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

Diana L. Huerta-Muñoz
Prolec GE
Apodaca, Mexico
diana.huertamunoz@ge.com

Roger Z. Ríos-Mercado[1]
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León, Mexico
AP 111-F, Cd. Universitaria
San Nicolas de los Garza, NL 66450, 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

April 2012

[1]Corresponding author

Abstract

This paper addresses a real-world customer segmentation problem from a beverage distribution firm. 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 work, we propose a detailed attribute formulation and an iterated greedy heuristic that iteratively destroys and reconstructs a given partition. The initial partition is obtained by using a modified k-means algorithm that involves a GRASP philosophy to get the initial configuration of centroids. The heuristic includes an improvement method that employs a variable neighborhood search procedure. Computational results and statistical analyses show the effectiveness of the proposed approach.

Keywords: Market segmentation; Metaheuristics; GRASP; Iterated greedy heuristics; 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 or subgroups of customers can be identified by a number of different features, depending on the purpose of each group. One of the main reasons for creating market segments is knowing more about the customers and trying different strategies to obtain profits through customer satisfaction.

This work addresses a real-world problem from a beverage distribution firm. Having 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. Another feature to consider is the compactness of the segments, that means, customers of each segment must be as close to each other as possible. This last feature along with the previously mentioned attributes allows the company to apply different product marketing strategies to satisfy customer needs and consequently to obtain greater benefits. The first contribution of this work is the development of a detailed attribute formulation to represent this particular clustering problem. From this model, an efficient Iterated Greedy Algorithm composed by destruction and reconstruction procedures, is implemented to generate feasible solutions. We applied Variable Neighborhood Search (VNS) to further improve the solution quality given by the destruction-reconstruction method. Empirical work indicates the effectiveness of the proposed heuristic.

This paper is organized as follows. In Section 2 we discuss some literature about work done on optimization problems in market segmentation. In Section 3, we describe the problem and the associated attribute formulation. Then, Section 4 details the proposed algorithm based on an Iterated Greedy Algorithm, which we have called IGACS (Iterated Greedy Algorithm for Customer Segmentation), implemented to solve this problem. In Section 5 computational results are shown, along with statistical analyses which demonstrate the suitability of the proposed approach. Finally, in Section 6, we draw some conclusions based on the results of this work.

2 Literature Review

Market segmentation has been widely studied since the seminal work of Smith (1956). An extensive review of the literature on market segmentation is given by Wedel and Kamakura (1998). 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 emphasis on the most powerful and flexible analytical approaches and statistical models.

Clustering algorithms are among the most popular methods used to solve market segmentation problems. According to Jain et al. (1999), clustering approaches are mainly split into two main

groups: (i) hierarchical and (ii) partitional approaches. Hierarchical methods produce a nested series of partitions, while partitional methods produce only one. A partitional clustering algorithm obtains a single partition of the data instead of a clustering structure, such as the dendrogram produced by a hierarchical technique. Partitional methods have advantages in applications involving large data sets for which the reconstruction of a dendrogram is computationally prohibitive. The partitional techniques usually produce clusters by optimizing a criterion function defined either locally (on a subset of the patterns) or globally (defined over all of the patterns). A combinatorial search of the set of possible labelings for an optimum value of a criterion is clearly computationally prohibitive. In practice, therefore, the algorithm is typically run multiple times with different starting states, and the best configuration obtained from all of the runs is used as the output clustering.

The most well-known centroid algorithm is the k-means (Jain et al., 1999). The k-means method partitions the data set into k subsets such that all points in a given subset are closest to the same centre. In detail, it randomly selects k of the objects or seeds to represent the cluster centers. Based on the selected attributes, all remaining instances are assigned to their closer center. The k-means then computes the new centers by taking the mean of all data points belonging to the same cluster. The operation is iterated until there is no change in the gravity centers. If k cannot be known ahead of time, various values of k can be evaluated until the most suitable one is found. The effectiveness of this method as well as of others relies heavily on the objective function used in measuring the distance between instances. The difficulty relies on finding a distance measure that works well with all types of data. There are several approaches to define the distance between instances. 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. The algorithm described above is classified as a batch method because it requires that all the data should be available in advance. However, there are variants of the k-means clustering process, which get around this limitation. Choosing the proper initial centroids is the key step in the basic k-means procedure.

Several partitioning algorithms developed in the past are discussed by Jain et al. (1999). They present a discussion about the comparison of techniques such as k-means, artificial neural networks (ANNs), genetic algorithms (GAs), tabu search (TS), and simulated annealing (SA). As a general conclusion, it is observed that the well-known k-means algorithm is best suited for large data sets, whereas the other approaches have been mainly tested on small data sets. This is because obtaining suitable learning/control parameters for ANNs, GAs, TS, and SA is difficult and their execution times are very long for large data sets. However, it has been shown by Selim and Ismail (1984) that the k-means method converges to a locally optimal solution. This behavior is linked with the initial seed selection in the k-means algorithm. Thus if a good initial partition can be obtained quickly using any of the other techniques, then k-means would work well even on problems with large data sets. One of the most commonly used objective functions for clustering problems is the mean square

error (MSE) measure.

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), data mining (Fayyad, 1996), and market segmentation, among many others. The most important clustering techniques in the literature are reviewed and discussed in Jain et al. (1999), and more recently in Berkhin (2006).

As of late, some authors have been improving the behavior of the k-means algorithm by developing some metaheuristic techniques. For instance, Chiu et al. (2009) integrate k-means and particle swarm optimization (PSO) into a decision support system for market segmentation. Merz (2003) proposes an iterated local search (ILS) heuristic that is capable of finding near optimal solutions when applied to gene expression data resulting from biological microarray experiments. Cano et al. (2002) present a greedy randomized adaptive search procedure (GRASP) for cluster analysis, with the objective of overcoming the convergence to a local solution. The presented method uses a probabilistic greedy Kaufman initialization for getting initial solutions and the k-means algorithm as a local search algorithm. The authors compare the presented approach with some typical initialization methods: Random, Forgy, Macqueen and Kaufman. The new approach obtains better solutions for the benchmark problems.

Market segmentation has been studied from a multiobjective perspective as well. Recent approaches using multiple dissimilarity matrices can be found, for instance, in Brusco et al. (2002), Brusco and Cradit (2005), Liu et al. (2005) and Caballero et al. (2011).

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 one of them 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 this features because it needs to apply different product marketing strategies. We first present the attribute formulation to represent the specific problem. From this we present an iterated greedy algorithm that is composed of two main phases, destruction and reconstruction, and uses a variable neighborhood search to improve the solution. As we will see, iterated greedy is a very simple technique that at the same time produces high quality solutions.

