An Iterated Greedy Heuristic for a Market Segmentation Problem with Multiple Attributes

Diana L. Huerta-Muñoz
Department of Statistics and Operations Research
Universitat Politècnica de Catalunya
Jordi Girona, Edifici C5
Campus Nord 08034, Barcelona, Spain
diana.huerta@upc.edu

Roger Z. Ríos-Mercado[1]
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León (UANL), Mexico
AP 111-F, Cd. Universitaria
San Nicolas de los Garza, NL 66455, Mexico
roger.rios@uanl.edu.mx

Rubén Ruiz
Grupo de Sistemas de Optimización Aplicada
Instituto Tecnológico de Informática,
Universitat Politècnica de València
Camino de Vera s/n, 46021 Valencia, Spain
rruiz@eio.upv.es

September 2014
Revised: November 2015, October 2016, January 2017

[1]Corresponding author

Abstract

A real-world customer segmentation problem from a beverage distribution firm is addressed. The firm wants to partition a set of customers, who share geographical and marketing attributes, into segments according to certain requirements: (a) customers allocated to the same segment must have very similar attributes: type of contract, type of store and the average difference of purchase volume; and (b) compact segments are desired. The main reason for creating a partition with these features is because the firm wants to try different product marketing strategies. In this paper, a detailed attribute formulation and an iterated greedy heuristic that iteratively destroys and reconstructs a given partition are proposed. The initial partition is obtained by using a modified k-means algorithm that involves a GRASP philosophy to get the initial configuration of centers. The heuristic includes an improvement method that employs two local search procedures. Computational results and statistical analyses show the effectiveness of the proposed approach and its individual components. The proposed metaheuristic is also observed very competitive, faster, and more robust when compared to existing methods.

Keywords: Metaheuristics; Market segmentation; Iterated greedy heuristics; GRASP; Variable neighborhood search.

1 Introduction

Market segmentation is a strategy that involves the division of a larger market into segments of customers that have common needs and applications for products and services offered in the market. These segments can be identified by a number of different features, depending on the composition of each group. One of the main reasons for creating market segments is to know more about the customers and to try different strategies in order to obtain greater customer satisfaction and increased profits.

A real-world problem from a beverage distribution firm is addressed in this paper. Given a set of customers, the firm wants to partition this set into segments. Each one must be composed of customers with the most similar type of contract, type of store, and average purchase volume. Customers of each segment must be as close to each other as possible. This last feature along with the previously mentioned attributes allow the company to apply different product marketing strategies to satisfy customer needs and to obtain more substantial profits. The first contribution of this paper is the development of a detailed attribute formulation to represent this particular clustering problem. From this model, an efficient Iterated Greedy Algorithm composed of destruction and reconstruction procedures to generate feasible solutions is proposed. An adapted k-means algorithm and an improvement method based on two local search procedures are applied to obtain a good initial solution and further improve the solution quality given by the destruction-reconstruction method, respectively. Empirical work indicates the effectiveness and robustness of the proposed heuristic.

This paper is organized as follows. In Section 2 some literature about work done on optimization problems in market segmentation is discussed. In Section 3, the problem and the associated attribute formulation are described. Then, Section 4 details the proposed algorithm based on an Iterated Greedy Algorithm to solve this problem. In Section 5 computational results are shown to demonstrate the suitability of the proposed approach. We wrap up in Sections 6 with the conclusions and directions for future research.

2 Literature Review

There has been a plethora of research on market segmentation approaches and solution methods, dating back from the seminal work of Smith (1956). An extensive review of the literature on market segmentation is given by Wedel and Kamakura (2000). They carefully review each of several approaches, along with a discussion of the supporting statistical methodology. Cooil et al. (2008) review general approaches to customer segmentation, with an emphasis on the most powerful and flexible analytical approaches and statistical models. A more recent survey can be found in Liu et al. (2012).

This section is divided into three parts. First, research on important computational issues regarding market segmentation is discussed. Then a discussion on some of the most important market segmentation techniques is provided. Finally, clustering methods are reviewed.

2.1 Market Segmentation Issues

Issues such as data measurement, the choice of computation models, the algorithm complexity, and multi-objective optimization methods have been discussed by many authors. These issues explain many commonalities and differences among market segmentation algorithms (Liu et al., 2012).

Similarity measures: Works such as Green et al. (1967) and Punj and Stewart (1983) have investigated and discussed the difficulties in determining the appropriate similarity measure in market segmentation. Kleinberg (2002) develops his impossibility theorem that describes the degree of complexity of a clustering process.

Data dimensionality: There are certain problems, such as tourist segmentation, where the data is characterized by small number of respondents and a large number of survey questions. This causes serious methodological problems that have typically been addressed by using factor-cluster analysis to reduce the dimensionality of the data; however, this kind of analysis has been shown to be unacceptable to the issue of high data dimensionality in segmentation (Dolnicar and Grün, 2008). To overcome this issue, techniques such as bi–clustering have been proposed and studied by Dolnicar et al. (2012) in a tourism segmentation context.

Computational complexity: Most market segmentation problems are difficult to solve. NP-hardness has been established for many clustering problems (Aloise et al., 2009; Brucker, 1978) Therefore, the need for heuristic approaches is well justified.

2.2 Market Segmentation Techniques

Clustering can be defined as a technique that groups entities similar in measured characteristics. Following Liu et al. (2012), this definition of clustering to refer to a number of traditional clustering methods that only optimize one within-segment homogeneity objective is used. According to this view, methods such as clusterwise regression and automatic interaction detection are not clustering methods because they do not optimize within-segment homogeneity. A fundamental task of market segmentation is to group customers based on similarities in their needs and preferences, and clustering is a common tool for such a purpose.

Descriptive market segmentation methods: Segmentation techniques can be classified into descriptive or predictive methods depending on whether the method distinguishes between independent or dependent variables. Descriptive techniques include clustering methods such as

k-means and its variations (Jain et al., 1999; Chiu et al., 2009), hierarchical clustering (Punj and Stewart, 1983), p-median clustering (Klastorin, 1985), case-based reasoning (Chen et al., 2010), and self-organizing map (Lee et al., 2002) methods. Probabilistic models are often used for clustering as they take advantage of of the statistical inference capability (Fraley and Raftery, 1998) when the density distribution assumption holds for a data set.

Predictive market segmentation methods: These methods include response modeling methods such as clusterwise regression methods (Spath, 1979), clusterwise logistic and mixture regression methods (Wedel and Kamakura, 2000). Predictive methods usually result in a better predictive model for an individual segment rather than for the population as a whole, and the within-segment homogeneity of predictors is relatively low.

Multi-objective market segmentation methods: Several multi-objective heuristics and metaheuristics have been developed in the past for market segmentation (Liu et al., 2010; Caballero et al., 2011). Other techniques to address the multi-objective nature of market segmentation are the multi-stage approach, which allows us to deal with one objective at a time iteratively (Krieger and Green, 1996; Mo et al., 2010), the transformation approach, which transforms and combines multiple objectives into a single one (DeSarbo and Grisaffe, 1998; Brusco et al., 2003), and modification of traditional descriptive clustering methods and predictive clusterwise methods (Vriens et al., 1996).

2.3 Clustering Methods

Clustering methods such as k-means are very popular due to their ease of implementation. This method partitions the data set into k subsets so that all points in a given subset are closest to the same cluster center. In a regular k-means algorithm, the cluster center is not necessarily one of the objects; however, in this work, we consider a discrete version of the k-means algorithm, that is, one where the center of each cluster explicitly corresponds to an object. Thus, in the reminder of the paper whenever we refer to the k-means algorithm we mean the discrete version. Generally, the k-means algorithm has the following important properties: (i) It is efficient in processing large data sets; (ii) it often terminates at a local optimum; (iii) the clusters have spherical shapes. Choosing the proper initial cluster centers is the key step in the classical k-means procedure. It is observed that the well-known k-means algorithm is best suited for large data sets, whereas other approaches such as artificial neural networks (ANNs), genetic algorithms (GAs), tabu search (TS), and simulated annealing (SA) have been mainly tested on small data sets (Jain et al., 1999; Xu and Tian, 2015). One of the most commonly used objective functions for clustering problems is the Mean Square Error (MSE) measure (Brusco and Stahl, 2005).

Clustering algorithms have been used in a large variety of applications such as image segmentation (Jain et al., 1995; Solberg et al., 1996), object recognition (Dorai and Jain, 1995), information

retrieval (Rasmussen, 1992), and data mining (Fayyad, 1996), among many others. A variety of clustering techniques in the literature are reviewed and discussed more recently by Xu and Tian (2015).

3 Problem Description

This work addresses a real-world problem of a beverage distribution firm. Given a set of customers, the firm wants to partition this set into segments. Each segment must be composed of customers with the most similar type of contract, type of store, and average purchase volume as possible. Another feature to consider is the compactness of the segments. The firm wishes to get a partition with these features because it needs to apply different product marketing strategies. The attribute formulation to represent the specific problem is presented first. Then, an iterated greedy algorithm that is composed of two main phases, destruction and reconstruction, and uses a variable neighborhood search to improve the solution is proposed.

3.1 Formulation of attributes

Let $V = \{1, 2, \ldots, n\}$ be a set of customers, $K = \{1, 2, \ldots, k\}$ be a set of segments, S be a set of types of products (SKU), C be a set of types of contracts, and E a set of types of stores. Note that $|S|$, $|C|$, and $|E|$ are the total number of SKU, types of contracts, and types of stores, respectively. The considered parameters are: d_{ij}, the Euclidean distance between customers i and j, $i \neq j, i < j \in V$; a matrix $A = \{a_{is}\}$, where a_{is} represents the purchased volume (number of boxes) of SKU $s \in S$ demanded by customer $i \in V$. The number of partitions k is also a given parameter.

