RAIRO-Oper. Res. 56 (2022) 1167–1185
https://doi.org/10.1051/ro/2022048

RAIRO Operations Research
www.rairo-ro.org

HYBRID MATHEURISTICS FOR THE MULTI-CAPACITATED CLUSTERING PROBLEM

Kennedy Anderson Gumarães de Araújo[1], Jedson Bernadino Guedes[2] and Bruno de Athayde Prata[3,*]

Abstract. The capacitated clustering problem is a well-known and largely studied combinatorial optimization problem with several industrial applications. Although a great attention has been paid to this problem in the literature, the deeming of the problem with clusters centers with multiple types and a unique capacity per type is quite limited. We introduce a novel variant of capacitated clustering problems named multi-capacitated clustering problem (MCCP), a NP-hard optimization problem in which there are clients with different types and units of services to offer that must be grouped into given centers that demand with limited capacity per type the services. It is taken into account the distance between each one of these clients and the potential clusters to which they can be allocated, aiming to minimize the sum of such distances. It is presented an integer programming model for this problem, which it is shown to have limited application solving large-sized instances. As solution procedures, we present the following algorithms. We propose a greedy heuristic to generate a tentative feasible solution within a negligible computational effort. We adapt a size-reduction (SR) matheuristic to solve the problem under study. Furthermore, we introduce an innovative matheuristic that hybridizes the constructive phase of the well-known GRASP metaheuristic with the SR algorithm. Also, we develop a variable fixing (VF) heuristic. Finally, we propose a hybrid matheuristic based on the SR and VF algorithms. Computational results over a set of 100 randomly generated test instances point out the quality of the solutions found by the proposed algorithms. Besides, the results are statistically tested, and thus, our proposals are recommended to solve the problem under study.

Mathematics Subject Classification. 90C11, 90C27, 90C59.

Received May 25, 2021. Accepted March 23, 2022.

1. Introduction

The capacitated clustering problem (CPP) is a hard combinatorial optimization problem widely studied in the last few decades. Firstly addressed by Mulvey and Beck [19], the CCP has several real-world applications in industry and services. Some examples that can be mentioned are the grouping of clients into capacitated vehicle

Keywords. Clustering, mixed-integer linear programming, combinatorial optimization, approximation methods and heuristics.

[1] Department of Applied Mathematics, University of São Paulo, São Paulo, Brazil.
[2] Department of Statistics and Applied Mathematics, Federal University of Ceará, Fortaleza, Brazil.
[3] Department of Industrial Engineering, Federal University of Ceará, Fortaleza, Brazil.
*Corresponding author: baprata@ufc.br

routes, partitioning of nodes in distributed networks, investment analysis, garbage collection, dengue disease combat, and newspaper delivery [11, 13, 19, 20].

In this paper, we consider that are clients with different types of demand/offer. Furthermore, it is also taken into account in this work the fact that there might exist several possibilities of choosing a certain amount of clusters, among the candidates. Thus, considering the situation just described here, we face a multi-capacitated clustering problem (MCCP). MCCP is different from capacitated p-median and capacitated cluster problems since the capacities may be different between types and medians.

The utility of solving a MCCP problem is huge for theoretical and practical purposes, since it may be applied in different situations in order to reduce costs and dispose of feasible solutions in reasonable time for planning reasons.

There are many real-world situations in which we can see a multi-capacitated clustering problem. First, let us think of the case we desire to assign people to a certain hospital among the existing options, aiming to minimize the distance between the people and the buildings in order to get a better response time in case of a health emergency. Each hospital has a different capacity for different types of patient. In fact, a certain hospital might not be able to receive a specific type of patient at all, for example, a children's hospital, or a hospital exclusive for cancer or mental illness. A quite similar example is in the educational area.

The MCCP could also be adapted to deal with nodes partitioning in a network of distributed computing in which each cluster has its own capacity of processing that could be different from the others for different types of applications. In this case, the clients could be the type of service required for each node, like video rendering, huge calculations, or simulating stochastic scenarios. Another example would be an application for oil refineries. Each refinery has different capacities to produce different types of oil derivatives in an specific quantity. It is really useful to find out how to best assign the oil by-products demands among the existing refineries facilities.

Figure 1 presents an example of situation one could define as MCCP. In this example, several clients of two different types must be assigned to two clusters. The cluster 1 is able to receive at most six blue clients and four brown clients, whilst the cluster 2 has capacity for only two clients of type blue, and seven for clients of type brown.

The main contributions of this paper are fourfold. First, we present an extended mathematical formulation for the MCCP. Second, we investigate some properties of the problem. Third, we propose some innovative solution procedures to solve the problem under study. We propose a greedy heuristic trying to generate a feasible solution within a negligible computational effort. We adapt a size-reduction (SR) matheuristic to solve the problem under study. Furthermore, we introduce an innovative matheuristic that hybridizes the well-known GRASP algorithm with the SR algorithm. Also, we develop a variable fixing heuristic (VF), which is an extension of the matheuristic presented by [17]. Finally, we propose a hybrid matheuristic based on the SR and VFH algorithms. Fouth, we perform extensive computational experimentation with randomly generated test instances.

The remainder of this paper is organized as follows: in Section 2, we present the literature review, in Section 3, we present the proposed mathematical formulation as well as some problem properties, in Section 4, we propose some size-reduction algorithms for the problem under study; in Section 5, we discuss some results from computational experiments; finally, in Section 6 we draw some conclusions and suggestions for future works.

2. Related approaches

Taking into consideration the possibility of a CCP to have clusters with capacities for each different type of client leads to a problem with significant industrial applications. Despite its practical relevance, such assumption has not been presented. As reported by Negreiros *et al.* [21], Prata [23] introduced the MCCP, presenting the description of the problem and a mixed integer programming formulation for it. Nevertheless, computational experiments aiming to evaluate its complexity, and an also algorithm to deal with it have not been reported in the revised literature.

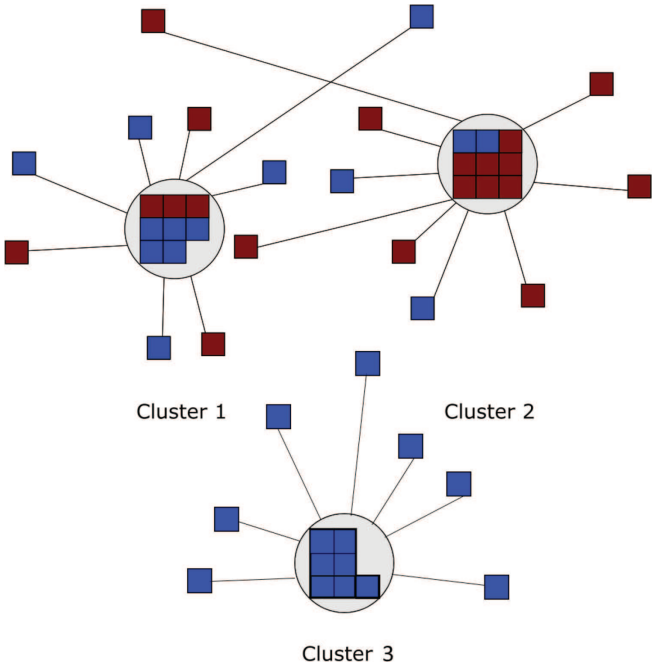

FIGURE 1. A feasible solution for the MCCP.

There are some studies that are closely related to the multi-capacitated clustering problem. Undoubtedly, the one most similar to MCCP is the capacitated clustering problem (CCP), which was firstly proposed by Mulvey and Beck [19]. With regard the CCP, several approaches have been presented in the literature.