3.1 Formulation of attributes

We define a set of customers $V = \{1, 2, \ldots, n\}$, a set of segments $K = \{1, 2, \ldots, p\}$, a set of types of products (SKU) $S = \{1, 2, \ldots, l_1\}$, a set C of types of contracts and a set E of types of stores, where $C = \{1, 2, \ldots, l_2\}$ and $E = \{1, 2, \ldots, l_3\}$. Note that l_1, l_2 and l_3 are the number of different

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 (measured in number of boxes) of SKU s demanded by customer i, $i \in V$, $s \in S$; a type of contract c_i and a type of store e_i for customer i; $i \in V$, where $c_i \in C$ and $e_i \in E$. Also note that due to the fact that p clusters are sought, we will use p and k indistinctly, thus referring to k-means is equivalent to referring to p-means.

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

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

where $f_{disp}(X), f_{sku}(X), f_{cont}(X)$ and $f_{est}(X)$ are 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: For measuring the dispersion of a given partition X we consider the sum of intra-cluster distances (Brusco and Stahl, 2005). This is, for each segment $q \in K$, we consider 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 literature.

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

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:

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

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.

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

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 and will be equal to one otherwise. 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 $h_{ij} = 1$ ($g_{ij} = 0$) 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.

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

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

It has to be noted that all these dimensions represent a specific objective function that follows the business practices of the firm. While these four dimensions are measured in different scales, an a posteriori multiobjective approach (for example, producing a Pareto front) is not acceptable to the firm due to its inherent complexity with four objectives (and the thousands of potentially non-dominated points in the objective space). A convex linear combination of the four dimensions, even if measured in different units, is a much more user-friendly approach, once the most suitable values of $\alpha_1, \ldots, \alpha_4$ have been agreed upon.

4 Proposed Heuristics

The proposed heuristics in this work are based on an Iterated Greedy Algorithm (IG) Ruiz and Stützle (2007) which we have called IGACS (Iterated Greedy Algorithm for Customer Segmentation). IG is a method related to the Iterated Local Search of Lourenço et al. (2002) and Stützle (1998) which is a metaheuristic 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. One of the most important advantages of the IG algorithm is its simplicity, and its extension property in order to be applicable to several problems. For example, Ruiz and Stützle (2007, 2008); Pan et al. (2008); Fanjul-Peyro and Ruiz (2010); Urlings et al. (2010) obtained state-of-the-art results for different scheduling problems. Iterated Greedy methods have been proposed under different names for example in Jacobs and Brusco (1995) and in Marchiori and Steenbeek (2000). Other authors refer to IG as "ruin and recreate" (Schrimpf et al., 2000) or even "iterative flattening" (Cesta et al., 2000). In any case, after the work of Ruiz and Stützle (2007), many other authors, specially in the field of scheduling, have been applying this methodology with good results. Examples are Yuan et al. (2008); Fubito et al. (2008); Pan et al. (2008); Framiñán (2008); Kahraman et al. (2010); Ying and Cheng (2010) and more recently Lozano et al. (2011) and Ribas et al. (2011).

IG 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 (in our problem this solution represents a partition obtained using the p-means algorithm of Hartigan and Wong 1979). Then, the reconstruction procedure applies a greedy constructive heuristic to reconstruct the partition. IG iterates over these steps until some stopping criterion is met. In this work we applied a local search procedure, once the p-means algorithm is applied and after the reconstruction phase of the IGACS, to improve the given partition.

Pseudocode 1 shows the steps of this Iterated Greedy Algorithm for Customer Segmentation (IGACS) proposed in this paper. As an initial step we obtain an initial partition using the aforementioned p-means algorithm of Hartigan and Wong (1979) with some modifications to handle this problem. Then a local search based on a Variable Neighborhood Search (VNS) (Hansen and Mladenovic, 2001) procedure is applied for improving this partition. The improved partition is one of the input parameters of the IGACS. At each iteration of the IGACS (Steps 4-39), D elements are removed, at random, from the solution and are saved in a vector X_R. This is the destruction phase. Then those elements are added, one by one, to the current partition by applying a greedy constructive heuristic. This is the reconstruction phase. Once the partition has been completely reconstructed, the VNS is applied again to improve the partition. Afterwards, an acceptance criterion is applied in order to decide whether the new reconstructed partition replaces the current one or not. Whenever a partition is better than the best found so far it is also replaced. The IGACS iterates until a maximum number of iterations is reached ($IterIG_{\max}$).

4.1 Initial Partition

The initial partition is obtained with a modified version of the well known p-means (k-means) algorithm of Hartigan and Wong (1979). Basically, in the p-means algorithm, p elements are selected to represent the initial centers (means) of each segment, the others $n - p$ elements are assigned to its closest center, and iteratively recalculates them based on the mean of each current segment. The p-means algorithm iterates until a stopping criterion is reached. One of the most important advantages of this algorithm is that it can find solutions quickly. The most notable disadvantage is that the solution depends strongly on the initial selection of centers. The basic p-means algorithm 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 (He et al., 2005; Huang, 1998; Guha et al., 2000). For the problem that we deal with in this paper we made some modifications to the basic p-means algorithm. Pseudocode 2 shows the general procedure for the obtention of the initial partition. We refer to this method as the MPM-Modified p-means.

One of the proposed modifications is a Greedy Randomized Adaptive Search Procedure (GRASP)

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

Input:
$IterIG_{\max}$: IGACS iterations;
X := Initial Partition
Output: X^{best} := Best partition found;

$X \leftarrow \text{VNS}(X)$;
$X^{best} \leftarrow X$;
$\tau = -1$;
while ($IterIG_{\max} > 0$) **do**
    $X' \leftarrow X$;
    for $i = 1$ to D **do**
        $i^* \leftarrow$ one_random_element(X'); //Destruction Phase
        $X_R(i) \leftarrow i^*$; $X' \leftarrow X' \setminus \{i^*\}$;
    end for
    for $i = 1$ to D **do**
        $i' \leftarrow X_R(i)$; $q^* \leftarrow \arg\min_{q \in K} \left\{\sum_{j \in X_q} f_{i'j}\right\}$; //Reconstruction Phase
        $X'_{q^*} \leftarrow X'_{q^*} \cup \{i'\}$;
    end for
    $X'' \leftarrow \text{VNS}(X')$; //Improvement Phase
    if ($f(X'') < f(X)$) **then**
        $X \leftarrow X''$;
        if ($f(X) < f(X^{best})$) **then**
            $X^{best} \leftarrow X$;
        end if
    else
        $gap = \frac{f(X) - f(X^{best})}{f(X^{best})} * 100$; //acceptance criterion for worse solutions
        if $\tau = -1$ **then**
            $\tau = 0.1 \cdot gap$ //initial value for τ
        end if
        if (($gap < \tau$) and ($gap \neq 0$)) **then**
            $X \leftarrow X''$; $++as; csa+ = f(X)$; $nas = 0$;
        else
            $++nas; cna+ = f(X)$; $as = 0$;
        end if
        if $nas = \lceil 0.2 * IterIG_{\max} \rceil$ **then**
            $\tau \leftarrow \frac{cna}{nas}$; $nas = 0$; $cna = 0$;
        else
            if $as = \lceil 0.25 * IterIG_{\max} \rceil$ **then**
                $\tau \leftarrow \frac{csa}{as}$; $as = 0$; $csa = 0$;
            end if
        end if
    end if
    $IterIG_{\max} \leftarrow IterIG_{\max} - 1$;
end while
return X^{best}.

