25.65	1229.10	o100a80g25	2002.72	128.60	16.30	875.68
o100a90g20	2431.40	91.20	14.24	1568.10	o100a90g25	2196.96	117.70	26.05	1053.60
o100a100g20	2488.50	124.40	20.60	1524.60	o100a100g25	2598.32	135.70	25.52	1259.52
o100a5g30	**102.73**	1.60	0.00	102.73					
o100a10g30	**267.67**	4.90	0.00	267.67					
o100a15g30	364.20	17.70	2.41	354.13					
o100a20g30	439.60	32.20	2.75	419.53					
o100a30g30	756.80	40.30	8.48	645.60					
o100a40g30	1265.47	45.30	9.22	1018.00					
o100a50g30	1510.60	72.90	12.30	1184.93					
o100a60g30	1901.20	89.70	15.54	1395.87					
o100a70g30	2365.07	88.10	13.36	1723.20					
o100a80g30	2695.40	104.60	16.40	1899.20					
o100a90g30	3075.93	111.00	13.49	2141.67					
o100a100g30	3714.67	129.00	21.48	2582.47					

Table 6: Computational results obtained with VNS algorithm

Instance	f_{best}	t_{avg}	σ	LB	Instance	f_{best}	t_{avg}	σ	LB
o250a5g2	**6.00**	4.00	0.00	6.00	o250a5g5	**13.60**	126.80	0.00	13.60
o250a10g2	**7.00**	9.00	0.00	7.00	o250a10g5	**29.60**	17.60	0.00	29.60
o250a15g2	**14.00**	16.90	0.00	14.00	o250a15g5	**35.20**	46.80	0.00	35.20
o250a20g2	**19.00**	28.80	0.00	19.00	o250a20g5	46.80	89.00	1.80	44.00
o250a30g2	28.00	68.70	2.18	24.00	o250a30g5	126.40	162.90	3.81	105.60
o250a40g2	67.00	91.00	3.01	59.00	o250a40g5	181.60	210.50	5.51	128.00
o250a50g2	76.00	130.00	3.91	52.00	o250a50g5	301.20	259.90	8.78	199.20
o250a60g2	112.00	159.70	6.97	62.00	o250a60g5	372.40	364.60	14.05	211.20
o250a70g2	142.00	177.90	9.05	78.00	o250a70g5	452.80	417.60	15.83	207.20
o250a80g2	176.00	236.30	10.68	82.00	o250a80g5	556.40	512.80	15.59	212.00
o250a90g2	227.00	236.40	12.12	105.00	o250a90g5	664.80	651.20	19.56	268.80
o250a100g2	248.00	267.90	14.09	94.00	o250a100g5	812.80	612.00	23.35	292.00
o250a5g10	**36.20**	8.50	0.00	36.20	o250a5g15	**37.87**	11.00	0.00	37.87
o250a10g10	**64.80**	25.00	0.00	64.80	o250a10g15	**83.47**	28.30	0.00	83.47
o250a15g10	**81.80**	93.30	0.54	81.80	o250a15g15	**130.40**	93.10	1.67	130.40
o250a20g10	144.60	131.80	2.48	137.40	o250a20g15	239.33	160.70	5.29	223.73
o250a30g10	292.80	238.40	8.19	231.00	o250a30g15	412.80	249.50	9.06	333.07
o250a40g10	473.40	304.30	9.85	325.40	o250a40g15	562.00	348.30	11.29	347.73
o250a50g10	629.80	333.80	15.13	359.60	o250a50g15	1013.33	401.60	19.06	588.80
o250a60g10	779.00	488.80	19.98	400.40	o250a60g15	1074.93	644.00	24.33	537.60
o250a70g10	973.80	578.10	25.49	468.40	o250a70g15	1525.87	720.90	31.88	814.93
o250a80g10	1093.20	638.40	23.79	506.60	o250a80g15	1641.47	761.90	36.33	778.40
o250a90g10	1402.80	801.10	30.41	629.80	o250a90g15	2056.53	834.60	40.35	929.60
o250a100g10	1622.80	958.10	36.56	687.60	o250a100g15	2433.47	897.70	38.77	1114.67
o250a5g20	**65.00**	11.60	0.00	65.00	o250a5g25	**79.68**	12.10	0.00	79.68
o250a10g20	**120.40**	30.60	0.00	120.40	o250a10g25	**203.20**	28.80	0.00	203.20
o250a15g20	**219.90**	95.50	0.80	219.90	o250a15g25	190.40	141.50	3.00	185.76
o250a20g20	261.90	161.90	4.47	239.60	o250a20g25	337.44	184.80	6.31	305.60
o250a30g20	462.90	283.60	12.35	332.00	o250a30g25	664.48	282.10	9.46	541.60
o250a40g20	795.80	335.20	16.74	523.40	o250a40g25	1028.08	368.30	13.91	652.64
o250a50g20	1003.70	373.30	17.91	540.80	o250a50g25	1378.80	455.00	30.78	842.72
o250a60g20	1478.60	542.20	30.94	779.00	o250a60g25	1636.88	607.50	20.47	810.56
o250a70g20	1798.10	612.80	33.71	885.10	o250a70g25	2081.36	631.40	38.35	952.48
o250a80g20	2367.30	825.40	48.02	1142.60	o250a80g25	2784.80	802.80	33.71	1300.80
o250a90g20	2420.20	883.20	45.36	1124.00	o250a90g25	3064.48	992.40	39.99	1346.40
o250a100g20	2996.10	1050.30	36.95	1344.00	o250a100g25	3684.72	990.60	41.91	1557.44
o250a5g30	**86.53**	13.30	0.00	86.53					
o250a10g30	**215.73**	27.90	0.00	215.73					
o250a15g30	232.60	143.10	2.99	228.33					
o250a20g30	465.13	207.40	7.05	433.60					
o250a30g30	879.00	321.70	16.60	709.80					
o250a40g30	1186.67	358.90	22.56	751.33					
o250a50g30	1401.73	547.90	31.35	716.73					
o250a60g30	1805.40	697.60	25.97	877.60					
o250a70g30	2611.07	621.10	32.01	1255.67					
o250a80g30	2822.53	923.70	39.25	1361.53					
o250a90g30	3431.80	1043.60	52.68	1655.80					
o250a100g30	4153.60	1166.00	46.11	1784.53					