Osman and Christofides [22] developed a hybrid simulated annealing and tabu search (SATS) algorithm to solve the CCP. An initial solution is generated by a constructive heuristic, which is improved by the SATS procedure. Using this hybrid procedure, the solution in a given iteration is always better or equal to that produced at any previous iteration. França *et al.* [11] presented a new mixed-integer linear programming formulation for the CCP. Besides, an adaptive tabu search (TS) is proposed as a solution procedure. The adaptive TS incorporates special characteristics, which gave the search the ability to guide the intensification and diversification purposes automatically. Shieh and May [25] presented a binary genetic algorithm (GA) with standard operators and an adaptive penalty function to handle the constraints related to the clusters' capacities. Ahmadi and Osman [1] developed a hybrid of greedy randomized adaptive search procedure (GRASP) and adaptive memory programming (AMP) for solving the CCP, called GRAMPS. A learning process returns information on the best components in an elite set of GRAMPS solutions. Scheurer adn Wendolsky [24] presented a standard scatter search metaheuristic to solve the CCP.

Geetha and Vanathi [13] developed a hybrid k-means algorithm using priority as a measure that directs the search for better solutions. This iterative procedure is used to allocate the items to the clusters. Deng and Bard [8] presented a reactive GRASP with path relinking (PR). In the first phase, two distinct algorithms are used to build a feasible initial solution. The selection of elements in the restricted candidate list is based on an adaptive procedure. In the second phase, three neighborhoods are explored in the local search procedures. Finally, a PR algorithm is used as a post-processing phase.

Martinez-Gavara *et al.* [17] developed a GRASP and an iterated greedy metaheuristics to solve the CCP. Besides, a matheuristic is proposed as a post-processing phase. Lai and Hao [15] proposed an iterated variable neighborhood search (IVNS) algorithm to solve the CCP that combines a variable neighborhood descent method with a randomized shake procedure. Thereby, this algorithm explores the search space effectively. Mai *et al.* [16],

developed a new metaheuristic that considers statistical models for the CPP. This new algorithm hybridizes an expectation-maximization algorithm with posterior regularization. The performance of the proposed approach is evaluated for the deterministic and stochastic variants. Brimberg *et al.* [2] proposed two types of variable neighborhood search (VNS) algorithms for the CCP. The first one is a standard VNS and the second one is a skewed VNS. Both algorithms outperformed all the other methods under comparison; however, the skewed VNS presented the best results. Zhou *et al.* [28] presented a tabu search and a memetic algorithm for the CCP. These algorithms presented competitive results in comparison to the existing state-of-art heuristics.

Stefanello *et al.* [26] proposed a matheuristic called iterated reduction matheuristic algorithm (IRMA) to solve the capacitated p-median problem, which is closely related to the CCP. This algorithm is an extension of the size-reduction heuristic introduced by Fanjul-Peyro and Ruiz [9]. IRMA presents three stages using two distinct size-reduction strategies, excluding iteratively decision variables of the mathematical model. The generated subproblems, which presented fewer integer variables, are solved by CPLEX. This approach outperforms the existing algorithms for the problem.

Yang *et al.* [27] introduced a new variant of the CCP in which a raw material from a set of suppliers is assigned to a set of potential places. As a solution procedure, a Lagrangian relaxation approach is presented. Negreiros and Palhano [20] presented a new variant for CCP named the capacitated centered clustering problem (CCCP), in which each cluster is composed of individuals from whom we can compute a center value and hence, determine a similarity measure. As a solution procedure, a two-phase algorithm is presented. Firstly a constructive heuristic is performed, which is followed by a VNS metaheuristic as a refinement phase. Chaves and Lorena [4] proposed a hybrid clustering search (CS) algorithm for the CCCP. This algorithm can identify promising areas of the search space, which are grouped into clusters and explored with local search heuristics. Chaves *et al.* [5] extended the previous CS algorithm hybridizing it with a GA. Thus, this hybrid metaheuristic outperformed the standard CS. Chaves *et al.* [6] proposed an adaptative biased random-key genetic algorithm (ABRKGA) for the CCCP. An innovative mechanism to control diversification and intensification is presented.

Chagas *et al.* [3] introduced a new variant of the clustering editing problem in which overlapping clusters are allowed. A hybrid heuristic to generate solutions to the proposed relaxation is proposed, as well as two metaheuristics and a mixed-integer linear programming model. Negreiros *et al.* [21] introduced four new variations of the Heterogeneous Capacitated Clustering Problems (HCCP), which are variants of the (CCP). The HCCP is applied to the layout of IT-Teams in a software factory. As a solution procedure, a hybrid multi-start metaheuristic with tabu search and path relinking is presented. Gnägi and Baumann [14] introduced a new matheuristic for the capacitated p-median problem. This solution procedure is composed of two phases: a global optimization phase, aiming to generate an initial feasible solution, and a local optimization phase, aiming to improve this initial solution. This matheuristic presented competitive results for the large-sized test instances. On the other hand, it requires seven parameters to be calibrated.

3. Problem statement

3.1. Mathematical formulation

In this section, we present a mixed-integer linear programming (MILP) formulation for the MCCP since it is a natural way to solve the problem under study. We find useful to present a mathematical programming model for the problem with the aim of assessing in Section 5 the quality of the heuristics proposed for small instances where feasible and/or optimal solutions can be found. Furthermore, the model is useful for us because we apply several matheuristics, as described in Section 4.

In the presented model, the clients have distinct characteristics, and each candidate to cluster has a certain capacity for each type of client. Let the following notation for the MCCP. We denote as M and N, the set of clients to be allocated and the set of candidates to cluster, respectively, along with the following notation:

- m: Cardinality of the set M, *i.e.*, number of clients.
- n: Cardinality of the set N, *i.e.*, number of cluster candidates.

– p: Number of clusters that must be presented in the solution.
– K: Number of different types of client.
– d_{ij}: Distance between the client i and the (candidate) cluster j.
– C_{jk}: The capacity of the (candidate) cluster j for the type of client k.
– q_{ik}: The quantity of demand/offer of type k of client i.

We also consider the following decision binary variables:

$$x_{ij} = \begin{cases} 1, & \text{if client } i \text{ is assigned to cluster } j; \\ 0, & \text{otherwise.} \end{cases} \qquad \forall i \in M,\ \forall j \in N,$$

$$y_j = \begin{cases} 1, & \text{if the candidate cluster } j \text{ is selected;} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall j \in N.$$

Thus, in mathematical notation, the MCCP can be expressed as follows.

$$(\text{MCCP})\ \min z = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}x_{ij}. \tag{3.1}$$

And the objective function above is subject to:

$$\sum_{j=1}^{n} y_j = p, \tag{3.2}$$

$$\sum_{i=1}^{m} x_{ij} \geq y_j, \qquad \forall\, j = 1, ..., n, \tag{3.3}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall\, i = 1, ..., m, \tag{3.4}$$

$$\sum_{i=1}^{m} q_{ik}x_{ij} \leq C_{jk}y_j, \qquad \forall\, j = 1, ..., n, \quad \forall\, k = 1, ..., K, \tag{3.5}$$

$$x_{ij} - y_j \leq 0, \qquad \forall\, i = 1, ..., m, \quad \forall\, j = 1, ..., n, \tag{3.6}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall\, i = 1, ..., m, \quad \forall\, j = 1, ..., n, \tag{3.7}$$

$$y_j \in \{0, 1\}, \qquad \forall\, j = 1, ..., n. \tag{3.8}$$

The objective function of our problem, (3.1), is to minimize the sum of the distances between the clients and the clusters to which they might be associated. The constraint (3.2) assures the number of clusters is p. The constraint set (3.3) assures a valid solution, by avoiding that a selected cluster receives no client. The equations (3.4) ensure that each client will be assigned to one, and only one, cluster. The set of constraints (3.5) avoids surpassing any capacity of a cluster. The constraints (3.6) ensure that a client will not be assigned to a non-selected cluster. The set of constraints (3.7) and (3.8) guarantee that the decision variables are binary. One can notice that this model differs from the one presented by Prata (2015) by the presence of the constraint set (3.3).