Pseudocode 2 MPM($V, p, \beta, Iter_{\max}$)

Input:

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

Output:

X^{best} : Best partition found;

1: $X^{best} \leftarrow \varnothing$; $iter \leftarrow 0$;
2: **while** ($iter < Iter_{\max}$) **do**
3: $P \leftarrow$ get_centroids(V, p, β); //Obtains p centroids
4: $X \leftarrow$ assigment(V, P); //Obtains partition X
5: **if** ($f(X) < f(X^{best})$) **then**
6: $X^{best} \leftarrow X$;
7: **end if**
8: $iter \leftarrow iter + 1$;
9: **end while**
10: **return** X^{best}.

for attempting to find an initial configuration of centroids (Step 3). GRASP (Feo and Resende, 1995; Resende and Ribeiro, 2003) is a multi-start metaheuristic that has been successfully applied to many combinatorial optimization problems. Following the GRASP spirit, in a particular iteration, given a partial set of centroids $C = c_1, \ldots, c_q$, a greedy function $\phi(j)$ that measures the cost of assigning a node j as a centroid is computed for all unassigned nodes $j \in V \setminus C$. 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 C. Once the p centroids are selected (get_centroids(V, p, β) function), the rest of the $n - p$ elements are assigned to its closest centroid (assignment(V, P) function) using the proposed objective function (1) to measure the distance (dissimilarity) between elements. When there are no elements to assign, centroids are recalculated in such a way that new centroids correspond to the most centered elements of each segment (the one which distance between all elements of the same segment is the minimum). The current solution is updated if the new solution is better. This process continues until a maximum number of iterations is reached. The best solution is returned. The goal of introducing a GRASP-construction is to find a good configuration of initial centroids and obtain better quality solutions once the p-means algorithm has been applied. We use this method to get the initial partition for IGACS because p-means is one of the best known algorithms for clustering problems due to its easy implementation and fast convergence to good solutions. As this solution depends on the initial configuration of centroids, we tried three different strategies within the GRASP construction phase to get this initial configura-

tion. The main idea of these strategies is based on greedy heuristics for the p-dispersion problem developed by Erkut et al. (1994) and correspond to the following:

- **Strategy 1**: Select the p most disperse elements (considering only the distance attribute). Under this strategy, first the two nodes i and j whose d_{ij} is minimum are chosen and added to C. Then, for the rest of the $n - |C|$ elements, (C is the set of the selected elements until now) we calculate the minimum distance between each one of them and the elements belonging to C, in other words, the greedy function is computed as $\phi(j) = \min\{d_{ji} : i \in C\}$. The RCL is formed by the elements of highest value. This procedure is carried out until $|C| = p$.

- **Strategy 2**: Select the p most dissimilar elements. It consists of the same procedure except that now we directly use the dissimilarity function (1) to establish the dissimilarity between elements. In this regard, the greedy function is computed as $\phi(j) = \min\{f_{ji} : i \in C\}$.

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

In order to increase the diversity of solutions, we implemented a GRASP philosophy where both, the first pair of selected items and the rest of the $p - 2$ elements are chosen from a restricted list of candidates (RCL) formed using a quality parameter β. Once p centroids are selected, the p-means algorithm is applied to obtain a partition. Some modifications are made to this algorithm. First, instead of using the mean of each segment like a center or centroid, we use medians, this is, the most centered (the least dissimilar) element of each segment according to the weighted sum of dissimilarities (Equation 1). Second, the distance between elements represents the dissimilarity with respect to the four attributes.

4.2 Local Search

The Variable Neighborhood Search (VNS, Hansen and Mladenovic 2001) is a well known metaheuristic used for solving optimization problems whose basic idea is the systematic change of neighborhood within a local search. A neighborhood structure, in the solution space Π, is an application $\Delta : \Pi \to 2^{\Pi}$ that associates each solution $X \in \Pi$ to a subset of solutions (neighborhood) $\Delta(X) \subset \Pi$ which are considered as a neighbors of X. We implemented a VNS procedure to improve the initial partition found by the modified p-means algorithm and the reconstructed partition obtained in the reconstruction phase of the IGACS. This improvement procedure is composed of two simple local searches or neighborhoods. Pseudocode 3 shows the general VNS procedure applied in this paper.

The first neighborhood applied is based on insertion moves (Pseudocode 4), this is, for a segment $q_1 \in K$ (chosen at random) we take one element $i \in X_{q_1}$ and insert it into another segment q_2, where $q_2 \neq q_1$. If the merit function ϕ (Step 15) is greater than zero the move is accepted (first-found

Pseudocode 3 VNS(X)

Input:
X : p-partition;

Output:
X: p-partition;

1: $save_{tot} \leftarrow 1$; $\epsilon \leftarrow 0$; iter $\leftarrow 0$;
2: **while** (($save_{tot} > 0$) and (iter < 3)) **do**
3: $X_{ins} \leftarrow$ insertion(X); //First Neighborhood
4: $X_{int} \leftarrow$ interchange(X); //Second Neighborhood
5: $save_{tot} \leftarrow f(X_{ins}) + f(X_{int})$;
6: **if** ($save_{tot} = 0$) **then**
7: iter $\leftarrow$ iter + 1;
8: **end if**
9: **end while**
10: **return** $X_{int} = (X_1, \ldots, X_p)$.

strategy). If not, another segment is taken and the move is evaluated again. When the total number of segments evaluated is equal to p, the best solution found is returned as the best solution found until now.

For the interchange neigborhood (Pseudocode 5) we take an element i from a segment $q_1 \in K$ (chosen at random) and one element j from a different segment $q_2 \in K$, $q_1 \neq q_2$. If a merit function (step 17) is greater than zero then swap both elements i and j (first-found strategy) and another element $i \in X_{q_1}$ is selected to be swaped with another element $j \in X_{q_2}$. Otherwise, another element j is taken from X_{q_2} and the move is evaluated again until a swap move is accepted. This local search finishes when the number of segments selected at random, represented by q_1, is equal to $\lceil \frac{p}{2} \rceil$.

4.3 Destruction, Reconstruction and Acceptance Criterion

The destruction phase, detailed in steps 6-9 of Pseudocode 1 is simple: given a partition, we destroy or remove some elements at random. In this phase, D elements are removed from the given partition at random. The removed elements are kept in list $X_R(i)$.

Once D elements have been removed, the reconstruction phase is carried out (steps 10-13 of Pseudocode 1). Basically, each removed element is tested in all segments and greedily reinserted into the segment that increases the partition dissimilarity by the smallest possible amount. Once that all elements are added, we apply the VNS procedure to the new partition obtained. If the total dissimilarity of the new partition is less than the current one, the current solution is updated. Then, if the total dissimilarity of this current solution is less than the best solution found until now, the best solution is updated too.