Given a collection Π of all feasible k-partitions from V, the problem consists of finding a k-partition $X = (X_1, \ldots, X_k) \in \Pi$ so as to minimize the following objective function:

$$\min_{X \in \Pi} f(X) = \alpha_1 f_{\text{disp}}(X) + \alpha_2 f_{\text{sku}}(X) + \alpha_3 f_{\text{toc}}(X) + \alpha_4 f_{\text{tos}}(X) \tag{1}$$

where $f_{\text{disp}}(X), f_{\text{sku}}(X), f_{\text{toc}}(X)$ and $f_{\text{tos}}(X)$ are normalized values of the dissimilarity functions for the attributes of dispersion, purchase volume, type of contract, and type of store, respectively, for a given partition X. The values of $\alpha_r \in [0, 1]$, where $\sum_{r=1}^{4} \alpha_r = 1$, represent the weights of these dissimilarity functions. These dissimilarity functions are explained next.

Dispersion: To measure the dispersion of a given partition X we consider the sum of intra-cluster distances (Brusco and Stahl, 2005). This is, for each segment $q \in K$, the total sum of the distances between all pairs of customers i and j, where $i < j, i, j \in X_q$. The sum of intra-cluster distances corresponds to the sum of these totals. Small values of this sum represent less dispersed segments. This is a very common measure used in the literature. In (2) $\bar{d}_{ij}$ is

the normalized value of d_{ij} given by $\bar{d}_{ij} = d_{ij}/d_{\max}$, where $d_{\max} = \max_{i<j\in V}\{d_{ij}\}$.

$$f_{\text{disp}}(X) = \sum_{q\in K}\sum_{i<j\in X_q} \bar{d}_{ij} \tag{2}$$

Purchase Volume: The way to represent the dissimilarity between customers, with respect to this attribute, arose from a specific requirement of the firm. It is represented by the total sum of squares of the differences between average purchase volumes for every pair of customers of the same segment. For a given pair of customers i and j the dissimilarity in purchase volume is represented by:

$$q_{ij}^{\text{sku}} = \sqrt{\frac{\sum_{s\in S}\left(\frac{a_{is}}{a_i^T} - \frac{a_{js}}{a_j^T}\right)^2}{|S|}} \tag{3}$$

where a_{is} is the volume (measured in boxes) that customer i demands from product (SKU) s and $a_i^T = \sum_{s\in S} a_{is}$ is the total purchase volume for each customer $i \in X_q$, $q \in K$, for all SKU s. Consequently, for a given partition X, the total dissimilarity corresponds to the total sum of squares of the differences between these volumes for all pairs of customers of each segment.

$$f_{\text{sku}}(X) = \sum_{q\in K}\sum_{i<j\in X_q} q_{ij}^{\text{sku}} \tag{4}$$

Type of Contract and Store: For the type of contract (store) the dissimilarity between customers of a given partition X will be zero if the type of contract (store) for these customers is the same; otherwise, it will be equal to one. Formally we can express it as $h_{ij} = 0$ ($g_{ij} = 0$) if the type of contract (store) of customer i is equal to the type of contract (store) of customer j, for $i \neq j, i < j \in X_q$, $q \in K$, and $h_{ij} = 1$ ($g_{ij} = 1$) otherwise. The sum of all dissimilarities between customers of each segment $q \in X$ represents the total dissimilarity in these attributes of the given partition.

$$f_{\text{toc}}(X) = \sum_{q\in K}\sum_{i<j\in X_q} h_{ij} \tag{5}$$

$$f_{\text{tos}}(X) = \sum_{q\in K}\sum_{i<j\in X_q} g_{ij} \tag{6}$$

Under this definition, the dissimilarity function for two specific customers $i, j \in V$ is given by

$$f_{ij} = \alpha_1\bar{d}_{ij} + \alpha_2 q_{ij}^{\text{sku}} + \alpha_3 h_{ij} + \alpha_4 g_{ij}. \tag{7}$$

Although it is well known that the scalar optimization approach to multiobjective combinatorial optimization may fail to identify unsupported Pareto efficient solutions, it has to be noted that all these dimensions represent a specific objective function that follows the business practices of the firm. A posteriori multiobjective approach (for example, producing a Pareto front) is not acceptable

to the firm due to its inherent complexity with the four objectives (and the thousands of potentially non-dominated points in the objective space). A convex linear combination of the four dimensions is a much more user-friendly approach, once the most suitable values of $\alpha_1, \ldots, \alpha_4$ have been agreed upon. Furthermore, the choice of this aggregation function is justified from a practical standpoint as there are only a few weight combinations that are of interest. The purpose of this study is based on the practical cases.

Pseudocode 1 IGACS($IterIG_{\max}$)

Input:

$IterIG_{\max}$: Iteration limit;

Output:

X^{best} : Best partition found;

$X \leftarrow \text{MKM}(V, k, \beta, Iter_{\max})$;
$X \leftarrow \text{LS}(X)$;
$X^{\text{best}} \leftarrow X$;
$iter \leftarrow 0$; $\tau_{\text{IG}} \leftarrow 0.0001$; $Improvement \leftarrow 1$;
while ($Improvement \geq \tau_{IG}$ **or** $iter < IterIG_{\max}$) **do**
    $Improvement \leftarrow 0$
    $X' \leftarrow \text{DestructionPhase}(X)$;
    $X' \leftarrow \text{ReconstructionPhase}(X')$;
    $X' \leftarrow \text{LS}(X')$;
    if ($f(X') < f(X)$) **then**
        $X \leftarrow X'$;
        if ($f(X) < f(X^{\text{best}})$) **then**
            $Improvement \leftarrow f(X^{best}) - f(X)$; $X^{\text{best}} \leftarrow X$;
        end if
    else
        $gap \leftarrow \frac{f(X') - f(X^{\text{best}})}{f(X^{\text{best}})}$;
        if ($iter == 0$) **then**
            $\tau \leftarrow 0.1 \cdot gap$
        end if
        if ($gap \leq \tau$ **and** $gap \neq 0$) **then**
            $X \leftarrow X'$; $as \leftarrow as + 1$; $csa \leftarrow csa + f(X)$; $nas \leftarrow 0$;
        else
            $nas \leftarrow nas + 1$; $cna \leftarrow cna + f(X)$; $as \leftarrow 0$;
        end if
        if $nas = \lceil \gamma \cdot IterIG_{\max} \rceil$ **then**
            $\tau \leftarrow \frac{cna}{nas}$; $nas \leftarrow 0$; $cna \leftarrow 0$;
        else if $as = \lceil \gamma \cdot IterIG_{\max} \rceil$ **then**
            $\tau \leftarrow \frac{csa}{as}$; $as \leftarrow 0$; $csa \leftarrow 0$;
        end if
    end if
    $iter \leftarrow iter + 1$;
end while
return X^{best}

4 Proposed Heuristics

The proposed heuristic in this paper, called Iterated Greedy Algorithm for Customer Segmentation (IGACS), follows an Iterated Greedy (IG) algorithm framework (Ruiz and Stützle, 2007) which is a metaheuristic related to the Iterated Local Search (ILS) of Lourenço et al. (2002) that iteratively applies local search in a special way focusing on the space of solutions that are locally optimal. Instead of iterating over a local search as done in ILS, IG iterates over a greedy reconstruction heuristic.

Besides its relatively simplicity, *destroy-and-reconstruct* heuristics have been particularly useful for problems related to scheduling (Ruiz and Stützle, 2007, 2008; Fanjul-Peyro and Ruiz, 2010; Urlings et al., 2010) and other generalized set covering problems (Jacobs and Brusco, 1995; Marchiori and Steenbeek, 2000), where it is especially challenging to produce feasible solutions. After the work of Ruiz and Stützle (2007), many other authors, especially in the field of scheduling, have been applying this methodology with good results (Lozano et al., 2011; Ribas et al., 2011). Local search heuristics based on variable neighborhood search have also been applied to clustering problems such as the maximally diverse grouping problem (Lai and Hao, 2016) and clustering data from heterogeneous dissimilarities(Santi et al., 2016).

In the same way as the IG, IGACS generates a sequence of solutions by iterating over greedy constructive heuristics using two main phases: destruction and reconstruction. During the destruction phase, some elements are removed from a previously complete candidate solution (partition) initially obtained by an adapted k-means algorithm. Then, the reconstruction procedure applies a greedy constructive heuristic to reconstruct the partition. The algorithm iterates over these steps until a stopping criterion is met. To improve the given partition, a Local Search (LS) procedure (described below) is applied.

Pseudocode 1 shows the steps of the proposed IGACS. As a first step (Step 1) an initial partition using a variation of the k-means algorithm proposed in this work, which we have called *Modified k-Means* (MKM), is obtained. Then a local search (LS) based on two neighborhood structures is applied (Step 2) to improve this partition. A full description of the proposed MKM and LS methods is given in Section 4.1 and 4.2, respectively. This improved partition enters the iterative part of the IGACS.

At each iteration of the IGACS (Steps 5–32), the solution goes through four main phases. First, destruction and reconstruction phases are applied (Steps 7–8). In the former, D elements are removed at random from the incumbent solution. In the latter, each removed element is reassigned to the segment that produces the lowest increment in the dissimilarity measure. These methods are further explained in Section 4.3. Once the partition has been completely reconstructed, the LS is applied again to attempt to improve the partition (Step 9). Afterwards, an acceptance criterion (Steps 10–30) is applied in order to decide whether the new reconstructed partition replaces the

current one or not.