A Q matrix row, written Q_i, is a vector that describes the percentage of types which compose the client i. If a row of the matrix Q looks like the highlighted row in the Figure 2, $\begin{bmatrix}0.5 & 0.5 & 0\end{bmatrix}$, for example, it means half of that client is of the first type, half of it is of the second type, and none of it is of the third type.

For the purpose of this research, we consider that a client is of only one type. For this reason, a row of the matrix Q will have only one of its entry as 1, and all the rest as zeros. The column of the unique number one in such row will define the type of this client. As an illustration, consider: $\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}$, which tells us that this client is 100% of the second type, for example.

$$\begin{bmatrix} & \vdots & \\ 0 & 0.3 & 0.7 \\ 0.5 & 0.5 & 0 \\ 0.2 & 0.4 & 0.4 \\ & \vdots & \end{bmatrix}$$

FIGURE 2. A portion of a Q matrix.

The second parameter, n, the number of candidates to become a cluster, is what determines the most the size of an instance.The greater the p, the smaller the number of possible situations. When p is equal to n, it means there is no need to choose what clusters will be in the solution. When $k = 1$ we face a CCP, capacitated clustering problem. We can observe that the MCCP complexity is NP-Hard as the capacitated p-Median Problem is also NP-Hard [12].

3.2. Lower bound

If at the constraint (3.3) it is considered an inequation of the type "less than or equal to", the number of possible cases will grow to $mtetimesp \times \sum_{r=1}^{p} \binom{n}{r}$. Since it doesn't make sense to have a solution with zero clusters, the summation starts on one. With this is mind, it is easy to see that trying to use a brute-force algorithm to test every single solution in order to find which one is the best is simply infeasible for large instances. Even using integer programming techniques and good softwares to find a solution in such scenarios take really long, as we show in the computational experience. Next, we present a lower bound for the problem under study.

Proposition 3.1. *A lower bound for the MCCP is given by Equation (3.9):*

$$\sum_{i=1}^{m} \min_{j=1,\ldots,n} d_{ij}. \tag{3.9}$$

Proof. The solution associated with the objective function value stated in Equation (3.9), is a solution such that each client is assigned to its closest cluster candidate, note that the number of cluster may be lower than p and the clusters capacities may be exceeded, since it is constructed in a greedy manner without consideration of these constraints. Directly, the minimum objective function value possible for a feasible solution is the lower bound stated in Equation (3.9). □

We can observe that the linear relaxation might possibly lead to better lower bounds than the proposed lower bound. However, the computational cost to solve the linear relaxation can be high. Even for the pure linear programming model, the resolution of large-sized instances is not trivial.

4. PROPOSED ALGORITHMS FOR THE MCCP

In the following subsections, we explain the three proposed matheuristics to solve the MCPP. Each matheuristic does not guarantee a feasible solution at the end. However, as presented in the computational experiments, the proposed matheuristics can return an integer solution to all instances evaluated, as reported in Section 5.

4.1. Greedy heuristic

Algorithm 1 presents the proposed constructive heuristic, which consists of the following steps: (i) selecting p distinct clusters candidates associated to the arcs with the lowest distances; and (ii) for each selected cluster, associating the closest client not yet selected without exceeding the cluster capacity, and repeat until all clients

are selected. Thereby, our procedure does not always guarantee the generation of feasible solutions. Although we could successfully find feasible solutions within negligible computational times for the complete test-bed set of instances used in this work. The first loop in the lines 3–7 takes $O(mnp)$ time and the second loop takes $O(m^2)$ time. The computational time complexity for the constructive heuristic is $O(mnp + m^2)$.

Algorithm 1. A greedy heuristic for the MCCP.

1: *input:* p, K, M, N, d, C, q *output:* feasible solution
2: **set** $\overline{N} \leftarrow N$; $P \leftarrow \emptyset$
3: **while** $(|P| < p)$ **do**
4: **set** $\bar{i}, \bar{j} \leftarrow \arg\min\{d_{ij} : i \in M, j \in \overline{N}\}$
5: $P \leftarrow P \cup \{\bar{j}\}$
6: $\overline{N} \leftarrow \overline{N} \setminus \{\bar{j}\}$
7: **end while**
8: **set** $j \leftarrow 1$; $x \leftarrow zeros(m, n)$; $\overline{M} \leftarrow M$; $cont \leftarrow 1$
9: **while** $\overline{M} \neq \emptyset$ **do**
10: **if** $(j = p)$ **then**
11: **set**: $j \leftarrow 1$
12: **end if**
13: **set**: $\bar{i} \leftarrow -1$
14: $\bar{i} \leftarrow \arg\min\{d_{ij} : i \in \overline{M},\ with\ \sum_{i' \in M} q_{i'k} x_{i'j} + q_{ik} \leq C_{jk},\ \forall k \in K\}$
15: **if** $\bar{i} \neq -1$ **then**
16: $x_{\bar{i}\bar{j}} \leftarrow 1$; $j \leftarrow j + 1$
17: $\overline{M} \leftarrow \overline{M} \setminus \{\bar{i}\}$
18: $cont \leftarrow 1$
19: **else**
20: $cont \leftarrow cont + 1$
21: **end if**
22: **if** $cont = p$ **then**
23: **return** *return no feasible solution found*
24: **end if**
25: **end while**
26: **return** x

4.2. Size-reduction heuristic (SR)

Fanjul-Peyro and Ruiz [9] introduced the size-reduction (SR) heuristic for the unrelated parallel machine scheduling problem. The SR is based on the idea that there is a small chance of a high-cost arc being included in a competitive solution. Therefore, in order to reduce the size of the problem and speed up its solution, a percentage of these arcs is eliminated before the beginning of the analysis.

The SR algorithm does not guarantee that the global optimal solution is found for a given instance and may not present high-quality solutions in some cases. However, this method can offer a fast and straightforward implementation at a low computational cost. The proposed SR heuristic consists of, given a parameter α, with $0 < \alpha \leq 1$, removing from the problem the arcs $\overline{ij}$, such that $d_{\overline{ij}} > \alpha \cdot \max\{d_{ij} : i \in M, j \in N\}$. Algorithm 2 describes an overview of the proposed SR, which is an extension of the algorithm presented by Stefanello *et al.* [26].

4.3. Greedy randomized size-reduction heuristic (GRSR)

Since greedy heuristics usually cannot escape from local optima, we can conclude that a greedy exclusion of arcs, in a traditional size-reduction approach, can also present difficulties for scaping from local optimal. In this sense, we can adapt the concepts of semi-greedy algorithms, as presented by Feo and Resende [10].

Algorithm 2. Size-reduction heuristic.

1: *input:* α, p, K, M, N, d, C, q *output:* feasible solution
2: **set** $ARCS \leftarrow \emptyset$; $d_{max} \leftarrow \max\{d_{ij} : i \in M, j \in N\}$
3: **for** $(i \in M,\ j \in N)$ **do**
4: **if** $(d_{ij} \leq \alpha \cdot d_{max})$ **then**
5: $ARCS \leftarrow ARCS \cup \{ij\}$
6: **end if**
7: **end for**
8: initiate the MCPP model considering only the arcs in $ARCS$
9: solve the model
10: **return** solution obtained from solving the model

We develop a new size-reduction approach based on the concepts of the well-known greedy randomized adaptive search procedures (GRASP). Our proposition, named greedy randomized size-reduction (GRSR) heuristic, works like the SR, although there is a probability of not removing the worst arcs. Algorithm 3 summarizes the proposed GRSR algorithm. The set ARCS compose the list of candidates to be used by the solution selected by a randomizing process indicated in line 4. In line 8, the model is solved with the selected arcs between clients and medians.

Algorithm 3. Greedy randomized size-reduction heuristic.