As with most IG methods, solutions are not only accepted if they are strictly better, as this quickly results in a premature convergence to strong local optimum solutions. Therefore, when a

Pseudocode 4 Insertion(X).

Input: $X = \{X_1, \ldots, X_p\}$;
Output: $X = \{X_1, \ldots, X_p\}$;

$m \leftarrow 0$; improve $\leftarrow$ YES; {segment counter and improvement status}
while $(m \leq p)$ **do**
  if (improve = YES) **then**
    $\bar{P} \leftarrow P$; $//P = \{1, 2, \ldots, p\}$
  end if
  improve $\leftarrow$ NO; $m \leftarrow m + 1$;
  Select a segment $q_1 \in \bar{P}$ at random;
  $\bar{P} \leftarrow \bar{P} \setminus \{q_1\}$; $\bar{Y} \leftarrow X_{q_1}$;
  $z \leftarrow \min\{|\bar{X}_{q_1}|, |\frac{n}{p}|\}$;
  while $(z > 0)$ **do**
    Select an element $i \in \bar{Y}$ at random;
    $\bar{Y} \leftarrow \bar{Y} \setminus \{i\}$; $z \leftarrow z - 1$; $r \leftarrow 1$;
    while $((r \leq q_1)$ and (improve = NO)) **do**
      if $(r \neq q_1)$ **then**
        $\phi(i, r) = \sum_{j \in X_{q_1}} f_{ij} - \sum_{j \in X_r} f_{ij}$;
        if $(\phi(i, r) > 0)$ **then**
          $X_{q_1} \leftarrow X_{q_1} \setminus \{i\}$; $X_r \leftarrow X_r \cup \{i\}$;
          improve $\leftarrow$ YES;
        end if
      end if
      $r \leftarrow r + 1$;
    end while
  end while
end while
return $X = \{X_1, \ldots, X_p\}$;

Pseudocode 5 Interchange(X).

Input: $X = \{X_1, \ldots, X_p\}$;
Output: $X = \{X_1, \ldots, X_p\}$;

$m \leftarrow 0$; improve $\leftarrow$ YES; {Segment counter and improvement status}
while ($m \leq \frac{p}{3}$) **do**
    if (improve = YES) **then**
        $\bar{P} \leftarrow P$;
    end if
    improve $\leftarrow$ NO; $t \leftarrow 0$; $m \leftarrow m + 1$;
    Select a segment $q_1 \in \bar{P}$ at random;
    $\bar{P} \leftarrow \bar{P} \setminus \{q_1\}$; $\bar{Y} \leftarrow X_{q_1}$;
    $z \leftarrow |X_{q_1}|$;
    while ($z > 0$) **do**
        Select an element $i \in \bar{Y}$ at random;
        $\bar{Y} \leftarrow \bar{Y} \setminus \{i\}$;
        $z \leftarrow z - 1$; $r \leftarrow 1$;
        while (($r \leq q_1$) and (improve = NO)) **do**
            if ($r \neq q_1$) **then**
                for each element $j \in X_r$ **do**
                    $$\phi(i,j) = \left(\sum_{q \in X_{q_1}} f_{iq} - \sum_{q \in X_r} f_{iq} \right) + \left(\sum_{q \in X_r} f_{jq} - \sum_{q \in X_{q_1}} f_{jq} \right);$$
                    if ($\phi(i,j) > 0$) **then**
                        $X_{q_1} \leftarrow X_{q_1} \setminus \{i\}$; $X_r \leftarrow X_r \cup \{i\}$;
                        $X_r \leftarrow X_r \setminus \{j\}$; $X_{q_1} \leftarrow X_{q_1} \cup \{j\}$;
                        improve $\leftarrow$ YES;
                    end if
                end for
            end if
            $r \leftarrow r + 1$;
        end while
    end while
end while
return $X = \{X_1, \ldots, X_p\}$;

new solution is found and it is not better than the current one, an acceptance criterion is applied. Basically a new non best solution can be accepted if its relative percentage deviation is less than or equal to a certain threshold value τ computed as follows:

$$\tau = \begin{cases} \frac{csa}{as} \text{ if } as = \lceil 0.25 * IterIG_{\max} \rceil \\ \\ \frac{cna}{nas} \text{ if } nas = \lceil 0.20 * IterIG_{\max} \rceil \end{cases}$$

where csa/as (cna/nas) represents the average dissimilarity after as (nas) iterations 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.

5 Computational Results and Statistical Analyses

The IGACS was implemented in C++ under the Ubuntu-Linux 9.04 operating system. The experiments were carried out on a Dell server with Intel Core(TM) 2 Quad 1 processor and 2.4 Ghz. CPU with 3.2 Gb. of RAM. The instances were generated using a uniform distribution for geographical coordinates and real-world instance information for the rest of the attributes. For all experiments, we created 30 instances for each one of three different sizes $n = \{1000, 3000, 5000\}$. Therefore, a total of 90 instances are generated. Each instance can be tested with several number of segments. In this paper we have considered three different number of segments $p = \{20, 25, 30\}$.

A total of 6 different vectors for α (α_r) are tested, namely (0.25,0.25,0.25,0.25), (0.10,0.40,0.40,0.10), (1.00,0.00,0.00,0.00), (0.00,1.00,0.00,0.00), (0.00,0.00,1.00,0.00) and (0.00,0.00,0.00,1.00). 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 or $AVRPD$ as follows:

$$AVRPD = \frac{1}{30} \sum \frac{f(X) - f(X^{best})}{f(X^{best})} * 100 \tag{7}$$

where $f(X)$ is the resulting objective function value found for each one of the 30 instances that exist for each size n and $f(X^{best})$ 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 p, and a specified α_r vector.

As a first experiment, we evaluate the p-means algorithm with our presented modifications (PMP algorithm). Table 1 shows the average relative percentage deviation found for each combination of α averaged across all instance sizes n (1000, 3000, and 5000), and number of p segments (20, 25, and 30). The values are displayed according to the Strategy employed and the quality parameter

values for the GRASP method (β). The MPM algorithm (selection of centroids and assignment) is executed 50 times for each instance and the average solution found for each one is shown. This means that each cell contains the average of 50 replicates over 90 instances tested for each segment size (13,500 results averaged at each cell).

Table 1: Average Relative Percentage Deviations for the MPM algorithm as a function of α_r, Strategy and β values of the GRASP.