If the total dissimilarity of the new partition is less than the current one, the current solution is updated (Steps 10–11). Then, if the total dissimilarity of this current solution is less than the best solution found until now, the $Improvement$ value is computed and the best solution is updated too (Steps 12–14). Otherwise, the algorithm goes through Steps 17–29. When a new solution is found and it is not better than the current one, it can be accepted if its relative percentage deviation (gap) is less than or equal to a certain threshold value τ (Steps 20–21) computed in Steps 25–29 (or Steps 17–19 in the very first iteration). Here, csa/as (cna/nas) represents the average dissimilarity after as (nas) iterations (proportional to the $IterIG_{\max}$ parameter), where as (nas) is the number of consecutive iterations where worst solutions were accepted (not accepted). Similarly, csa (cna) is the cost of lower quality accepted (not accepted) solutions. The IGACS iterates until the solution improvement is less than a τ_{IG} threshold and a maximum number of iterations ($IterIG_{\max}$) is reached.

4.1 Initial Partition

To obtain the initial partition, a Modified k–Means (MKM) is proposed. Basically, in the traditional k-means algorithm (Hartigan and Wong, 1979), k elements are selected to represent the initial centers (means) of each segment, the others $n-k$ elements are assigned to its closest center, and it iteratively recalculates them based on the mean of each current group. This algorithm iterates until a stopping criterion is reached. One of the most important advantages of this approach is that it can find solutions quickly. The most notable disadvantage is that the solution strongly depends on the initial selection of centers. The classic k-means is designed for elements which can be represented with numerical data; however, some variations of this algorithm have been made to address the nominal case (Guha et al., 2000; He et al., 2005). Pseudocode 2 shows the general MKM procedure for the obtention of the initial partition.

The MKM uses the most centered (the least dissimilar) elements of each segment according to the weighted sum of dissimilarities (1). The distance between each pair of elements represents the dissimilarity with respect to the four attributes studied in this work (7).

A Greedy Randomized Adaptive Search Procedure (GRASP) (Feo and Resende, 1995) is proposed to find a better initial configuration of centers (Step 3). In a particular iteration, given a partial set of centers $\bar{K} = \{c_1, \ldots, c_q\}$, a greedy function $\phi(j)$ that measures the cost of assigning a node j as a center is computed for all unassigned nodes $j \in V \setminus \bar{K}$. Then a Restricted Candidate List (RCL) is built by taking those candidates whose greedy function evaluation falls within β % from the best possible value, in other words, the RCL is restricted by the quality parameter β. An element from the RCL is randomly chosen and added to $\bar{K}$. Once the k centers are selected (Step 3), the rest of the $n-k$ elements are assigned to its closest center (Step 7) according to (1). At the end of this

procedure, if there are no further elements to assign, centers are recalculated in such a way that new centers correspond to the most centered elements of each segment (Step 8). The best solution is updated if the new solution has been improved (Steps 9-11). The inner cycle iterates until the configuration of centers does not change. The complete process is performed a maximum number of iterations $Iter_{\max}$.

Pseudocode 2 MKM$(V, k, \beta, Iter_{\max})$

Input:

V : Set of customers;
k : Number of segments;
β : GRASP RCL quality parameter;
$Iter_{\max}$: Maximum number of iterations;

Output:

X^{best} : Best partition found;

```
1:  X^best ← ∅; iter ← 0;
2:  while (iter < Iter_max) do
3:      K̄ ← locate_centers(V, k, β);
4:      K' ← ∅;
5:      while (K' ≠ K̄) do
6:          K' ← K̄;
7:          X ← assignment(V, K');
8:          K̄ ← reallocate_centers(X);
9:          if (f(X) < f(X^best)) then
10:             X^best ← X;
11:         end if
12:         iter ← iter + 1;
13:     end while
14: end while
15: return X^best
```

The purpose of introducing a GRASP-construction mechanism is to find a good configuration of initial centers and obtain better quality solutions once the assignment step has been applied. We use this method to get the initial partition for IGACS because k-means is one of the best known algorithms for clustering problems due to its easy implementation and fast convergence to good solutions. To further improve this partition, we tried three different strategies, based on greedy heuristics for the k-dispersion problem developed by Erkut et al. (1994), within the GRASP construction phase to get this initial configuration.

- **Strategy 1**: Select the k most disperse elements (considering only the dispersion attribute). First, two nodes i and j whose d_{ij} is the maximum are chosen and added to $\bar{K}$. Then, for

the rest $n - |\bar{K}|$ elements, the minimum distance between each one of them and the elements belonging to $\bar{K}$ is calculated, i.e., the greedy function is computed as $\phi(j) = \min\{d_{ji} : i \in \bar{K}\}$. The RCL is formed by the elements with the highest value. This procedure is carried out until $|\bar{K}| = k$.

- **Strategy 2**: Select the k most dissimilar elements. It consists of the same procedure except that now we directly use the dissimilarity function (7) to establish the dissimilarity between each pair of elements. In this regard, the greedy function is computed as $\phi(j) = \min\{f_{ji} : i \in \bar{K}\}$.

- **Strategy 3**: Select the k most dissimilar elements as in Strategy 2, except that now the distance from a given node j to set $\bar{K}$ is measured by the sum, not by the minimum value, i.e., $\phi(j) = \sum_{i \in \bar{K}} f_{ji}$.

In order to increase the diversity of solutions, we introduced a GRASP philosophy in each strategy to select these k elements from a restricted list of candidates (RCL) which is formed using a quality parameter β. Once k centers are selected, the assignment procedure of the MKM is applied to obtain a partition.

4.2 Improvement Procedure

The basic idea of the proposed local search (LS) procedure is to carry out the application of two different neighborhood search schemes in a sequential manner. The LS is used to improve the partitions obtained by the MKM procedure and the reconstruction phase of the IGACS. This LS (Pseudocode 3) is composed of two simple local searches or neighborhoods performed iteratively (Steps 3 and 4). The algorithm terminates when no significant improvements (measured by ϵ) are found in $Iter_{\text{LS}}$ iterations.

The first neighborhood applied is based on insertion moves. The idea is to choose iteratively a segment q randomly from K. Then an element i from X_q is removed and reinserted into a segment r. If the merit function $\phi(i, r)$, that measures the benefit of inserting element i in segment r, is positive then the move is accepted and the corresponding segments X_q and X_r are updated. This iterative process terminates when the current segment X_q has been totally explored or as soon as an improved move is found (first-found strategy). If no improved move is found, then another segment is evaluated. The algorithm terminates when all the segments have been evaluated.

The swap neighborhood is based on swapping elements $i \in X_q$ and $j \in X_r$ from different segments ($q \neq r$) considering the merit function $\phi(i, j)$. If the benefit is positive, the move is accepted and the corresponding segments q and r are updated. The search stops when the number of segments selected at random exceeds a predefined maximum number ($P_{\max}$) of segments to evaluate, due to the largest computational effort in each iteration.

Pseudocode 3 LS(X)

Input:
X : A k-partition;
Output:
X': A k-partition;

$\epsilon \leftarrow 1.0 \times 10^{-6}$; $iter \leftarrow 0$;
while ($iter < Iter_{\text{LS}}$) **do**
    $X' \leftarrow$ Insertion(X);
    $X' \leftarrow$ Swap(X');
    if ($|f(X') - f(X)| < \epsilon$) **then**
        $iter \leftarrow iter + 1$;
    end if
    $X \leftarrow X'$;
end while
return X'

4.3 Destruction, Reconstruction, and Acceptance Criterion

The destruction phase detailed in Step 5 of Pseudocode 1 is simple: given a partition X, it removes or unassigns D elements at random from X, and outputs the remaining partial solution or partition. The reconstruction phase (Step 6 of Pseudocode 1) takes the incomplete partition X as input and reassigns the unassigned customers into the segment that produces the lowest increment on the partition dissimilarity measure. That is, for any unassigned element $i \in V \setminus X$, assign i to the segment X_{q^*} where $q^* = \arg\min_{q \in K} \left\{ \sum_{j \in X_q} f_{ij} \right\}$. Its output is a complete partition X.

5 Computational Experience

The IGACS was implemented in C++ under a 64 bits Windows 7 operating system. The experiments were carried out on a HP Workstation with Intel(R) @3.5 GHz, 16 Gb. of RAM. Since there is only one real-world instance available for this specific problem, we developed previously a descriptive analysis of the data in order to generate two sets of instances. The first set was generated using a uniform distribution for geographical coordinates on a real–world case interval $X \sim U(25.343, 25.978)$ and $Y \sim U(-100.309, -99.722)$, and pseudo-real instance information for the remaining attributes. The second set considers real–world coordinates from different store locations obtained from the Mexican Institute of Statistics and Geography database (INEGI, `http://www3.inegi.org.mx/sistemas/mapa/denue/default.aspx`). For all experiments that consider the first set, 15 instances for each size of $n = 1000, 3000, 5000$ were generated. For the second set, 15 instances (5 of each size) are created. Therefore, a total of 60 instances were considered for the computational tests. Instances can be tested with any value for the number of segments

k. Nevertheless, for this study we have considered four different values of k. Values of $k = 5, 10$ represent the number of segments which obtained the best Davies–Bouldin Index evaluation (Davies and Bouldin, 1979) and the values of $k = 20, 25$ represent the number of segments proposed by the firm in the real-world case such that $k = 5, 10, 20, 25$ are the values of k considered to evaluate the performance of the proposed algorithm.

A total of 6 different vectors for α (α_r, where $r \in \{1, 2, 3, 4\}$) are tested, namely (0.25,0.25,0.25,0.25), (0.1,0.4,0.4,0.1), (1,0,0,0), (0,1,0,0), (0,0,1,0) and (0,0,0,1). The first vector is referred to as "equal weights" where the same preference is given to all objectives. The second vector corresponds to the weights set by the company or the "commercial weights". The other four vectors give complete weight to each one of the four objectives or attributes (dispersion, purchase volume, type of contract and type of store, respectively), ignoring the others. The response variable studied in this paper is the average relative percentage deviation (ARPD) which is as follows:

$$\text{ARPD} = \frac{1}{N} \sum \frac{f(X) - f(X^{\text{best}})}{f(X^{\text{best}})} \times 100 \tag{8}$$

where $f(X)$ is the resulting objective function value found for each one of the N instances tested that exist for each size n ($N = 45$ for the first set and $N = 15$ for the second set) and $f(X^{\text{best}})$ is the best objective function value found for each one of these instances under the specified tested conditions, i.e., a given number of segments k, and a specified weight vector α.