1: *input:* α, p, K, M, N, d, C, q *output:* feasible solution
2: **set** $ARCS \leftarrow \emptyset$; $d_{max} \leftarrow \max\{d_{ij} : i \in M, j \in N\}$
3: **for** $(i \in M,\ j \in N)$ **do**
4: **if** $(d_{ij} \leq \alpha \cdot d_{max} + rand(0,1) \cdot (d_{max} - \alpha \cdot d_{max}))$ **then**
5: $ARCS \leftarrow ARCS \cup \{ij\}$
6: **end if**
7: **end for**
8: initiate the mathematical model considering only the arcs in $ARCS$
9: solve the model
10: **return** solution obtained from solving the model

4.4. Variable fixing heuristic (VF)

Martinez-Gavara *et al.* [17] presented a hybrid approach for the CCP. First, the algorithm applies a GRASP – Iteration Greedy metaheuristic for the generation of an initial solution. Second, a matheuristic uses this initial solution to fix a subset of decision variables, improving the solution previously found. In this sense, we adapt the concepts of the algorithm proposed by [17] for the problem under study. Algorithm 4 describes the proposed variable fixing heuristic. Firstly, we use the greedy heuristic (Algorithm 1) as an initial solution. After that, we set as 1 in the mathematical model all the decision variables related to the arcs which the greedy heuristic presents in its solution. Thus, we solve the reduced model, with the variables previously fixed.

4.5. Hybrid size-reduction and variable fixing heuristic (HSRVF)

We propose a hybrid matheuristic which consists of combining the size-reduction and the variable fixing heuristics in one algorithm. We use the greedy heuristic for setting the decision variables that are fixed as 1 in the reduced model and the size-reduction approach for setting the decision variables as 0 in the reduced model. Algorithm 5 describes the proposed HSRVF matheuristic.

Firstly, we consider the greedy heuristic described in lines 2–7 of Algorithm 1 to create a feasible solution. Taking this initial solution into account, we set the allocation found in the mathematical model, fixing the corresponding decision variables in 1. Subsequently, we adopt the standard size-reduction heuristic, as described in lines 2–7 of Algorithm 2, to set as zero with the larger associated costs, considering the parameter α.

Algorithm 4. Variable fixing heuristic.

input: γ, p, K, M, N, d, C, q *output:* feasible solution
set $\overline{x} \leftarrow$ solution obtained *via* Algorithm 1; $ARCS_x \leftarrow \{ij : \overline{x}_{ij} = 1\}$; $ARCS_{fix} \leftarrow \emptyset$
while $(|ARCS_{fix}| < \lceil \gamma m \rceil)$ **do**
  set $\overline{ij} \leftarrow \{\arg\min\{d_{ij} : ij \in ARCS_x\}\}$
  $ARCS_x \leftarrow ARCS_x \setminus \{\overline{ij}\}$
  $ARCS_{fix} \leftarrow ARCS_{fix} \cup \{\overline{ij}\}$
end while
initiate the mathematical model
attribute the value 1 to the decision variables x_{ij}, where $ij \in ARCS_{fix}$
solve the model
return solution obtained from solving the model

Algorithm 5. Hybrid size-reduction and variable fixing heuristic.

input: α, γ, p, K, M, N, d, C, q *output:* feasible solution
set $\overline{x} \leftarrow$ solution obtained *via* Algorithm 1; $ARCS_x \leftarrow \{ij : \overline{x}_{ij} = 1\}$; $ARCS_{fix} \leftarrow \emptyset$
while $(|ARCS_{fix}| < \lceil \gamma m \rceil)$ **do**
  set $\overline{ij} \leftarrow \{\arg\min\{d_{ij} : ij \in ARCS_x\}\}$
  $ARCS_x \leftarrow ARCS_x \setminus \{\overline{ij}\}$
  $ARCS_{fix} \leftarrow ARCS_{fix} \cup \{\overline{ij}\}$
end while
set $ARCS \leftarrow \emptyset$; $d_{max} \leftarrow \max\{d_{ij} : i \in M, j \in N\}$
for $(i \in M,\ j \in N)$ **do**
  if $((d_{ij} \leq \alpha \cdot d_{max})$ **or** $(ij \in ARCS_x))$ **then**
    $ARCS \leftarrow ARCS \cup \{ij\}$
  end if
end for
initiate the mathematical model considering only the arcs in $ARCS$
attribute the value 1 to the decision variables x_{ij}, where $ij \in ARCS_{fix}$
solve the model
return solution obtained from solving the model

5. Computational results

5.1. Computational experience characteristics

Since test instances for the MCCP are not available, we generate our testbed with the following values: $m \in \{250, 500, 1000, 2000, 3000, 4000, 5000, 7000, 8000\}$ and $n \in \{m/5, 3m/10\}$, $p \geq \lceil m/4 \rceil$, and $k \in \{6, 7, 8, 9, 10\}$ We generate the distances and the capacities using the following uniform distribution $U[1, 100]$. Table 1 describes the characteristics of the 100 randomly generated test instances. All the evaluated test instances are available in this link. In Table 1 we use the following notation: M: number of clients, N: number of cluster candidates, K: number of different types of clusters, and p: number of clusters.

As the indicator for evaluation measure, we use the lower bound relative deviation (LBD) as a metric, Equation (5.1):

$$\text{LBD} = \frac{ip - lb}{lb} \times 100\%, \tag{5.1}$$

in which lb stands for the objective function lower bound value, and expressed in Equation (3.9), and ip stands for objective function value for the best solution found by some method.

We consider the following methods in our computational experiments:

- mixed integer linear programming model – MILP;
- greedy heuristic – GH;

TABLE 1. Summary of the proposed test instance characteristics.

Instance	M	N	p	K	Instance	M	N	p	K	Instance	M	N	P	K	Instance	M	N	p	K	Instance	M	N	p	K
MCCP1	250	50	12	6	MCCP21	1000	200	50	6	MCCP41	3000	600	150	6	MCCP61	5000	1000	250	6	MCCP81	7000	1400	350	6
MCCP2	250	75	18	6	MCCP22	1000	300	75	6	MCCP42	3000	900	225	6	MCCP62	5000	1500	375	6	MCCP82	7000	2100	525	6
MCCP3	250	50	12	7	MCCP23	1000	200	50	7	MCCP43	3000	600	150	7	MCCP63	5000	1000	250	7	MCCP83	7000	1400	350	7
MCCP4	250	75	18	7	MCCP24	1000	300	75	7	MCCP44	3000	900	225	7	MCCP64	5000	1500	375	7	MCCP84	7000	2100	525	7
MCCP5	250	50	12	8	MCCP25	1000	200	50	8	MCCP45	3000	600	150	8	MCCP65	5000	1000	250	8	MCCP85	7000	1400	350	8
MCCP6	250	75	18	8	MCCP26	1000	300	75	8	MCCP46	3000	900	225	8	MCCP66	5000	1500	375	8	MCCP86	7000	2100	525	8
MCCP7	250	50	12	9	MCCP27	1000	200	50	9	MCCP47	3000	600	150	9	MCCP67	5000	1000	250	9	MCCP87	7000	1400	350	9
MCCP8	250	75	18	9	MCCP28	1000	300	75	9	MCCP48	3000	900	225	9	MCCP68	5000	1500	375	9	MCCP88	7000	2100	525	9
MCCP9	250	50	12	10	MCCP29	1000	200	50	10	MCCP49	3000	600	150	10	MCCP69	5000	1000	250	10	MCCP89	7000	1400	350	10
MCCP10	250	75	18	10	MCCP30	1000	300	75	10	MCCP50	3000	900	225	10	MCCP70	5000	1500	375	10	MCCP90	7000	2100	525	10
MCCP11	500	100	25	6	MCCP31	2000	400	100	6	MCCP51	4000	800	200	6	MCCP71	6000	1200	300	6	MCCP91	8000	1600	400	6
MCCP12	500	150	37	6	MCCP32	2000	600	150	6	MCCP52	4000	1200	300	6	MCCP72	6000	1800	450	6	MCCP92	8000	2400	600	6
MCCP13	500	100	25	7	MCCP33	2000	400	100	7	MCCP53	4000	800	200	7	MCCP73	6000	1200	300	7	MCCP93	8000	1600	400	7
MCCP14	500	150	37	7	MCCP34	2000	600	150	7	MCCP54	4000	1200	300	7	MCCP74	6000	1800	450	7	MCCP94	8000	2400	600	7
MCCP15	500	100	25	8	MCCP35	2000	400	100	8	MCCP55	4000	800	200	8	MCCP75	6000	1200	300	8	MCCP95	8000	1600	400	8
MCCP16	500	150	37	8	MCCP36	2000	600	150	8	MCCP56	4000	1200	300	8	MCCP76	6000	1800	450	8	MCCP96	8000	2400	600	8
MCCP17	500	100	25	9	MCCP37	2000	400	100	9	MCCP57	4000	800	200	9	MCCP77	6000	1200	300	9	MCCP97	8000	1600	400	9
MCCP18	500	150	37	9	MCCP38	2000	600	150	9	MCCP58	4000	1200	300	9	MCCP78	6000	1800	450	9	MCCP98	8000	2400	600	9
MCCP19	500	100	25	10	MCCP39	2000	400	100	10	MCCP59	4000	800	200	10	MCCP79	6000	1200	300	10	MCCP99	8000	1600	400	10
MCCP20	500	150	37	10	MCCP40	2000	600	150	10	MCCP60	4000	1200	300	10	MCCP80	6000	1800	450	10	MCCP100	8000	2400	600	10