		β for the GRASP method in MPM						
α_r	GRASP strategy	0	0.2	0.4	0.6	0.8	1	Average
(0.25, 0.25, 0.25, 0.25)	Strategy 1	8.11	0.42	0.77	1.73	3.37	5.30	3.28
equal weights	Strategy 2	11.67	3.78	3.36	3.71	4.87	5.36	5.46
	Strategy 3	18.34	8.32	9.28	6.63	4.85	5.26	8.78
	Average	12.71	4.17	4.47	4.02	4.36	5.30	
(0.10, 0.40, 0.40, 0.10)	Strategy 1	13.22	1.74	2.06	2.35	3.18	3.89	4.41
commercial weights	Strategy 2	36.98	15.51	9.04	4.88	3.84	3.96	12.37
	Strategy 3	4.28	0.64	0.80	1.22	1.89	2.53	1.89
	Average	18.16	5.96	3.97	2.81	2.97	3.46	
(1.00, 0.00, 0.00, 0.00)	Strategy 1	4.28	0.64	0.80	1.22	1.89	2.53	1.89
dispersion	Strategy 2	4.28	0.63	0.84	1.27	1.91	2.42	1.89
	Strategy 3	19.24	8.08	6.58	4.05	2.79	2.43	7.19
	Average	9.27	3.12	2.74	2.18	2.20	2.46	
(0.00, 1.00, 0.00, 0.00)	Strategy 1	19.26	2.37	2.58	2.59	2.57	2.52	5.32
purchase volume	Strategy 2	63.24	31.89	20.48	8.62	2.80	2.62	21.61
	Strategy 3	32.91	21.19	24.08	15.47	3.72	2.67	16.67
	Average	38.47	18.49	15.71	8.89	3.03	2.60	
(0.00, 0.00, 1.00, 0.00)	Strategy 1	139974.00	54203.14	44797.07	40034.56	50092.53	17194.46	57715.96
type of contract	Strategy 2	3260.86	0.00	0.00	0.00	0.00	47603.09	8477.32
	Strategy 3	3833.97	0.00	0.00	0.00	0.00	33703.41	6256.23
	Average	49022.94	18067.71	14932.36	13344.85	16697.51	32833.65	
(0.00, 0.00, 0.00, 1.00)	Strategy 1	3127.04	72.01	64.81	106.21	84.94	83.40	589.74
type of store	Strategy 2	655.89	0.00	0.00	0.00	0.00	126.62	130.42
	Strategy 3	588.80	0.00	0.00	0.00	0.00	82.94	111.96
	Average	1457.24	24.00	21.60	35.40	28.31	97.65	

We can observe that Strategy 1 obtained better solutions for most cases, specially when $\beta = 0.2$. We observe aberrant results in the cases where the maximum weight is given to the attribute of type contract or store. As we will see 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 removed.

We carried out some statistical testing to guarantee that results using Strategy 1 are indeed statistically significant. First we separate the results into four blocks (according to the combination

of α values where Strategy 1 was better) and study a single factor of each block, the values of β correspond to the levels (6 levels) and the average relative percentage deviation is the response variable. We analyzed the data using the Kruskal-Wallis non-parametric test. Figures 1-4 show the box plot and the pairwise comparisons for each block analized using the Dunn's Test. The graph on the left displays boxplots of the groups with their sign confidence intervals for the medians. This graph is interesting because one can visually see the locations of the groups with respect to the others. The graph on the right displays the non-absolute group mean rank standardized differences. This second graph is also extremely useful because one can look at not only the magnitude of the group differences but the direction as well. It also displays the positive and negative critical z-values so one can see if a difference is significant. If we observe GAP(0.2) (at the Y-axis) in Figures 1-3, we can see that GAP(0.6), GAP(0.8) and GAP(1) are significant since this distance goes beyond the z-value, while GAP(0.4) does not. However, in Figure 4 we can observe that there is not a significant difference between groups since all comparisons are inside the z-value interval.

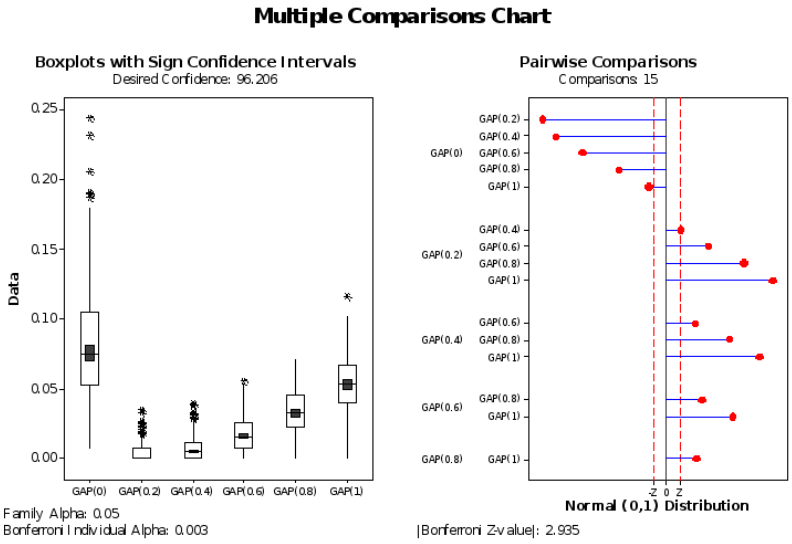

Figure 1: Box plot and pairwise comparisons for each level (β values) corresponding to the block when α_r=(0.25, 0.25, 0.25, 0.25) at a 95% of confidence level.

Apart from non-parametric analysis, we carry out a parametric Analysis of Variance after considering the first four α_r values, β, n, p and the Strategy level as factors, being the average relative percentage deviation the response variable. The corresponding means plot of the interaction between the Strategy and β factors is shown in Figure 5. As can be seen, Strategy 1 is superior and β=0.2 provide the best overall results.

As a second experiment, we evaluate the number of elements (D) to remove from the solution (partition) in the destruction phase of the IGACS algorithm. For this experiment we use the 30 instances of each size, setting the number of segments again to $p = \{20, 25, 30\}$. We employ the

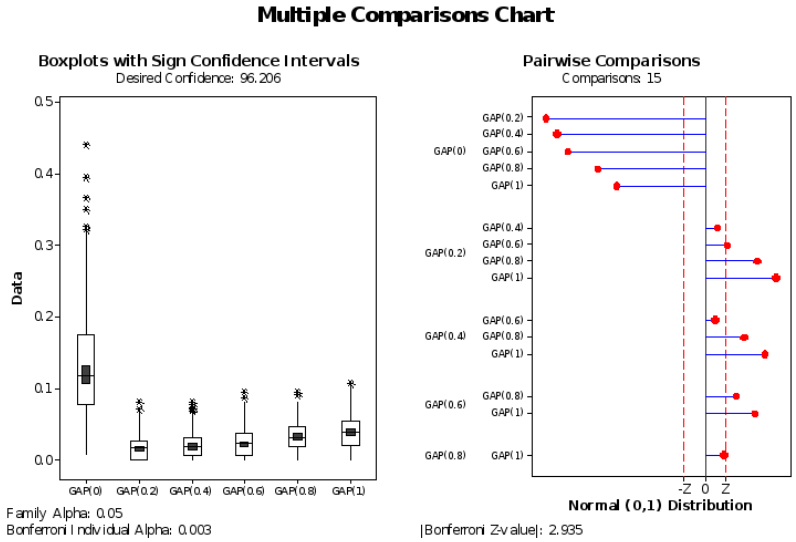

Figure 2: Box plot and pairwise comparisons for each level (β values) corresponding to the block when α_r=(0.10, 0.40, 0.40, 0.10) at a 95% of confidence level.