5.1 Calibration of Parameters

Initial experiments are performed to calibrate the parameters used for each component of IGACS. These experiments are described in the following subsections.

Calibration of Strategy and β

As a first experiment, the MKM algorithm is assessed using the first set of instances. Table 1 shows the average deviation from the best solution found computed for each combination of α averaged across all instance sizes n (1000, 3000, and 5000), number of k segments (5, 10, 20, and 25), strategy applied, and the GRASP quality parameter (β). The GRASP iteration limit is set at $Iter_{\max} = 100$, except for the case when $\beta = 0.0$ as this is the greedy deterministic case corresponding to a single execution of this algorithm for which $Iter_{\max} = 1$ suffices. This experiment contains the average of 100 replicates over the 45 instances tested for each segment size (18,000 results averaged). A parametric Analysis of Variance is carried out after considering the first four α_r values, β, n, k, and the Strategy level as factors and ARPD as the response variable.

It was observed that Strategy 2 obtained better solutions for most of the cases (in comparison with Strategies 1 and 3), especially when $\beta = 0.2$ except when the maximum weight is given to

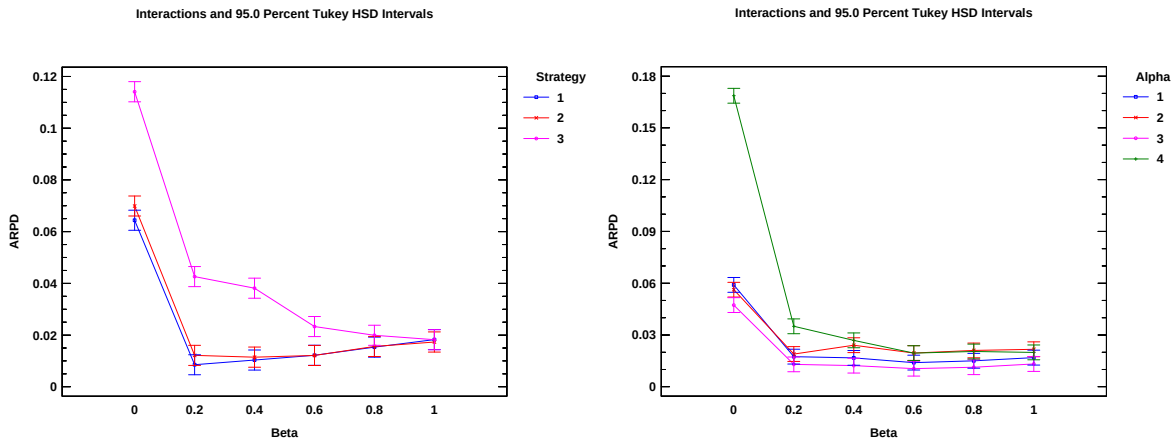
Table 1: Assessment of Strategy, and β within the MKM algorithm.

		β for the GRASP method in MKM						
α	GRASP strategy	0	0.2	0.4	0.6	0.8	1	Average
(0.25, 0.25, 0.25, 0.25)	Strategy 1	3.24	0.42	0.65	0.95	1.30	1.73	1.4
equal weights	Strategy 2	3.23	0.35	0.70	1.00	1.41	1.63	1.4
	Strategy 3	11.23	4.47	3.66	2.24	1.80	1.70	4.2
	Average	5.90	1.75	1.67	1.40	1.50	1.69	
(0.10, 0.40, 0.40, 0.10)	Strategy 1	4.27	0.37	0.73	0.97	1.59	2.15	1.68
commercial weights	Strategy 2	3.66	0.36	0.71	1.17	1.83	2.07	1.63
	Strategy 3	8.94	4.96	5.79	3.73	2.89	2.30	4.77
	Average	5.62	1.90	2.41	1.96	2.10	2.17	
(1.00, 0.00, 0.00, 0.00)	Strategy 1	2.37	0.37	0.64	0.72	1.08	1.37	1.09
dispersion	Strategy 2	2.37	0.35	0.61	0.76	1.01	1.30	1.06
	Strategy 3	9.46	3.18	2.42	1.67	1.31	1.29	3.22
	Average	4.73	1.30	1.22	1.05	1.13	1.32	
(0.00, 1.00, 0.00, 0.00)	Strategy 1	15.88	2.25	2.12	2.24	2.17	2.05	4.45
purchase volume	Strategy 2	18.71	3.81	2.57	1.92	1.99	1.93	5.16
	Strategy 3	15.99	4.45	3.38	1.68	1.97	1.99	4.91
	Average	16.86	3.50	2.69	1.95	2.04	1.99	
(0.00, 0.00, 1.00, 0.00)	Strategy 1	263143.27	15435.75	16281.16	16373.83	16015.33	13574.44	56803.96
type of contract	Strategy 2	705.63	649.84	648.03	644.57	646.69	14462.35	2959.52
	Strategy 3	2054.15	1868.05	1850.42	1827.01	1831.98	9036.91	3078.09
	Average	88634.35	5984.55	6259.87	6281.80	6164.67	12357.90	
(0.00, 0.00, 0.00, 1.00)	Strategy 1	342.47	102.72	150.45	149.75	146.79	75.24	161.24
type of store	Strategy 2	63.72	12.27	13.06	13.19	13.26	73.04	31.42
	Strategy 3	69.14	12.50	12.79	13.49	13.46	71.87	32.21
	Average	158.44	42.50	58.77	58.81	57.83	73.38	

the purchase volume attribute the Strategy 1 with $\beta = 1$ showed better results. Aberrant results were observed in the cases where the maximum weight is given to the attribute of type contract or store. As it will be seen later on, this is rapidly fixed when the initial partition is subject to local search or to the IGACS algorithm. For the remaining statistical experiments these two last cases are no longer considered. Another interesting observation that can be made is by comparing the deterministic MKM algorithm (column $\beta = 0$) with MKM when $\beta > 0$, particularly with the column $\beta = 0.2$ where the best results were observed. For instance, for the first weight combination (0.25, 0.25, 0.25, 0.25) the solution quality found indicates absolute average improvements of 2.82, 2.88, and 6.77 %, and relative average improvements of 6.73, 8.22, and 1.52 % with respect to the solutions found by the deterministic MKM under Strategies 1, 2, and 3, respectively. Similarly, for the second weight combination (0.10, 0.40, 0.40, 0.10) the solution quality found showed average

relative improvements of 10.39, 9.28, and 0.80 %, over the ones found under Strategies 1, 2, and 3, respectively. For all other weight combinations, this significant improvement of MKM over the deterministic MKM is clearly seen. Now, given that deterministic MKM corresponds to a single iteration (with β fixed at 0.0), the computational effort employed by the MKM when $\beta > 0.0$ is roughly $Iter_{\max}$ times as much as the one employed by the deterministic one. However, as can be seen later in the last experiment, even for the largest instances tested, the time spent by MKM is less than 300 seconds on average. Therefore, we conclude that the proposed GRASP version of the MKM algorithm is indeed worthwhile as it significantly improves the solution quality with respect to the deterministic MKM with a low computational effort.

A parametric Analysis of Variance is carried out after considering the first four α_r values, β, n, k, and the Strategy level as factors, and the ARPD as the response variable. The corresponding means plot of the interaction between the Strategy and β factors and β and α_r factors are shown in Figures 1a and 1b, respectively. As it can be seen in 1a, Strategies 1 and 2 outperform Strategy 3. In the interaction figures, $\beta = 0.2$ seems to provides the best overall results; however, there is no significant difference when compared to the other values (except $\beta = 0$). Means plot considers Tukey's Honest Significant Difference (HSD) at 95% confidence intervals for the interaction between Strategy and β factors. When $\alpha = (0.25, 0.25, 0.25, 0.25), (0.10, 0.40, 0.40, 0.10)$ and $(1.00, 0.00, 0.00, 0.00)$, Strategy 2 can be applied. For $\alpha = (0.00, 1.00, 0.00, 0.0)$, Strategy 1 shows a better performance than Strategy 2 and 3. On the other hand, in Figure 1b, the four weight vectors of α interact with β values. It can be seen that better results are obtained when $\beta = 0.2$ using weight vectors $\alpha = (0.25, 0.25, 0.25, 0.25), (0.1, 0.4, 0.4, 0.1)$, and $(1, 0, 0, 0)$. When $\alpha = (0, 1, 0, 0)$ there is not significant difference between values $\beta \geq 0.4$.

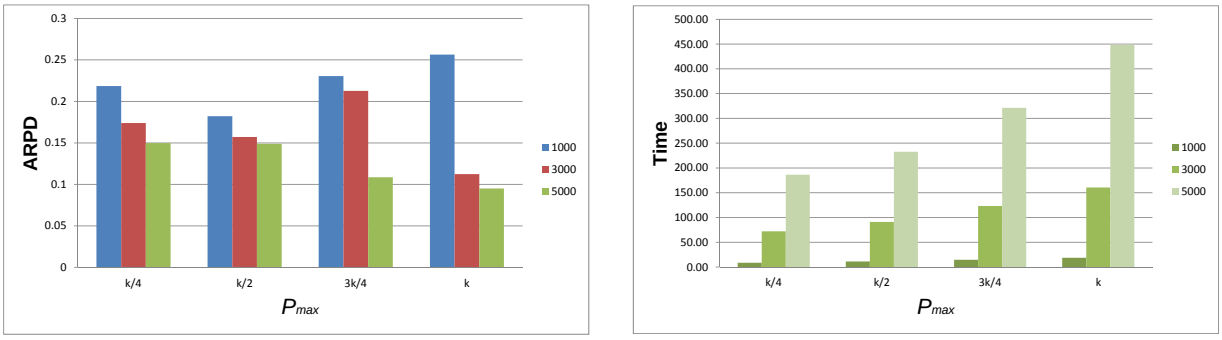


(a) (b)

Figure 1: (a) Interaction between β and Strategy, and (b) Interaction between β and α_r.