– greedy randomized size-reduction heuristic – GRSR ($\alpha = \{0.10\}$);
– hybrid size-reduction and variable fixing heuristic – HSRVF ($\alpha = 0.2, \gamma \in \{0.10, 0.25, 0.50, 0.75, 0.90\}$;
– size-reduction – SR ($\alpha \in \{0.10, 0.20, 0.30\}$), an adaptation of the algorithm proposed by [26];
– variable fixing heuristic – VF ($\gamma \in \{0.10, 0.25, 0.50, 0.75, 0.90\}$, an adaptation of the algorithm proposed by [17].

We implement all the proposed algorithms in C++ with Codeblocks IDE (http://www.codeblocks.org/). The source code for the models and algorithms used in this experiment can be accessed in this link. For the pure MILP model as well as the matheuristics the solver used was IBM ILOG CPLEX version 12.8. We perform the computational experience on a PC with Intel Core i7-8700 CPU 3.20 GHz and 32 GB memory. The operating system is the Ubuntu 18.04.1 LTS. For all methods under comparison, we adopt a time limit $t_{\text{lim}} = 600$ s. We execute all the evaluated methods one time, except by GRSR, which we execute 5 times. For each method, we present the average values for each class of instances.

5.2. Results for lower bound deviation

In Figures 3–6 we compare the evaluated methods taking into consideration the LBD performance indicator. We can observe that these graphs present y-axis in logarithmic scale and we use MILP and GH as reference. In general lines, the MILP method and GH are outperformed by all the proposed matheuristics.

With respect to the summarized results for MCCP instances in Figures 3–6 we can emphasize the following points. MILP method cannot find feasible integer solutions for the test instances of groups 5, 6, 7, 8, 9, and 10, given the specified time limit. In this context, the increase in the number of integer decision variables becomes prohibitive to the resolution for the pure MILP model. GH presents the worst results in the most evaluated test problems. However, GH produces better results than SR0.1 and GRSR for group 10 and then SR0.2 for group 9. This situation arises because the reduction of 10% and 20% in the number of integer decision variables is insufficient for the resolution of large-sized instances within the specified time limit. On the other hand, the SR0.3 fails to find feasible integer solutions for the test instances of groups 8, 9, and 10. All the VF-based algorithms present a similar behavior, although the VF075 has presented slightly worse results. GRASP-reduction presents competitive results, except for the test instances of group 10. All the hybrid size-reduction and variable fixing heuristics present similar results, except HSRVF01, which cannot produce feasible solutions for the test instances

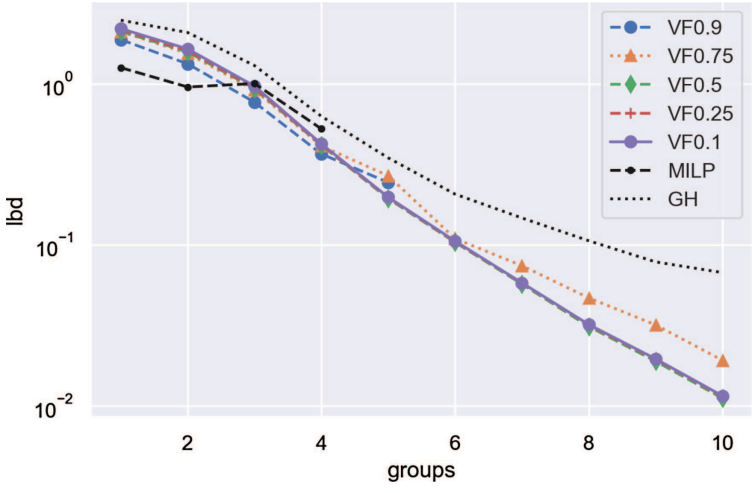

FIGURE 3. Lower bound deviation for variable fixing heuristics.

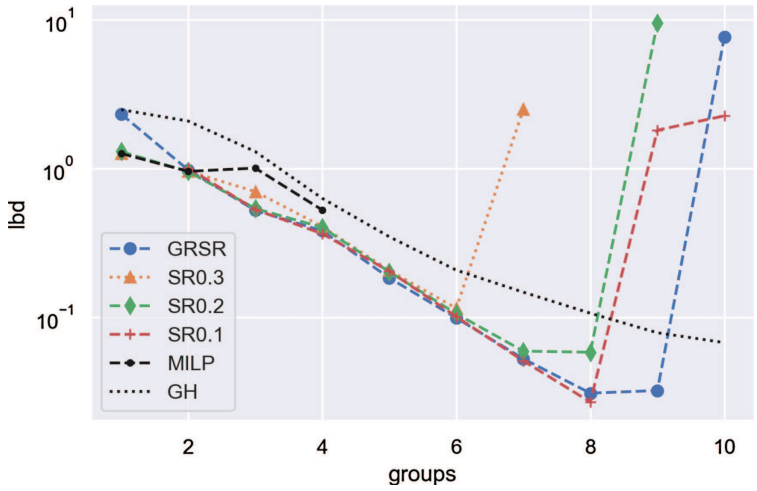

FIGURE 4. Lower bound deviation for greedy randomized size-reduction heuristic and size-reduction heuristics.

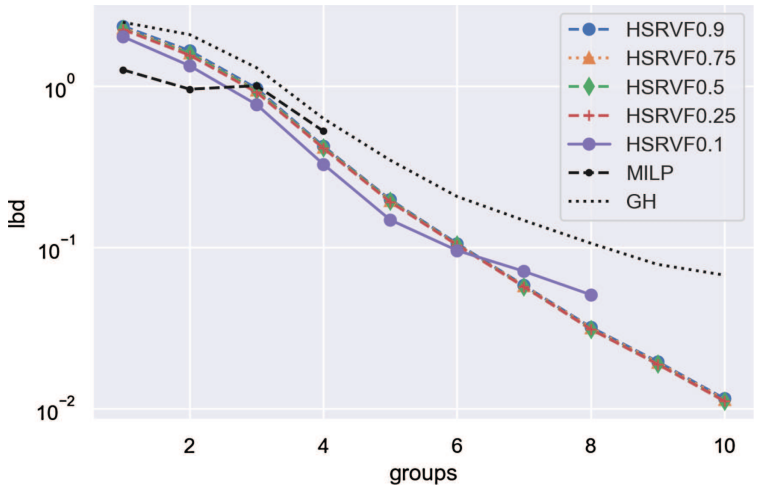

FIGURE 5. Lower bound deviation for hybrid size-reduction and variable fixing heuristics.

groups 9 and 10. Taking into consideration the four heuristics that found more feasible solutions (VF05, SR01, HSRVF025, and GH), VF025 and HSRVF025 present better results.