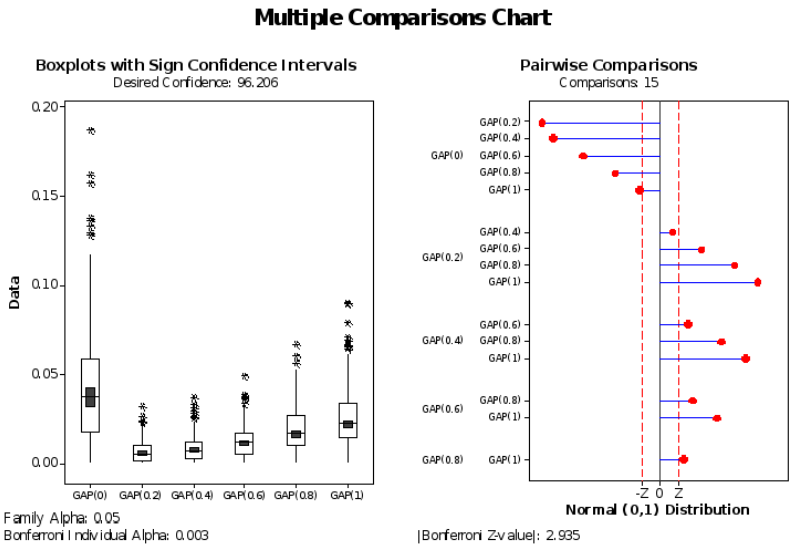

Figure 3: Box plot and pairwise comparisons for each level (β values) corresponding to the block when α_r=(1.00, 0.00, 0.00, 0.00) at a 95% of confidence level.

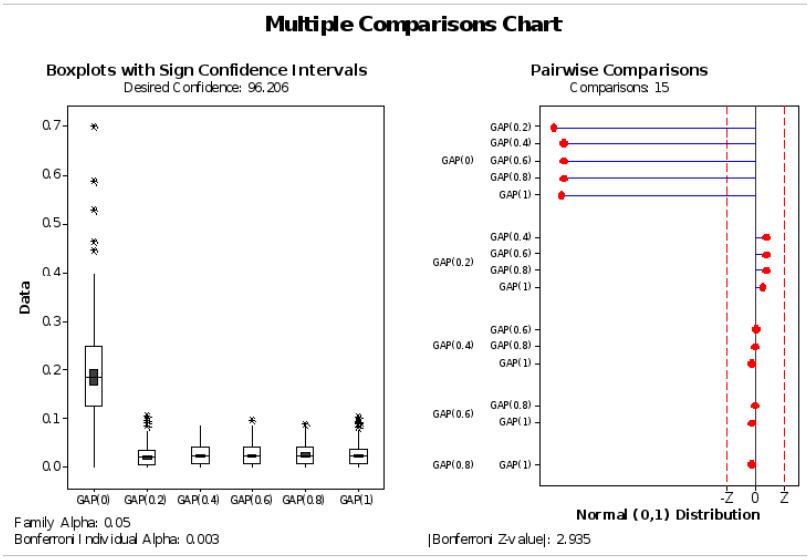

Figure 4: Box plot and pairwise comparisons for each level (β values) corresponding to the block when α_r=(0.00, 1.00, 0.00, 0.00) at a 95% of confidence level.

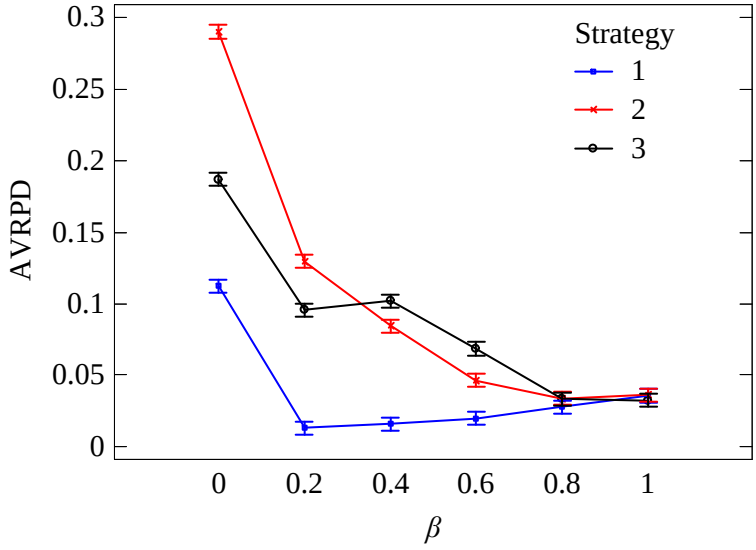

Figure 5: Means plot with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for the interaction between Strategy and β factors. MPM algorithm.

previous MPM algorithm to create the initial partition fixing $\beta = 0.2$ (since this obtained the best evaluation in the previous experiment in general) and using Strategy 1 for the cases where α_r values do not assign the maximum weight to the type contract or store attributes, and Strategy 2 when these two cases occur. The MPM algorithm (GRASP for centroids and modified p-means) is executed 50 times and IGACS is iterated 15 times ($IterIG_{\max} = 15$). We evaluate four different values for D based on a certain percentage of the instance size. For example, for an instance of size $n = 1000$, a 5% value for D equals to 50 elements to remove. If $n = 3000$ then $D = 150$ and if $n = 5000$ the respective value of D is equal to 250 elements.

Table 2 shows the average relative percentage deviation (grouped for each combination of α values) using different size of D. Each table contains the average deviations calculated over 15 replicates, 90 instances and for three values of p (4,050 results averaged at each cell).

Table 2: Average Relative Percentage Deviations for the IGACS algorithm as a function of α_r and D.

	D			
α_r	5%	10%	15%	20%
(0.25, 0.25, 0.25, 0.25)	0.27	0.22	0.17	0.14
(0.10, 0.40, 0.40, 0.10)	0.09	0.19	0.15	0.10
(1.00, 0.00, 0.00, 0.00)	0.38	0.39	0.33	0.30
(0.00, 1.00, 0.00, 0.00)	0.28	0.20	0.17	0.12
(0.00, 0.00, 1.00, 0.00)	0.09	0.13	0.42	0.58
(0.00, 0.00, 0.00, 1.00)	3.13	0.83	1.67	0.56
Average	0.71	0.33	0.49	0.30

In general, the results are better removing 20% (depending of the size of the instance) of the elements from the solution. Except on the case where attributes of purchase volume and type of contract have greater weight (0.10, 0.40, 0.40, 0.10), and the maximum weight is assigned to the type contract (0.00, 0.00, 1.00, 0.00). Statistical testing shows that $D = 20\%$ gives the overall best results.