Calibration of $P_{\max}$

A second experiment is carried out to establish the maximum number of iterations ($P_{\max}$) for the Swap procedure in the LS method. This stopping criterion is based on the number of segments to evaluate. Four different values related to a specific percentage of this number of segments were tested in order to set this parameter. Instance sizes of $n = 1000, 3000, 5000$, a weight vector $\alpha = (0.25, 0.25, 0.25, 0.25)$, a $\beta = 0.2$, and the number of segments $k = 5, 10, 20, 25$ are fixed. Results in Figure 2 show that, on average, the best ARPD value is obtained when $P_{\max} = k$; however, when it is seen as the average for each instance size the $P_{\max} = \frac{k}{2}$ (which is the second best average result) show less deviation specifically for instances of size $n = 1000$. Therefore, the required computation time is less than the first case. For that reason, this parameter is fixed to $P_{\max} = \frac{k}{2}$.



(a) ARPD

(b) Computing Time

Figure 2: Assessment of parameter $P_{\max}$ in the LS procedure.

Calibration of $IterIG_{\max}$, γ, and D

Once we have fixed the $P_{\max}$, the $IterIG_{\max}$ and γ values used in the IGACS algorithm are set. For this, 15 instances of size $n = 5000$, the maximum number of segments $k = 25$, and the $Iter_{\max} = 100$, are used. We have fixed α=(0.25, 0.25, 0.25, 0.25) and used the MKM with $\beta = 0.2$ and Strategy 2 since all the attributes are considered and this combination obtained the best evaluation in the previous experiment, respectively. The stopping criteria considered for IGACS are as follows: stop once the improvement is less than 0.0001 and the maximum number of iterations is $IterIG_{\max} = 50$. The goal of this test is to reduce the $IterIG_{\max}$ in order to decrease computation time without quality loss. Results show that most of the instances finished before 30 iterations. Also, there is no significant improvement in most of them when $IterIG_{\max} > 30$. Therefore, $IterIG_{\max} = 30$ is used for the next experiments.

For the γ parameter, we have considered the first set of instances (15 per each size $n = 1000, 3000, 5000$) and the parameters fixed in previous tests: Strategy 1, $\beta = 1$ for the purchase volume attribute and Strategy 2 with $\beta = 0.2$ for the remaining values, the number of segments

$k = 5, 10, 20, 25$, the $Iter_{\max} = 100$, $P_{\max} = \frac{k}{2}$, and $IterIG_{\max} = 30$. Five different values for $\gamma = 0.10, 0.20, 0.30, 0.40, 0.50$ were tested. Each value represents the proportion of the number of iterations of IGACS to be performed before update τ which is the self-adjusting parameter used to evaluate if a worse solution should be accepted. Figure 3 shows that using $\gamma = 0.5$ the algorithm provides the best ARPD values when compared to the other values of γ.

On the other hand, the number of elements (D) to be removed from the solution (partition) in the destruction phase of the IGACS algorithm is evaluated as well. For this experiment, the 15 instances from Set 1 are used and all the parameters are fixed as mentioned before. Four different values of D based on a certain percentage of the instance size are evaluated. For example, for an instance with $n = 1000$, 3000, and 5000, a 5% value for D equals 50, 150, and 250 elements to be removed, respectively. Results show (Figure 4) that the ARPD (5,400 results averaged for each combination of α values) is better when a size of $D = 15\%$ or 20% is considered. Since there is only a slight difference in the average computation time between both options, we have selected to remove 20% of the elements for the remaining experiments.

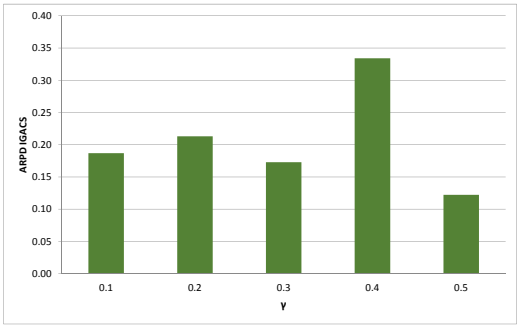


Figure 3: Assessment of parameter γ.

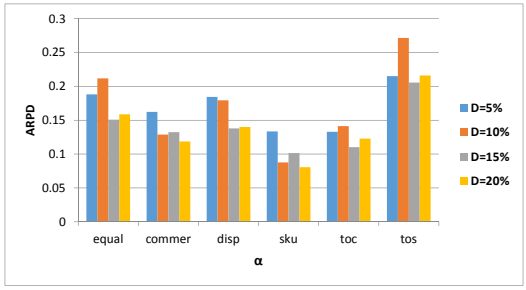


Figure 4: Assessment of parameter D.

5.2 Evaluation of IGACS

Once parameters of the proposed methods have been set, the following experiment is carried out. For each weight vector *alpha*, the proposed MKM algorithm (Strategy 1 for the purchase attribute using $\beta = 1$ and Strategy 2 for the remainder α's with $\beta = 0.20$ in all cases) was tested with a total of 100 replications per instance. The proposed LS algorithm was also assessed, i.e., the result obtained in the MKM algorithm and then a single run of the LS method. Lastly, the IGACS method is also run 30 times after the initialization with the MKM+LS methods using $D = 20\%$. Tables 2-7 show the improvement of the ARPDs between the MKM-LS and the LS-IGACS (Pseudocode 1) procedures, and also their required computational time (in seconds). The results are further detailed and grouped by n and k. Columns 3-4 describe the ARPD improvement percentage obtained by applying the LS to the MKM solutions and the improvement obtained after the iterative part of the IGACS is applied to the resulting partitions from the LS.

Table 2: Algorithm comparison, $\alpha = (0.25, 0.25, 0.25, 0.25)$.

		Improvement (%)		Time (sec)			
n	k	VNS	IGACS	MKM	LS	IGACS	Total
1000	5	0.54	0.00	6.66	0.70	9.60	16.96
	10	1.61	0.03	9.35	0.89	11.35	21.59
	20	3.63	0.08	13.02	0.99	12.11	26.12
	25	4.92	0.12	11.56	0.77	9.51	21.84
3000	5	0.28	0.00	48.46	8.37	106.05	162.87
	10	1.23	0.01	54.66	12.62	119.33	186.60
	20	2.96	0.06	52.10	13.41	125.18	190.68
	25	3.47	0.07	41.92	18.71	132.99	193.63
5000	5	0.28	0.00	136.46	27.02	330.87	494.34
	10	1.05	0.00	152.85	49.61	404.70	607.16
	20	2.30	0.03	141.23	51.91	429.01	622.15
	25	3.03	0.05	134.11	68.15	436.80	639.05
Average		2.11	0.04	66.86	21.09	177.29	265.25

Table 3: Algorithm comparison, $\alpha = (0.1, 0.4, 0.4, 0.1)$.

		Improvement (%)		Time (sec)			
n	k	VNS	IGACS	MKM	LS	IGACS	Total
1000	5	0.64	0.02	7.06	0.78	10.77	18.61
	10	1.94	0.03	8.39	0.89	11.13	20.42
	20	4.98	0.25	12.10	1.09	12.65	25.84
	25	5.67	0.21	11.10	0.77	10.00	21.87
3000	5	0.45	0.00	42.35	8.28	100.85	151.48
	10	1.57	0.01	42.91	11.32	119.42	173.64
	20	3.67	0.06	42.07	16.17	125.57	183.81
	25	4.30	0.04	41.92	18.71	132.99	193.63
5000	5	0.37	0.00	117.18	25.83	317.11	460.12
	10	1.29	0.04	109.51	11.32	119.42	240.25
	20	3.38	0.03	97.63	49.97	363.94	511.55
	25	4.02	0.05	90.78	68.49	385.59	544.86
Average		2.69	0.06	51.92	17.80	142.45	212.17

Table 4: Algorithm comparison, $\alpha = (1.0, 0.0, 0.0, 0.0)$.

		Improvement (%)		Time (sec)			
n	k	VNS	IGACS	MKM	LS	IGACS	Total
1000	5	0.51	0.00	7.09	0.61	9.66	17.36
	10	1.44	0.04	9.69	0.85	11.09	21.62
	20	3.63	0.13	13.06	0.97	11.78	25.81
	25	4.35	0.29	11.70	0.77	9.40	21.87
3000	5	0.31	0.00	55.05	8.73	106.61	170.39
	10	0.86	0.01	68.35	10.32	120.83	199.50
	20	1.94	0.01	64.32	12.12	123.07	199.51
	25	2.44	0.07	61.14	11.65	125.72	198.51
5000	5	0.20	0.00	115.24	19.37	245.84	380.45
	10	0.75	0.00	147.91	28.73	278.28	454.92
	20	1.57	0.02	141.62	33.12	282.12	456.86
	25	1.94	0.06	133.61	37.42	285.91	456.93
Average		1.66	0.05	69.07	13.72	134.19	216.98

Table 5: Algorithm comparison, $\alpha = (0.0, 1.0, 0.0, 0.0)$.