5.3. Results for number of instances with at least one solution found

Since the MCCP is a hard combinatorial optimization problem, the methods under comparison do not always find feasible solutions in the stipulated time limit. In Table 2 as well as in Figures 7–9 we analyze the behavior of all the methods under comparison in terms of the number of feasible solutions found. We can observe that the MILP model fails for obtaining feasible solutions for the test instances of the groups 5, 6, 7, 8, 9, and 10. Taking into consideration the variable-fixing heuristics, we can highlight that the VF01 algorithm presents the worst results. Also, the algorithms VF025, SR01, and HSRVF025 return at least a feasible solution in all of their executions.

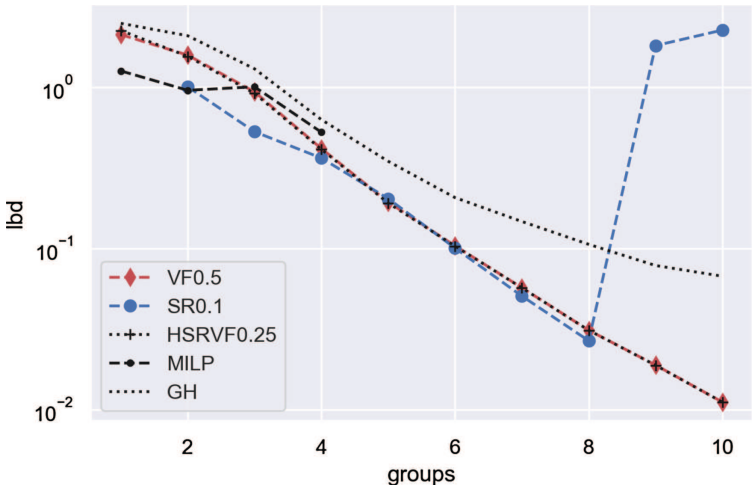

FIGURE 6. Lower bound deviation for the four heuristics that found more feasible solutions.

TABLE 2. Number of instances with at least one solution found.

Group	VF0.1	VF0.25	VF0.5	VF0.75	VF0.9	HSRVF0.1	HSRVF0.25	HSRVF0.5	HSRVF0.75	HSRVF0.9	GRSR	SR0.3	SR0.2	SR0.1	MILP
1	10	10	10	10	10	10	10	10	10	10	10	10	10	0	10
2	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
3	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
4	10	10	10	10	10	10	10	10	10	10	10	10	10	10	6
5	5	5	10	10	10	10	10	10	10	10	10	10	10	10	0
6	0	9	10	10	10	7	10	10	10	10	10	9	10	10	0
7	0	10	10	10	10	5	10	10	10	10	10	6	10	10	0
8	0	10	10	10	10	3	10	10	10	10	10	0	5	10	0
9	0	10	10	10	10	0	10	10	10	10	5	0	2	10	0
10	0	4	10	10	10	0	10	10	10	10	5	0	0	10	0

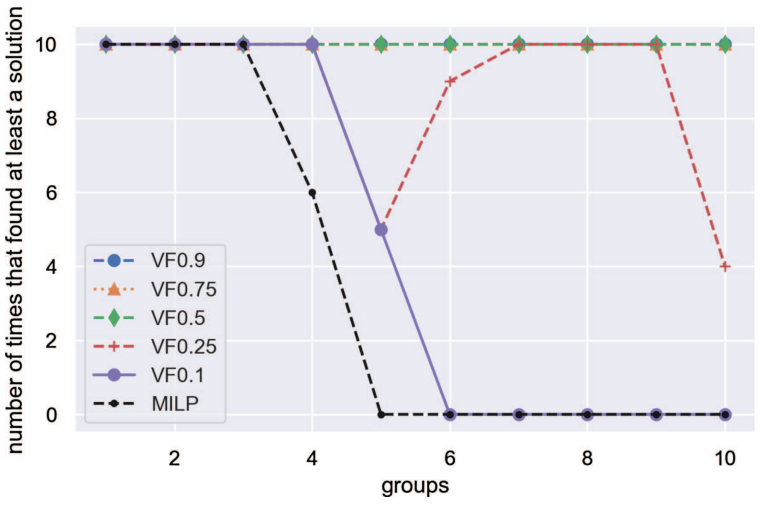

FIGURE 7. Number of instances with solutions found by variable fixing heuristics.

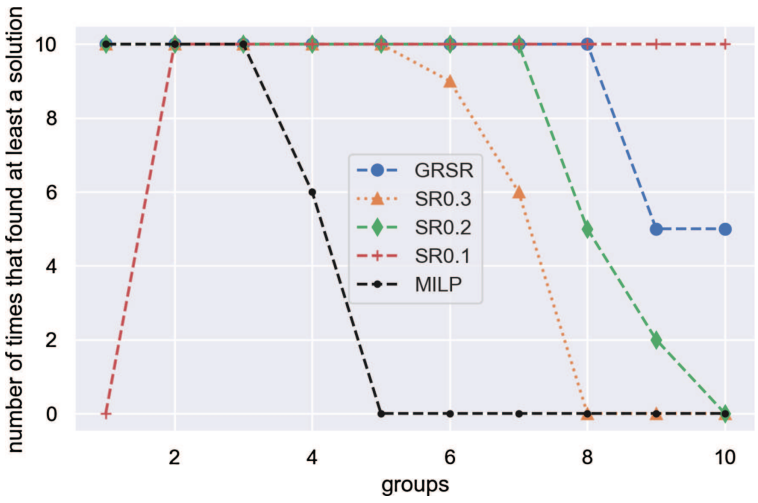

FIGURE 8. Number of instances with solutions found by greedy randomized size-reduction heuristic and size reduction heuristics.

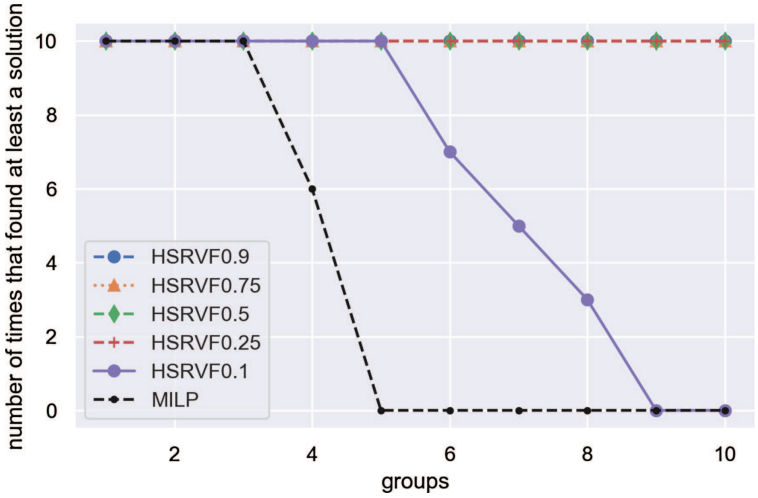

FIGURE 9. Number of instances with solutions found by hybrid size reduction and variable fixing heuristics.

5.4. Analysis of variance

Aiming to evaluate if the difference between the methods under comparison is statistically significant, we perform an analysis of variance – ANOVA approach [18]. We did not perform a normalization of the data. For treating the missing data (that is, the situation in which a given method did not return a feasible solution), we adopt a listwise exclusion approach. Tables 3 and 4 present, respectively, the average computational times and the average LBD's for all the evaluated methods in each set of test instances. In these tables, the symbol * stands for infeasibility. Figures 10–12 present, respectively, the computational times for the variable fixing, greedy randomized size-reduction, and hybrid size-reduction and variable fixing methods.