For the final experiment we test, for each α_r value the proposed MPM algorithm (Strategy 1 for the first four α_r values and Strategy 2 for the last two and $\beta = 0.2$ in all cases) with a total of 50 replications per instance. We also test the presented VNS algorithm, i.e., the result obtained in the MPM algorithm and then a single run of the VNS method. Lastly, the IGACS method is also run 15 times after the initialization with the MPM+VNS methods using $D = 20\%$. Tables 3-8 show the full results of the average relative percentage deviations and computational time (in seconds). The results are further detailed and grouped by n and p.

Table 3: Average Relative Percentage Deviation and computational time for the proposed algorithms when $\alpha_r = (0.25, 0.25, 0.25, 0.25)$.

n	p	Methods (AVRPD)			Time (Seconds)			
		MPM	VNS	IGACS	MPM	VNS	IGACS	Total
1000	20	8.17	1.89	0.20	3.37	1.6	6.92	11.89
	25	10.53	1.90	0.14	3.59	1.86	6.74	12.19
	30	12.26	2.52	0.22	3.87	1.93	7.53	13.33
3000	20	6.37	0.51	0.18	28.26	22.72	59.45	110.43
	25	7.97	0.55	0.12	27.96	23.83	60.36	112.15
	30	8.96	0.61	0.08	28.38	24.88	64.83	118.10
5000	20	5.70	0.40	0.10	85.29	68.38	151.57	305.24
	25	6.98	0.50	0.12	86.33	73.92	159.22	319.47
	30	8.39	0.53	0.08	85.68	78.47	164.66	328.81
Average		8.37	1.05	0.14	39.19	33.07	75.70	147.96

Table 4: Average Relative Percentage Deviation and computational time for each method when $\alpha_r = (0.1, 0.4, 0.4, 0.1)$.

n	p	Methods (AVRPD)			Time (Seconds)			
		MPM	VNS	IGACS	MPM	VNS	IGACS	Total
1000	20	16.02	1.27	0.15	3.34	3.41	8.69	15.43
	25	19.60	1.87	0.17	3.57	3.61	8.89	16.07
	30	23.03	1.86	0.13	3.84	3.33	9.20	16.38
3000	20	11.71	0.44	0.07	28.02	46.39	80.15	154.56
	25	15.07	0.63	0.08	28.17	39.32	83.51	151.00
	30	17.79	0.68	0.09	28.96	42.16	84.48	155.60
5000	20	10.71	0.52	0.10	78.29	124.42	201.36	404.08
	25	13.46	0.51	0.07	79.41	128.57	203.46	411.43
	30	16.12	0.61	0.08	81.23	130.98	215.48	427.69
Average		15.95	0.93	0.10	37.20	58.02	99.47	194.69

Table 5: Average Relative Percentage Deviation and computational time for each method when $\alpha_r = (1.0, 0.0, 0.0, 0.0)$.

n	p	Methods (AVRPD)			Time (Seconds)			
		MPM	VNS	IGACS	MPM	VNS	IGACS	Total
1000	20	4.06	1.47	0.42	4.72	0.76	5.28	10.75
	25	5.12	2.35	0.43	4.76	0.60	5.23	10.60
	30	5.62	2.50	0.36	4.91	0.74	5.46	11.12
3000	20	2.59	0.50	0.31	53.71	14.29	43.08	111.08
	25	3.01	0.63	0.32	51.43	14.45	43.28	109.16
	30	3.74	0.70	0.30	49.68	13.83	44.41	107.92
5000	20	1.84	0.31	0.16	175.91	37.34	105.50	318.76
	25	2.52	0.46	0.17	168.03	39.92	115.28	323.23
	30	2.81	0.59	0.25	163.18	40.71	116.90	320.79
Average		3.48	1.06	0.30	75.15	18.07	53.83	147.05

Table 6: Average Relative Percentage Deviation and Computational time for each method when $\alpha_r = (0.0, 1.0, 0.0, 0.0)$.

n	p	Methods (AVRPD)			Time (Seconds)			
		MPM	VNS	IGACS	MPM	VNS	IGACS	Total
1000	20	18.53	1.34	0.19	3.40	3.90	9.69	16.99
	25	21.46	1.60	0.18	3.69	3.82	9.86	17.36
	30	23.90	1.93	0.16	3.91	3.83	10.40	18.14
3000	20	15.53	0.39	0.08	29.77	57.18	94.89	181.84
	25	17.69	0.45	0.10	30.25	53.93	96.72	180.90
	30	20.57	0.53	0.11	30.68	54.32	102.12	187.12
5000	20	14.44	0.44	0.11	83.83	159.33	248.58	491.75
	25	16.13	0.44	0.09	83.78	161.31	242.56	487.64
	30	18.94	0.50	0.09	83.24	161.56	260.59	505.40
Average		18.58	0.85	0.12	39.17	73.24	119.49	231.90

Table 7: Average Relative Percentage Deviation and computational time for each method when $\alpha_r = (0.0, 0.0, 1.0, 0.0)$.

n	p	Methods (AVRPD)			Time (Seconds)			
		MPM	VNS	IGACS	MPM	VNS	IGACS	Total
1000	20	346.67	3.65	0.77	4.40	0.33	6.55	11.28
	25	153.33	0.00	0.00	4.75	0.71	3.89	9.35
	30	36.67	0.00	0.00	5.16	1.31	1.43	7.90
3000	20	1348.34	32.06	0.24	39.84	2.49	44.74	87.07
	25	1559.29	36.89	1.35	42.19	1.82	60.58	104.59
	30	1694.59	28.97	1.75	44.92	2.00	187.22	234.14
5000	20	1538.36	23.17	0.00	113.69	6.17	68.20	188.05
	25	1871.90	41.61	0.18	118.73	8.25	86.86	213.83
	30	2167.20	40.45	0.97	123.75	7.42	168.27	299.43
Average		1190.71	22.98	0.58	55.27	3.39	69.75	128.41

Table 8: Average Relative Percentage Deviation and computational time for each method when $\alpha_r = (0.0, 0.0, 0.0, 1.0)$.

n	p	Methods (AVRPD)			Time (Seconds)			
		MPM	VNS	IGACS	MPM	VNS	IGACS	Total
1000	20	1344.55	39.48	0.91	4.76	0.46	2.96	8.19
	25	1559.08	42.48	1.70	5.02	0.38	5.25	10.64
	30	1701.55	27.46	2.37	5.32	0.50	10.45	16.26
3000	20	1651.51	64.10	0.00	41.49	2.38	27.01	70.88
	25	2058.47	29.62	0.00	43.65	2.30	32.21	78.16
	30	2444.37	68.16	0.06	46.13	3.19	36.34	85.67
5000	20	1712.61	14.00	0.00	120.27	14.48	94.95	229.70
	25	2158.83	0.80	0.00	126.13	16.61	108.79	251.53
	30	2591.80	5.45	0.00	130.54	19.24	119.24	269.01
Average		1913.64	32.39	0.56	58.14	6.61	48.58	113.34

As can be seen, the results of the MPM 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 $\alpha_r = (0.0, 0.0, 1.0, 0.0)$ or $\alpha_r = (0.0, 0.0, 0.0, 1.0)$. The additional CPU time needed increases substantially, specially for bigger problems. For example, MPM needs 39.19 seconds on average for $\alpha_r = (0.25, 0.25, 0.25, 0.25)$ and VNS requires an additional 33.07 seconds on average. The results of the proposed IGACS are much better and improve those of VNS by a large margin. This is important for the last two cases of α_r as IGACS reports average relative percentage deviations of 0.58 and 0.56, respectively whereas VNS obtains much larger deviations of 22.98 and 32.39, respectively. Of course, such improvements come at a computational cost, as the additional CPU time of IGACS exceeds 260 seconds in the worst case for largest instances of 5000 clients. In any case, the total CPU time of applying IGACS (which comes after applying MPM, VNS and then the IGACS) rarely exceeds 500 seconds in the worst case, which, considering the large size of the real clustering problem dealt with in this paper (up to 5000 clients) is readily acceptable.