		Improvement (%)		Time (sec)			
n	k	VNS	IGACS	MKM	LS	IGACS	Total
1000	5	3.61	0.04	4.53	1.56	11.79	17.88
	10	8.23	0.07	4.21	2.27	13.87	20.35
	20	16.49	0.30	4.11	2.41	15.95	22.46
	25	19.32	0.30	3.20	1.70	12.48	17.39
3000	5	9.67	0.00	38.62	19.32	124.18	182.12
	10	7.20	0.04	30.22	37.72	172.15	240.08
	20	13.13	0.10	24.69	42.51	215.14	282.34
	25	15.13	0.09	23.06	49.10	204.62	276.79
5000	5	2.80	0.00	77.17	82.17	288.61	447.95
	10	6.59	0.05	57.17	105.35	421.16	583.68
	20	11.77	0.08	45.60	126.83	453.46	625.88
	25	14.08	0.06	43.13	120.07	473.69	636.88
Average		10.67	0.10	29.64	49.25	200.59	279.48

Table 6: Algorithm comparison, $\alpha = (0.0, 0.0, 1.0, 0.0)$.

		Improvement (%)		Time (sec)			
n	k	VNS	IGACS	MKM	LS	IGACS	Total
1000	5	100.07	0.00	8.23	0.55	8.01	16.79
	10	0.00	0.00	8.74	0.04	0.00	8.78
	20	0.00	0.00	11.42	0.04	0.00	11.46
	25	0.00	0.00	10.84	0.14	2.28	13.27
3000	5	103.67	0.00	60.64	6.02	83.42	150.08
	10	0.00	0.00	51.02	0.12	0.00	51.14
	20	0.00	0.00	56.16	0.11	0.00	56.27
	25	0.00	0.00	58.07	0.11	0.00	58.19
5000	5	106.24	0.00	122.60	13.23	179.15	314.98
	10	0.00	0.00	104.68	0.22	0.00	104.90
	20	0.00	0.00	109.46	0.20	0.00	109.66
	25	0.00	0.00	113.02	0.20	0.00	113.22
Average		25.83	0.00	59.57	1.75	22.74	84.06

Table 7: Algorithm comparison, $\alpha = (0.0, 0.0, 0.0, 1.0)$.

		Improvement (%)		Time (sec)			
n	k	VNS	IGACS	MKM	LS	IGACS	Total
1000	5	164.96	0.00	9.14	1.10	9.08	19.32
	10	422.04	0.07	9.95	0.72	6.92	17.59
	20	712.38	0.00	12.23	0.47	5.71	18.41
	25	650.03	0.00	10.84	0.14	2.28	13.27
3000	5	161.68	0.00	68.44	12.72	90.66	171.83
	10	444.19	0.00	62.68	7.64	58.92	129.23
	20	739.60	0.00	57.15	4.04	45.36	106.56
	25	613.59	0.00	58.86	1.64	21.86	82.37
5000	5	159.83	0.00	147.31	31.42	198.52	377.25
	10	452.43	0.00	131.25	17.15	123.49	271.90
	20	751.24	0.01	118.35	9.84	105.89	234.08
	25	6575.53	0.00	115.81	4.00	56.02	175.83
Average		487.29	0.01	66.83	7.57	60.39	134.80

Columns 5-7 represent the computation time required by each method to obtain. Finally, column

8 shows the total time required by the complete IGACS procedure to obtain the final solution. As can be seen, the results of the MKM algorithm improve considerably after a single pass of the VNS method is applied. This is particularly important for the last two cases where the maximum weight is given to the attributes of type of contract or store with $\alpha = (0.0, 0.0, 1.0, 0.0)$ or $\alpha = (0.0, 0.0, 0.0, 1.0)$. The additional CPU time needed increases substantially, especially for larger problems. For example, MKM needs 66.86 seconds on average for $\alpha = (0.25, 0.25, 0.25, 0.25)$ and VNS requires an additional 21.09 seconds on average. The solution quality observed after applying IGACS is slightly better then the one observed from LS. Of course, such improvements come at a computational cost, as the additional CPU time of IGACS exceeds 473 seconds in the worst case for the largest instances of 5000 customers. In any case, the total CPU time of applying IGACS (which comes after applying MKM, LS, and then IGACS) rarely exceeds 630 seconds in the worst case. Considering the large size of the real-world clustering problem dealt with in this paper (up to 5000 customers), this is highly acceptable.

Table 8: Evaluation of LS within IGACS.

		ARPD (%)				Time (sec)			
n	k	Case A	Case B	Case C	Case D	Case A	Case B	Case C	Case D
1000	5	0.00	0.06	0.71	0.22	17.42	17.82	5.15	19.12
	10	0.02	0.00	1.38	0.02	17.98	18.39	10.50	12.39
	20	0.36	0.00	3.14	0.03	21.39	21.68	11.35	17.10
	25	0.00	0.02	6.58	0.02	20.80	17.76	9.53	24.39
3000	5	0.13	0.00	0.80	0.80	163.47	165.18	53.31	53.31
	10	0.06	0.00	0.40	0.40	160.79	163.37	52.75	52.75
	20	0.22	0.00	1.24	1.24	167.89	169.86	51.94	51.94
	25	0.18	0.00	2.86	2.86	163.70	167.32	52.32	52.32
5000	5	0.00	0.12	0.99	0.99	406.65	412.52	127.76	127.76
	10	0.00	0.00	0.33	0.33	416.00	422.90	124.59	124.59
	20	0.00	0.06	0.44	0.44	415.27	426.70	118.15	118.15
	25	0.18	0.00	1.44	1.44	417.90	427.80	115.28	115.28
Average		0.10	0.02	1.69	0.73	202.61	199.10	61.05	64.09

The LS method applied after the MKM method improved enough the final solution and the improvement obtained for the destruction and reconstruction procedures is not as great in comparison with the one obtained from the LS at the beginning of IGACS. For that reason, the following test was made in order to analyze clearly the performance of the IGACS method considering or not the LS procedure inside of it. Four different cases were evaluated:

- Case A: The proposed IGACS.
- Case B: Remove LS after MKM and before the iterative part of IGACS (the LS procedure still remains to improve the reconstructed solutions).
- Case C: IGACS without LS (in any part of the algorithm).
- Case D: Apply LS to improve the final solution of IGACS after all reconstructions.

Table 8 shows the effectiveness of the LS method within IGACS. The best results were obtained when LS is removed from the beginning of the IGACS and it is only used to improve the reconstructed partition. The worst (highest) ARPD was obtained when LS is removed completely from the proposed method. Also the IGACS required more time when LS was not applied to improve the MKM solution, but at the end it showed the best results for all cases. Computation time decreases significatively when LS is applied at the end of IGACS; however, it was not able to find better solutions than the ones obtained when LS is applied after reconstructing each solution.

5.3 Comparison with Existing Methods

A final question remains about the comparison between the proposed approaches and the segmentation methods used by the firm. While a quantitative analysis cannot be provided, it can be clearly stated that the proposed approach is vastly superior. This is based on the fact that the methods employed by the firm were basically manual and only guided by a Geographical Information System (GIS). This manual process required hours of intensive work and the results were far from optimal. Furthermore, the final result did not consider all the attributes in a meaningful way. With our presented approaches the result is obtained much faster and the partitions have better values in all tested attributes. Furthermore, no software licenses have to be paid. Nevertheless, as a good practice, the performance of IGACS should be compared with other clustering approaches.

With this in mind, the purpose of this section is to present a comparison of the proposed metaheuristic with some of the existing methods in literature. The Waikato Environment for Knowledge Analysis (WEKA) is an open source software written in Java which contains a collection of machine learning techniques developed by a research group from the University of Waikato. The k-means algorithm is one of the methods of this collection. More information about this tool and its methods can be found in Witten et al. (2011). A second approach used for the comparison is the hybrid data mining metaheuristic (DM-G) proposed by Plastino et al. (2011) for the p-median problem. The C++ code of this metaheuristic was provided by the authors and tests were performed considering the same server features as IGACS but under Ubuntu 14.04 Operating System to avoid possible inconsistencies as it was developed for a Linux kernel. A third method, a GRASP metaheuristic with path-relinking (PR-G), proposed by Frinhani et al. (2011), is considered in this

evaluation. The Java code was provided by the authors and tests were performed under the Windows 7 Operating System.

For this experiment, the second set of instances, the combination of weights $\alpha = (0.25, 0.25, 0.25, 0.25)$ and the number of segments $k = 5, 10, 20, 25$ were taken into account. Parameters of IGACS are fixed as proposed in Section 5. All source codes were adapted in order to match the objective function of our studied problem. These five approaches are compared: the k-means algorithm from WEKA, the DM-G, the PR-G, the proposed MKM (with $\beta = 0.20$), and the IGACS approach which takes the MKM solution as initial solution to be improved. In a first test we set the stopping condition for both DM-G and IGACS to 30 iterations, while for WEKA and MKM we use 100 iterations. On the other hand, PR-G was stopped after 1 elapsed iteration due to its high time requirement. These results are shown in Table 9.

Table 9: Comparison of MKM and IGACS with existing methods (WEKA, DM-G, and PR-G).