TABLE 3. Average computational times for all methods in each set of instances.

Group	VF0.1	VF0.25	VF0.5	VF0.75	VF0.9	HSRVF0.1	HSRVF0.25	HSRVF0.5	HSRVF0.75	HSRVF0.9	GRSR	SR0.3	SR0.2	SR0.1	MILP
1	0.49	0.02	0.02	0.02	0.01	0.12	0.01	0.01	0.00	0.00	15.83	47.22	27.81	*	262.43
2	6.49	1.14	0.08	0.07	0.06	1.00	0.11	0.02	0.01	0.01	600.35	600.04	600.23	601.04	600.17
3	125.31	18.50	1.95	0.33	0.27	23.85	0.97	0.18	0.06	0.04	600.07	600.09	600.06	600.03	600.62
4	419.77	81.96	5.23	1.81	1.34	269.96	5.82	0.44	0.28	0.21	600.32	601.08	600.20	603.34	601.37
5	589.38	22.30	16.22	6.50	3.36	344.15	40.84	1.42	0.74	0.54	600.59	600.97	600.55	605.19	600.00
6	600.00	64.47	35.53	15.30	6.53	346.71	27.99	3.80	1.50	1.05	600.80	601.85	601.13	602.55	600.00
7	600.00	133.04	68.24	27.08	11.01	484.48	41.03	8.29	2.76	1.79	601.33	608.03	608.14	575.70	600.00
8	600.00	219.83	115.85	44.26	17.81	590.05	32.48	15.28	4.81	2.66	601.85	600.00	601.91	525.17	600.00
9	600.00	322.62	179.07	63.20	26.07	600.00	52.28	23.98	7.61	4.07	604.70	600.00	600.82	602.70	600.00
10	600.00	335.41	305.47	92.14	39.01	600.00	77.25	36.07	11.81	5.50	600.88	600.00	600.00	601.92	600.00

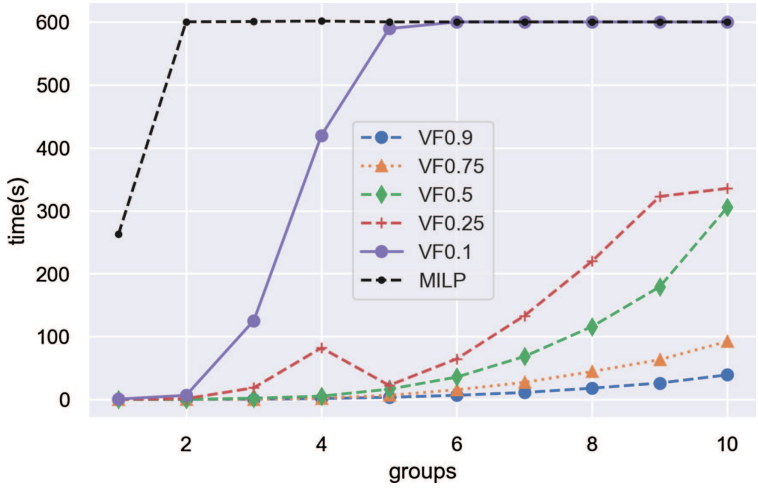

FIGURE 10. Time of variable fixing heuristics.

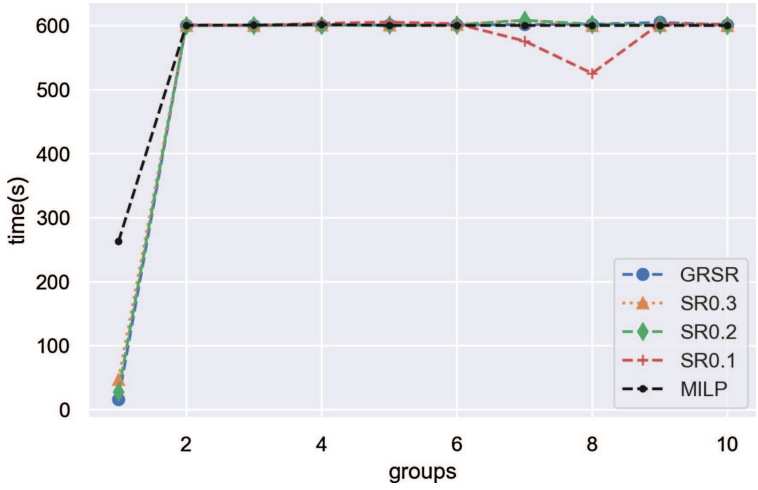

FIGURE 11. Time of greedy randomized size-reduction heuristic and size reduction heuristics.

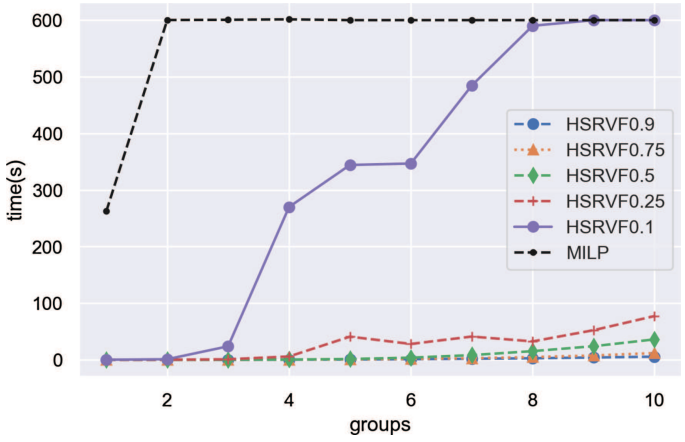

FIGURE 12. Time of hybrid size reduction and variable fixing heuristics.

FIGURE 13. Tukey confidence intervals for average LDB values. 95% of confidence level.

TABLE 4. Average LBD values for all methods in each set of instances.

Group	VF0.9	VF0.75	VF0.5	VF0.25	VF0.1	HSRVF0.1	HSRVF0.25	HSRVF0.5	HSRVF0.75	HSRVF0.9	GRSR	SR0.3	SR0.2	SR0.1	MILP	GH
1	1.885457	2.111204	2.120343	2.160473	2.208312	2.028956	2.248829	2.262046	2.302349	2.357831	2.328050	1.260374	1.300843	*	1.262925	2.490888
2	1.333834	1.546879	1.583225	1.608296	1.651263	1.337499	1.555468	1.591814	1.616885	1.659852	0.978641	0.959475	0.955169	1.007810	0.957838	2.087110
3	0.770203	0.916755	0.936642	0.946689	0.970489	0.769732	0.916755	0.936642	0.946689	0.970489	0.523241	0.699852	0.538617	0.530311	1.006878	1.301332
4	0.367737	0.412366	0.417236	0.419012	0.424944	0.328373	0.412366	0.417236	0.419012	0.424944	0.381062	0.407712	0.401922	0.364067	0.526450	0.630408
5	0.244604	0.269255	0.194222	0.196285	0.198083	0.148311	0.191557	0.194222	0.196285	0.198083	0.183663	0.206016	0.201760	0.204005		0.347643
6		0.109572	0.103940	0.104714	0.105839	0.095816	0.103590	0.103940	0.104714	0.105839	0.098765	0.114792	0.105488	0.100690		0.207017
7		0.074380	0.057200	0.057960	0.058220	0.071160	0.057060	0.057200	0.057960	0.058220	0.052420	2.499800	0.059040	0.050780		0.147200
8		0.046917	0.031150	0.031733	0.032067	0.050944	0.031150	0.031150	0.031733	0.032067	0.030783		0.058033	0.026833		0.106217
9		0.031900	0.018957	0.019400	0.019614		0.018900	0.018957	0.019400	0.019614	0.032057		9.510643	1.812700		0.078500
10		0.019219	0.011175	0.011425	0.011563		0.011163	0.011175	0.011425	0.011563	7.708000			2.262675		0.067388

Taking into consideration the computational times, after the ANOVA test we find a p-value equals to $= 8.78156E - 38$. Thus, there is a significative difference between the evaluated methods. In Figures 10–12, this difference is evidential. We can observe that the methods MILP, VF0.1, and HSRVF0.1 present higher computational times. Taking into account the LBD values, after the ANOVA test, we find a p-value equals to $= 0.961337485$. Therefore, the difference between the evaluated methods is not significant. However, we can emphasize that several methods were not able to find feasible solutions for all the test instances. Thus, the treatment of missing data could lead to a better evaluation of the methods under comparison.