A final question remains about the comparison between the proposed approaches and the segmentation methods used by the firm. While we cannot provide a quantitative analysis, we can clearly state that the proposed approach is vastly superior. This is motivated by the fact that the methods employed by the firm were basically manual and just 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.

6 Conclusions

In this paper we proposed a formulation of attributes to represent a market segmentation problem that considers multiple attributes, which arose from a real-world application in a beverage distribution firm. We have also proposed an Iterated Greedy Algorithm (IGACS) to solve the problem efficiently. Also, we used an adaptation of the well known p-means algorithm to create partitions to our problem. The modifications include three different GRASP-based strategies to select the initial configuration of centroids in an attempt to obtain better solutions. A local search based on a variable neighborhood search procedure is developed and implemented after the p-means algorithm to improve the solution, and it is incorporated after the reconstruction phase of the IGACS for improving solutions.

Comprehensive computational and statistical experiments have been performed to fix some parameters of the adapted p-means algorithm and IGACS method. Sound statistical tests were made in order to guarantee that the observed differences in the average results are indeed statistically

significant. Empirical results shown that applying the VNS procedure after the MPM algorithm improve the dissimilarity of the partition significantly. Also, IGACS can improve even more the dissimilarity of the partition found by the VNS procedure. The three methods obtain excellent results in less of 261 seconds on average.

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, and IT511-10. Rubén Ruiz is partially funded by the Spanish Ministry of Science and Innovation, under the project "SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances" with reference DPI2008-03511/DPI. Rubén Ruiz also thanks the IMPIVA - Institute for the Small and Medium Valencian Enterprise, for the project TASER with reference IMDEEA/2011/142. The authors also acknowledge the support of Fabián López (Grupo ARCA) for providing the information needed to carry out the experiments.

References

Berkhin, P. (2006). Survey of clustering data mining techniques. In Kogan, J., Nicholas, C., and Teboulle, M., editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 25–71. Springer, Berlin, Germany.

Brusco, M. J. and Cradit, J. D. (2005). Bicriterion methods for partitioning dissimilarity matrices. *British Journal of Mathematical and Statistical Psychology*, 58(2):319–332.

Brusco, M. J., Cradit, J. D., and Stahl, S. (2002). A simulated annealing heuristic for a bicriterion partitioning problem in market segmentation. *Journal of Marketing Research*, 39(1):99–109.

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.

Cano, J. R., Cordón, O., Herrera, F., and Sánchez, L. (2002). A GRASP algorithm for clustering. In Garijo, F. J., Riquelme, J. C., and Toro, M., editors, *Advances in Artificial Intelligence – IBERAMIA 2002*, volume 2527 of *Lecture Notes in Computer Science*, pages 214–223. Springer, Berlin, Germany.

Cesta, A., Oddi, A., and Smith, S. F. (2000). Iterative flattening: A scalable method for solving multi-capacity scheduling problems. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 742–747. AAAI Press / The MIT Press.

Chiu, C.-Y., Chen, Y.-F., Kuo, I.-T., and Chun Ku, H. (2009). An intelligent market segmenta-

tion 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.

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.

Framiñán, J. M. y Leisten, R. (2008). Total tardiness minimization in permutation flow shops: a simple approach based on a variable greedy algorithm. *International Journal of Production Research*, 46(22):6479–6498.

Fubito, T., Kenji, S., and Juichi, M. (2008). An iterated greedy algorithm for the node placement problem in bidirectional Manhattan street networks. In Keijzer, M., editor, *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08)*, pages 579–584, Atlanta.

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

Hansen, P. and Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

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.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with mixed numeric and categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304.

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.

Kahraman, C., Engin, O., Kaya, I., and Elif Öztürk, R. (2010). Multiprocessor task scheduling in multistage hybrid flow-shops: A parallel greedy algorithm approach. *Applied Soft Computing*, 10(4):1293–1300.

Liu, Y., Ram, S., and Lusch, R. (2005). A unified market segmentation method for generating pareto optimal solutions sets. In *Proceedings of the 15th Annual Workshop on Information Technolgies and Systems (WITS)*, pages 249–243, Las Vegas. URL: http://ssrn.com/abstract=882884.

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.

Merz, P. (2003). An iterated local search approach for minimum sum-of-squares clustering. In Berthold, M. R., Lenz, H.-J., Bradley, E., Kruse, R., and Borgelt, C., editors, *Advances in Intelligent Data Analysis V*, volume 2810 of *Lecture Notes in Computer Science*, pages 286–296. Springer, Berlin, Germany.

Pan, Q.-K., Wang, L., and Zhao, B.-H. (2008). An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *The International Journal of Advanced Manufacturing Technology*, 38(7–8):778–786.

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.

Resende, M. G. C. and Ribeiro, C. (2003). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G. A., editors, *Handbook of Metaheuristics*, International Series in Operations Research and Management Science, chapter 8, pages 219–249. Kluwer, Boston.

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.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171.

Selim, S. Z. and Ismail, M. A. (1984). K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):81–87.

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.

Stützle, T. (1998). Applying iterated local search to the permutation flow shop problem. Technical report AIDA–98–04, Department of Computer Science, Intellectis Group, TU Darmstadt, Darmstadt, Germany.

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.

Wedel, M. and Kamakura, W. (1998). *Market Segmentation: Conceptual and Methodological Foundations*. Kluwer, Boston.

Ying, K.-C. and Cheng, H.-M. (2010). Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications*, 37(4):2848–2852.

Yuan, Z., Fugenschuh, A., Homfeld, H., Balaprakash, P., Stützle, T., and Schoch, M. (2008). Iterated greedy algorithms for a real-world cyclic train scheduling problem. In Blesa, M. J., Blum, C., Cotta, C., Fernández, A. J., Gallardo, J. E., Roli, A., and Sampels, M., editors, *Hybrid Metaheuristics: 5th International Workshop, HM 2008, Proceedings*, volume 5296 of *Lecture Notes in Computer Science*, pages 102–116. Springer, Berlin, Germany.