		ARPD (%)					Time (sec)				
n	k	WEKA	DM-G	PR-G	MKM	IGACS	WEKA	DM-G	PR-G	MKM	IGACS
1000	5	17.83	6.20	0.00	8.19	0.06	5.39	30.89	25.22	2.07	23.02
	10	58.75	11.71	0.96	29.55	0.00	4.63	30.46	38.81	2.37	23.45
	20	71.38	16.24	0.00	38.41	0.26	11.52	30.34	58.70	3.78	30.30
	25	60.06	14.68	0.04	34.70	0.00	28.27	30.26	69.23	4.51	33.78
3000	5	22.06	6.69	0.00	7.91	0.11	55.87	313.68	1886.05	13.03	248.26
	10	53.22	11.93	0.00	31.02	0.46	56.16	310.46	1177.97	11.29	250.84
	20	56.46	17.94	0.00	36.02	0.27	59.87	309.48	1696.32	12.65	295.01
	25	67.37	16.40	0.00	35.55	0.66	67.51	309.42	1781.26	12.73	307.80
5000	5	22.40	6.82	0.00	8.38	0.59	178.26	744.57	6809.83	40.91	731.17
	10	53.55	11.55	0.00	32.94	0.18	171.42	737.20	7648.47	28.84	611.12
	20	56.80	18.17	0.00	32.23	0.06	200.02	735.29	10279.05	22.50	614.25
	25	54.20	14.58	0.76	29.56	0.00	157.21	734.87	13338.75	22.33	693.02

In each cell, the ARPD of five instances, for a given number of clusters k, is shown. It is remarkable than the MKM outperforms WEKA. However, although DM-G obtains better results than the MKM, it does not outperform IGACS. PR-G obtained, in most of the cases, better results than IGACS but the results of the latter are not so far from the best ones reported by PR-G. In fact, IGACS can obtain very similar results in much less time. PR-G needs a large amount of time to perform only one iteration.

In a second experiment, we set now the stopping condition for DM-G and IGACS to the time required by PR-G after one elapsed iteration per each n and each k. This is given in Table 10. Note that columns referring to WEKA and MKM were omitted in the remaining tables since their results

do not change significantly with respect to Table 9. We will only focus on the three approaches that have yielded the best results. In Table 10 we can observe that, as the time increased, it was possible for IGACS to improve the best solution of the PR-G approach in a few cases. The values in parentheses indicate the number of best solutions encountered during experimentation.

Table 10: Comparison among DM-G, PR-G, and IGACS approaches. Stopping criterion for DM-G and IGACS set to that of 1 iteration of PR-G according to each combination of n and k.

		ARPD (%)			Time (sec)		
n	k	DM-G	PR-G	IGACS	DM-G	PR-G	IGACS
1000	5	6.32(0)	0.00(5)	0.48(2)	25.79	25.22	25.43
	10	10.35(0)	0.00(5)	0.06(3)	39.26	38.81	38.73
	20	16.16(0)	0.00(5)	0.22(3)	59.06	58.70	58.81
	25	15.17(0)	0.89(0)	0.00(5)	69.58	69.23	68.90
3000	5	6.37(0)	0.01(1)	0.00(5)	1922.99	1886.05	1887.60
	10	11.44(0)	0.21(1)	0.03(4)	1186.76	1177.97	1139.39
	20	16.89(0)	0.00(5)	1.30(1)	1702.45	1696.32	1702.54
	25	15.97(0)	0.68(0)	0.00(5)	1786.41	1781.26	1780.16
5000	5	6.55(0)	0.00(5)	0.41(2)	6926.77	6809.83	6823.89
	10	9.20(0)	0.00(5)	0.00(5)	7698.9	7648.47	7668.59
	20	16.39(0)	0.00(5)	0.57(2)	10309.62	10279.05	10304.86
	25	12.94(0)	0.36(2)	0.00(5)	13372.68	13338.75	13370.41

In the following experiment, we set the stopping condition for DM-G, PR-G and IGACS to the maximum time required by the PR-G to perform one iteration per each size n. That is, for $n =$ 1000, 3000, and 5000, time limit is set to 70, 1890, and 13340 sec, respectively. Results are displayed in Table 11.

PR-G needs a big amount of time to perform only one iteration and, as aforementioned, we used this time of a single iteration as a stopping time for both DM-G and IGACS. Basically, PR-G cannot be stopped before this CPU time, which is a serious limitation of the method. Furthermore, it seems that IGACS cannot reach the very best results in all cases when compared to PR-G. Still, the difference in solution quality between both methods is largely minor and the speed advantage of IGACS more than compensates this potential shortcoming.

In a last experiment, Table 12 shows another advantage of IGACS, as it can be used in collaboration with PR-G to further improve the solution obtained by PR-G. First, the PR-G was run and, once the solution after a single iteration is obtained, it was used as an input to IGACS. The stopping criterion for IGACS was half of the time required by PR-G.

Table 11: Comparison among DM-G, PR-G, and IGACS approaches. Stopping time for DM-G, PR-G and IGACS set to the maximum time required by the PR-G to perform one iteration among all k for each fixed n.

		ARPD (%)			Time (sec)		
n	k	DM-G	PR-G	IGACS	DM-G	PR-G	IGACS
1000	5	5.87(0)	0.00(5)	1.03(1)	71.61	73.24	70.02
	10	10.97(0)	0.69(3)	0.00(5)	70.77	75.32	70.01
	20	16.16(0)	0.00(5)	0.46(1)	70.37	84.87	70.06
	25	13.94()	0.00(5)	0.15(1)	70.36	79.62	70.41
3000	5	6.07(0)	0.05(4)	0.00(5)	1922.99	2226.36	1895.64
	10	10.28(0)	0.00(5)	0.18(4)	1903.27	2431.10	1890.91
	20	16.61(0)	0.00(5)	0.47(1)	1896.21	2387.95	1891.83
	25	15.39(0)	0.20(1)	0.00(5)	1895.11	2626.98	1896.16
5000	5	6.54(0)	0.00(5)	0.64(0)	13566.36	14658.45	13343.40
	10	8.90(0)	0.02(3)	0.00(5)	13427.80	14544.31	13367.26
	20	16.66(0)	0.00(5)	0.23(2)	13379.88	16365.22	13361.42
	25	13.01(0)	0.00(5)	0.07(2)	13372.68	16757.67	13355.31

Table 12: Comparison between PR-G and IGACS taking as input the PR-G solution.

		ARPD (%)		Time (sec)		
n	k	PR-G	IGACS	PR-G	IGACS	Total
1000	5	0.00	0.00	32.46	15.51	47.97
	10	0.01	0.00	47.51	23.51	71.03
	20	0.07	0.00	70.35	34.61	104.96
	25	0.40	0.00	86.44	41.04	127.47
3000	5	0.00	0.00	2659.93	1335.43	3995.35
	10	0.00	0.00	2025.81	1012.34	3038.16
	20	0.05	0.00	3130.50	1570.14	4700.65
	25	0.02	0.00	2905.46	1449.85	4355.31
5000	5	0.00	0.00	12123.77	6072.68	18196.44
	10	0.00	0.00	12947.98	6492.73	19440.71
	20	0.01	0.00	15158.60	7574.55	22733.15
	25	0.01	0.00	23395.06	11699.91	35094.96

It was observed that IGACS was able to further improve the solution of the PR-G in almost all

runs. Given all these characteristics that IGACS presents with respect to PR-G, it can be concluded that IGACS obtains solutions as almost as good as those obtained by PR-G but in much less time. This makes it more robust and efficient than PR-G especially for large size instances which are common in real-world applications.

6 Conclusions

In this study a formulation to represent a market segmentation problem that considers multiple attributes, arising from a real-world application in a beverage distribution firm, is presented. An iterated greedy algorithm (IGACS) to solve the problem efficiently was proposed. In addition, an adaptation of the well-known k-means algorithm was developed to create partitions to our problem. The modifications include three different GRASP-based strategies to select the initial configuration of centers in an attempt to obtain better solutions. To improve the solution, a local search procedure was developed and incorporated after the MKM and the reconstruction phase of the IGACS.

Comprehensive computational and statistical experiments have been performed for fine tuning the algorithmic parameters of the IGACS. Some statistical tests were made to guarantee that the observed differences in the average results were indeed statistically significant. Empirical results show that applying the LS procedure after the MKM algorithm improve the dissimilarity of the partition significantly. Moreover, IGACS can improve this dissimilarity even more. At the end of IGACS, excellent results were obtained in a reasonable amount of time. Furthermore, the proposed approach was compared with the k-means algorithm from WEKA and two metaheuristics proposed in the literature. Results showed that IGACS outperformed the tested algorithms in terms of efficiency and robustness.

About future research directions, although IGACS has shown an efficient performance during the tests, it has room for improvement. One of the opportunity areas is still the selection of the initial cluster centers. We have used the well-known k-means algorithm because of its simplicity and its rapid convergence, and we tried to improve some of its main drawbacks such as the selection of the initial centers creating a GRASP procedure and the problems caused by the presence of outliers considering the most centered (similar) customers of each segment as cluster centers instead of means. A new improvement to consider is the one proposed by Kanungo et al. (2002); Likas et al. (2003); Žalik (2008). They use a k-d tree structure in which some information is stored during the optimization process in order to avoid recalculating it in future iterations and make the algorithm faster. Another improvement can be the use of a new alternative of the initialization method; for example, Celebi et al. (2013) studied a computational efficiency comparison among eight known initialization methods for the k-means algorithm and give some recommendations of which algorithms are better suitable for certain situations. Another improvement can be to eliminate the dependency on the number of clusters in the MKM as it is proposed in Cheung (2003). Other works involving more sophisticated procedures can be found in Liu and Li (2014).

Another important issue about scalarization is how each of the individual objective functions in the scalar objective (1) is normalized. As seen in Section 3, we normalized each individual objective function by dividing by a quantity that guarantees each measure is in the 0-1 scale. A more robust approach would be to normalize each function by considering the single-objective optimal value (or best known value) as normalization factor. In this particular problem, the inherent difficulty of the problem rules out the possibility of computing single-objective optimal objective function values. Furthermore, no good bounds about the quality of individual best known values are known either. Therefore, deriving either exact values or good quality bounds for the single-objective optimal values become an interesting area for further research.