Figure 13 illustrates the Tukey multiple comparisons of means with 95% family-wise confidence level, taking into account the instances with feasible integer solutions within the stipulated time limit. Based on the results obtained, we can observe that the differences among the methods under comparison are not statistically significant. Thus, our proposals have presented competitive results in comparison with the algorithms adapted from Stefanello *et al.* [26] and Martinez-Gavara *et al.* [17].

6. Conclusions

In this paper, we investigate the multi-capacitated clustering problem (MCCP). The objective function is to minimize the sum of the distances between the clients and the potential clusters. We develop a new integer linear programming model, a greedy constructive heuristic, as well as four variable-fixing matheuristics.

The idea of the proposed matheuristics is to reduce the number of integer decision variables using heuristic procedures. Two procedures are extensions of the size-reduction algorithm proposed by Stefanello *et al.* [26] and the variable-fixing heuristic proposed by Martinez-Gavara *et al.* [17]. In our two innovative proposals, we hybridize the size-reduction with the well-known GRASP metaheuristic as well as the variable-fixing heuristic proposed by Martinez-Gavara *et al.* [17].

This paper reports results for different classes of instances proposed to the MCCP found by the greedy heuristic, MILP model, and the four presented matheuristics. The solution methods have been compared on an extensive benchmark of randomly generated test instances, considering the lower bound deviation as the performance measure. In most tested problem instances, the MILP model is not able to find feasible solutions. On the other hand, size-reduction and variable-fixing heuristics present feasible solutions in the most evaluated problems. In particular, the VF025 and HSRV025 matheuristics present a good trade-off between quality and computational effort. The results also show that the proposed approaches are competitive for solving the MCCP in reasonable computational times.

As extensions of this work, we recommend the use of metaheuristics in order to improve the solutions generated by the greedy heuristic. Furthermore, the development of a machine learning approach for the improvement of size-reduction and variable-fixing heuristics [7] is another promissing research avenue.

Acknowledgements. This study was financed in part by the Coordenaço de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This study was financed in part by the National Council for Scientific and Technological Development (CNPq), through grants 303594/2018-7, 309755/2021-2, and 407151/2021-4.

Conflict of interest. The authors declare that they have no conflict of interest.

References

[1] S. Ahmadi and I.H. Osman, Greedy random adaptive memory programming search for the capacitated clustering problem. *Eur. J. Oper. Res.* **162** (2005) 30–44.

[2] J. Brimberg, N. Mladenović, R. Todosijević and D. Urošević, Solving the capacitated clustering problem with variable neighborhood search. *Ann. Oper. Res.* **272** (2019) 289–321.

[3] G.O. Chagas, L.A.N. Lorena and R.D.C. dos Santos, A hybrid heuristic for the overlapping cluster editing problem. *Appl. Soft Comput.* (2019) 105–482.

[4] A.A. Chaves and L.A.N. Lorena, Clustering search algorithm for the capacitated centered clustering problem. *Comput. Oper. Res.* **37** (2010) 552–558. Hybrid Metaheuristics.

[5] A.A. Chaves and L.A.N. Lorena, Hybrid evolutionary algorithm for the capacitated centered clustering problem. *Exp. Syst. App.* **38** (2011) 5013–5018.

[6] A.A. Chaves, J.F. Gonçalves and L.A.N. Lorena, Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. *Comput. Ind. Eng.* **124** (2018) 331–346.
[7] A.T. Dauer and B.A. Prata, Variable fixing heuristics for solving multiple depot vehicle scheduling problem with heterogeneous fleet and time windows. *Optim. Lett.* **15** (2021) 153–170.
[8] Y. Deng and J.F. Bard, A reactive grasp with path relinking for capacitated clustering. *J. Heuristics* **17** (2011) 119–152.
[9] L. Fanjul-Peyro and Rubén Ruiz, Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Comput. Oper. Res.* **38** (2011) 301–309. Project Management and Scheduling.
[10] T.A. Feo and M.G.C. Resende, Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6** (1995) 109–133.
[11] P.M. França, N.M. Sosa and V. Pureza, An adaptive tabu search algorithm for the capacitated clustering problem. *Int. Trans. Oper. Res.* **6** (1999) 665–678.
[12] M.R. Garey and D.S. Johnson, Computers and Intractability. Vol. 174. Freeman, San Francisco (1979).
[13] S. Geetha, G. Poonthalir and P.T. Vanathi, Improved k-means algorithm for capacitated clustering problem. *INFOCOMP J. Comput. Sci.* **8** (2009) 52–59.
[14] M. Gnägi and P. Baumann, A matheuristic for large-scale capacitated clustering. *Comput. Oper. Res.* **132** (2021) 105304.
[15] X. Lai and J.-K. Hao, Iterated variable neighborhood search for the capacitated clustering problem. *Eng. App. Artif. Intell.* **56** (2016) 102–120.
[16] F. Mai, M.J. Fry and J.W. Ohlmann, Model-based capacitated clustering with posterior regularization. *Eur. J. Oper. Res.* **271** (2018) 594–605.
[17] A. Martínez-Gavara, D. Landa-Silva, V. Campos and R. Martí, Randomized heuristics for the capacitated clustering problem. *Inf. Sci.* **417** (2017) 154–168.
[18] D.C. Montgomery, Design and Analysis of Experiments. John Wiley & Sons (2017).
[19] J.M. Mulvey and M.P. Beck, Solving capacitated clustering problems. *Eur. J. Oper. Res.* **18** (1984) 339–348.
[20] M. Negreiros and A. Palhano, The capacitated centred clustering problem. *Comput. Oper. Res.* **33** (2006) 1639–1663.
[21] M.J. Negreiros, N. Maculan, P.L. Batista, J.A. Rodrigues and A.W.C. Palhano, Capacitated clustering problems applied to the layout of it-teams in software factories. *Ann. Oper. Res.* (2020) 1–29.
[22] I.H. Osman and N. Christofides, Capacitated clustering problems by hybrid simulated annealing and tabu search. *Int. Trans. Oper. Res.* **1** (1994) 317–336.
[23] B.A. Prata, The multi capacitated clustering problem. Technical report, Federal University of Ceará, Brazil (2015).
[24] S. Scheuerer and R. Wendolsky, A scatter search heuristic for the capacitated clustering problem. *Eur. J. Oper. Res.* **169** (2006) 533–547.
[25] H.-M. Shieh and M.-D. May, Solving the capacitated clustering problem with genetic algorithms. *J. Chin. Inst. Ind. Eng.* **18** (2001) 1–12.
[26] F. Stefanello, O.C.B. de Araújo and F.M. Müller, Matheuristics for the capacitated p-median problem. *Int. Trans. Oper. Res.* **22** (2015) 149–167.
[27] Z. Yang, H. Chen and F. Chu, A lagrangian relaxation approach for a large scale new variant of capacitated clustering problem. *Comput. Ind. Eng.* **61** (2011) 430–435. Combinatorial Optimizatiion in Industrial Engineering.
[28] Q. Zhou, U. Benlic, Q. Wu and J.-K. Hao, Heuristic search to the capacitated clustering problem. *Eur. J. Oper. Res.* **273** (2019) 464–487.