Acknowledgements: This research has been supported by the Mexican National Council for Science and Technology (CONACYT) through grants CB2005-01-48499Y and CB2011-01-166397, and a scholarship for graduate studies, and by the Universidad Autónoma de Nuevo León through its Scientific and Technological Research Support Program (PAICYT), grants CA1478–07, CE012–09, IT511–10, and CE331–15. Rubén Ruiz is partially supported by the Spanish Ministry of Economy and Competitiveness, under the project "SCHEYARD – Optimization of Scheduling Problems in Container Yards" (No. DPI2015-65895-R) financed by FEDER funds. We would like to thank Rafael Frinhani, Richard Fuchshuber, and their corresponding research teams for providing us the source code of their algorithms to carry out the corresponding tests. Furthermore, we are grateful to the editor and the four anonymous reviewers for their careful reading of our manuscript and their constructive comments and suggestions which helped us improve its quality.

References

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248.

Brucker, P. (1978). On the complexity of clustering problems. In Henn, R., Korte, B., and Oettli, W., editors, *Optimization and Operations Research*, volume 157 of *Lecture Notes in Economics and Mathematical Systems*, pages 45–54. Springer, Berlin, Germany.

Brusco, M. J., Cradit, J. D., and Tashchian, A. (2003). Multicriterion clusterwise regression for joint segmentation settings: An application to customer value. *Journal of Marketing Research*, 40(2):225–234.

Brusco, M. J. and Stahl, S. (2005). *Branch-and-Bound Applications in Combinatorial Analysis*. Springer, New York.

Caballero, R., Laguna, M., Martí, R., and Molina, J. (2011). Scatter tabu search for multiobjective clustering problems. *Journal of the Operational Research Society*, 62(1):2034–2046.

Celebi, M. E., Kingravi, H. A., and Vela, P. A. (2013). A comparative study of efficient initialization

methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210.

Chen, Y.-K., Wang, C.-Y., and Feng, Y.-Y. (2010). Application of a 3NN+1 based CBR system to segmentation of the notebook computers market. *Expert Systems with Applications*, 37(1):276–281.

Cheung, Y.-M. (2003). k*-means: A new generalized k-means clustering algorithm. *Pattern Recognition Letters*, 24(15):2883–2893.

Chiu, C.-Y., Chen, Y.-F., Kuo, I.-T., and Chun Ku, H. (2009). An intelligent market segmentation system using k-means and particle swarm optimization. *Expert Systems with Applications*, 36(3):4558–4565.

Cooil, B., Aksoy, L., and Keiningham, T. L. (2008). Approaches to customer segmentation. *Journal of Relationship Marketing*, 6(3–4):9–39.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227.

DeSarbo, W. S. and Grisaffe, D. (1998). Combinatorial optimization approaches to constrained market segmentation: An application to industrial market segmentation. *Marketing Letters*, 9(2):115–134.

Dolnicar, S. and Grün, B. (2008). Challenging factor cluster segmentation. *Journal of Travel Research*, 47(1):63–71.

Dolnicar, S., Kaiser, S., Lazarevski, K., and Leisch, F. (2012). Biclustering: Overcoming data dimensionality problems in market segmentation. *Journal of Travel Research*, 51(1):41–49.

Dorai, C. and Jain, A. K. (1995). Shape spectra based view grouping for free-form objects. In *Proceedings of the International Conference on Image Processing (ICIP-95)*, pages 249–243, Washington DC. IEEE Computer Society.

Erkut, E., Ürküsal, Y., and Yenycerioğlu, O. (1994). A comparison of p-dispersion heuristics. *Computers & Operations Research*, 21(10):1103–1113.

Fanjul-Peyro, L. and Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207(1):55–69.

Fayyad, U. M. (1996). Data mining and knowledge discovery: Making sense out of data. *IEEE Expert*, 11(5):20–25.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

Fraley, C. and Rafferty, A. E. (1998). How many clusters? which clustering methods? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588.

Frinhani, R. M. D., Silva, R. M. A., Mateus, G. R., Festa, P., and Resende, M. G. C. (2011). GRASP with path-relinking for data clustering: A case study for biological data. In Pardalos, P. M. and

Rebennack, S., editors, *Experimental Algorithms: Proceedings of the 10th International Symposium (SEA 2011)*, volume 6630 of *Lecture Notes in Computer Science*, pages 410–420, Berlin, Germany. Springer-Verlag.

Green, P. E., Frank, R. E., and Robinson, P. J. (1967). Cluster analysis in test market selection. *Management Science*, 13(8):B387–B400.

Guha, S., Rastogi, R., and Shim, K. (2000). ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366.

Hartigan, J. A. and Wong, M. A. (1979). A K-means clustering algorithm. *Journal of the Royal Statistical Society, Series C: Applied Statistics*, 28(1):100–108.

He, Z., Deng, S., and Xu, X. (2005). Improving k-modes algorithm considering frecuencies of attribute values in mode. In Hao, Y., Liu, J., Wang, Y., Cheung, Y., Yin, H., Jiao, L., Ma, J., and Jiao, Y.-C., editors, *Computational Intelligence and Security*, volume 3801 of *Lecture Notes in Computer Science*, pages 157–162. Springer, Berlin, Germany.

Jacobs, L. W. and Brusco, M. J. (1995). A local search heuristic for large set-covering problems. *Naval Research Logistics Quarterly*, 42(7):1129–1140.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

Jain, R., Kasturi, R., and Schunk, B. G. (1995). *Machine Vision*. McGraw-Hill Series in Computer Science. McGraw-Hill, New York.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892.

Klastorin, T. D. (1985). The p-median problem for cluster analysis: A comparative test using the mixture model approach. *Management Science*, 31(1):84–95.

Kleinberg, J. (2002). An impossibility theorem for clustering. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 463–470, Vancouver, Canada. MIT Press.

Krieger, A. M. and Green, P. E. (1996). Modifying cluster-based segments to enhance agreement with an exogenous response variable. *Journal of Marketing Research*, 33(3):351–363.

Lai, X. and Hao, J.-K. (2016). Iterated maxima search for the maximally diverse grouping problem. *European Journal of Operational Research*, 254(3):780–800.

Lee, S. C., Suh, Y. H., Kim, J. K., and Lee, K. J. (2002). A cross-national market segmentation of online game industry using SOM. *Expert Systems with Applications*, 27(4):559–570.

Likas, A., Vlassis, N., and Verbeek, J. (2003). The global k-means clustering algorithm. *Pattern*

Recognition Letters, 36(2):451–461.

Liu, X. and Li, M. (2014). Integrated constraint based clustering algorithm for high dimensional data. *Neurocomputing*, 142:478–485.

Liu, Y., Kiang, M., and Brusco, M. (2012). A unified framework for market segmentation and its applications. *Expert Systems with Applications*, 39(11):10292–10302.

Liu, Y., Ram, S., Lusch, R. F., and Brusco, M. (2010). Multicriterion market segmentation: A new model, implementation, and evaluation. *Marketing Science*, 29(5):880–894.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2002). Iterated local search. In Glover, F. and Kochenberger, G. A., editors, *Handbook of Metaheuristics*, International Series in Operations Research and Management Science, chapter 11, pages 321–353. Kluwer, Norwell.

Lozano, M., Molina, D., and García-Martínez, C. (2011). Iterated greedy for the maximum diversity problem. *European Journal of Operational Research*, 214(1):31–38.

Marchiori, E. and Steenbeek, A. (2000). An evolutionary algorithm for large set covering problems with applications to airline crew scheduling. In Cagnoni, S., Poli, R., Li, Y., Smith, G., Corne, D., Oates, M. J., Hart, E., Lanzi, P. L., Boers, E. J. W., Paechter, B., and Fogarty, T. C., editors, *Real-World Applications of Evolutionary Computing*, volume 1803 of *Lecture Notes in Computer Science*, pages 367–381. Springer, Berlin, Germany.

Mo, J., Kiang, M. Y., Zou, P., and Li, Y. (2010). A two-stage clustering approach for multi-region segmentation. *Expert Systems with Applications*, 37(10):7120–7131.

Plastino, A., Fuchshuber, R., Martins, S. d. L., Freitas, A. A., and Salhi, S. (2011). A hybrid data mining metaheuristic for the p-median problem. *Statistical Analysis and Data Mining*, 4(3):313–335.

Punj, G. and Stewart, D. W. (1983). Cluster analysis in marketing research: Review and suggestion for application. *Journal of Marketing Research*, 20(2):118–134.

Rasmussen, E. (1992). Clustering algorithms. In Frakes, W. B. and Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice-Hall, Upper Saddle River.

Ribas, I., Companys, R., and Tort-Martorell, X. (2011). An iterated greedy algorithm for the flowshop scheduling problem with blocking. *Omega*, 39(3):293–301.

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Ruiz, R. and Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187(3):1143–1159.

Santi, É., Aloise, D., and Blanchard, S. J. (2016). A model for clustering data from heterogeneous dissimilarities. *European Journal of Operational Research*, 253(3):659–672.

Smith, W. R. (1956). Product differentiation and market segmentation as alternative marketing strategies. *The Journal of Marketing*, 21(1):3–8.

Solberg, A., Taxt, T., and Jain, A. (1996). A Markov random field model for classification of multisource satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 34(1):100–113.

Spath, H. (1979). Clusterwise linear regression. *Computing*, 22(4):367–373.

Urlings, T., Ruiz, R., and Stützle, T. (2010). Shifting representation search for hybrid flexible flowline problems. *European Journal of Operational Research*, 207(2):1086–1095.

Žalik, K. (2008). An efficient k'-means clustering algorithm. *Pattern Recognition Letters*, 29(9):1385–1391.

Vriens, M., Wedel, M., and Wilms, T. (1996). Metric conjoint segmentation methods: A Monte Carlo comparison. *Journal of Marketing Research*, 33(1):73–85.

Wedel, M. and Kamakura, W. A. (2000). *Market Segmentation: Conceptual and Methodological Foundations*, volume 8 of *International Series in Quantitative Marketing*. Springer, New York, 2nd edition.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*, chapter 10, pages 403–406. Morgan Kaufmann, Burlington, 3rd edition.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193.