A mixed-integer programming approach to the clustering problem with an application in customer segmentation

Burcu Sağlam a, F. Sibel Salman a, Serpil Sayın b, Metin Türkay a,∗

a Department of Industrial Engineering, College of Engineering, Koç University, Rumelifeneri Yolu, 34450 Sarıyer, Istanbul, Turkey
b College of Administrative Sciences and Economics, Koç University, Rumelifeneri Yolu, 34450 Sarıyer, Istanbul, Turkey

Received 1 November 2004; accepted 26 April 2005
Available online 15 November 2005

Abstract

This paper presents a mathematical programming based clustering approach that is applied to a digital platform company’s customer segmentation problem involving demographic and transactional attributes related to the customers. The clustering problem is formulated as a mixed-integer programming problem with the objective of minimizing the maximum cluster diameter among all clusters. In order to overcome issues related to computational complexity of the problem, we developed a heuristic approach that improves computational times dramatically without compromising from optimality in most of the cases that we tested. The performance of this approach is tested on a real problem. The analysis of our results indicates that our approach is computationally efficient and creates meaningful segmentation of data. © 2005 Elsevier B.V. All rights reserved.

Keywords: Data mining; Clustering; Segmentation; Mixed-integer programming

1. Introduction

In recent years, companies have concentrated on understanding the needs and expectations of their customers and grouping the existing and potential customers into classes with the purpose of improving the
efficiency of their marketing strategies and increasing their market share. The abundance of large data collections and the need to extract hidden knowledge within them has triggered the development of algorithms to detect unknown patterns in data sets. Clustering analysis is a data mining technique developed for the purpose of identifying groups of entities that are similar to each other with respect to certain similarity measures.

There is a large number of approaches to the clustering problem, including optimization based methods that involve mathematical programming models for developing efficient and meaningful clustering schemes. Exact and heuristic algorithms for these models have been proposed. However, most of these algorithms suffer from efficiency as the size and the dimension of the data set increases.

In a clustering problem, given a data set with \( n \) data items (instances) in \( m \) dimensions (attributes); the aim is to find an exact partitioning of data items into \( k \) clusters. One of the key points is the definition of the term “similarity”, which helps define the form of the objective function of the optimization model [21]. The objective function of the clustering problem can be defined in several ways such as minimization of the sum of within-cluster distances and minimization of the maximum within-cluster distances within cluster [13]. Fisher [20] modeled the first objective function studied by Rao [13] for the single-dimension case and proposed a least-squares algorithm without a stopping criterion. In later years, the \( K \)-Means algorithm is implemented by applying this criterion as an error function. In the widely-known \( K \)-Means approach, the iterative objective is to minimize the summation of 2-norm distances between each data point and the center of the cluster which it belongs to [15]. Another consideration in clustering is that the resulting clusters are expected to be homogeneous and compact with respect to certain characteristics. In addition, one should decide how the clusters are constructed. The clusters could be ‘exclusive’, ‘overlapping’ or ‘probabilistic’ [10]. In the latter case, a data point belongs to a particular cluster with a certain probability, hence fuzzy clustering is achieved.

In this paper, we propose a mixed-integer programming model to partition the data set into exclusive clusters, where we assume that the number of desired clusters \( k \) is known a priori. The objective function of the model is to minimize the maximum diameter of the generated clusters with the goal of obtaining evenly compact clusters. The original formulation turns out to be computationally demanding. Moreover, there exist alternative optimal solutions since the objective function of the model is insensitive to assignments except for the ones that occur in the “largest” cluster. Hence, we develop a heuristic approach that is based on solving the model with initial seeds in order to improve the solution time, followed by a reassignment heuristic aimed at improving the cluster quality of the model by incorporating sum of within cluster distance averages as a measure.

We used real data from a satellite broadcasting company, Digiturk, in our computational experiments. The company, founded in 1999, is a private digital platform operating in Turkey. The firm has around 800,000 customers and provides five product packages, three pay-per-view services and also various channels, interactive channels and events to its customers. Digiturk is eager to find out the opportunities in customer relationship marketing, such as one-to-one marketing. The company would like to segment its customers based on the transactional factors, such as their package subscriptions, pay-per-view purchases, and interactive event interests. We give an interpretation of the customer segments obtained by our clustering algorithm applied to Digiturk data in Section 6.

This paper is organized as follows. We survey related work on clustering and optimization based clustering methods in Section 2. We present our proposed model in Section 3. The proposed heuristic clustering approach is given in Section 4. In Section 5, we explain the solution of the algorithm on an illustrative example and compare the results with the results of the \( K \)-Means algorithm. Then, in Section 6, we apply the proposed algorithm to the Digiturk problem to analyze its performance and efficiency, and we compare our findings with the solution of the \( K \)-Means algorithm. Finally, the paper is concluded by presenting the results of the study and discussing ideas for future work in the domain of optimization and clustering.
2. Literature review

Data mining (DM) has been an integral part of customer relationship management (CRM) studies, with the premise that companies can achieve successful customer relations if they understand their customers’ characteristics and desires as also pointed out by Nemati and Barko [9]. Rygielski et al. [4] gave an overview of data mining; its applications in industry and the techniques used under this topic, and explained the relation and interaction between DM and CRM applications from various aspects. The role of optimization in DM, has been elaborated by Padmanabhan and Tuzhilin [3] who discussed that data mining and optimization can help each other in the development of new CRM applications, such as maximizing customer lifetime value, customer analysis and customer interactions.

Another important point in clustering is how to initiate the partitioning of the instances. Most commonly, analytical clustering techniques are analyzed in two types: hierarchical and nonhierarchical [2,16,19]. The hierarchical clustering (HC) methods proceed as a series of partitioning operations starting with a single cluster containing all instances and ending when a predefined terminating criterion is achieved. The HC methods do not require the number of clusters to be known at the beginning, which constitutes a robust advantage over nonhierarchical methods. On the other hand, once an instance is assigned to a cluster, the assignment is irrevocable. Therefore, Sharma [18] stated that the HC methods are applied to generate some interpretations over the data set and the solution of a HC may be used as an input for a nonhierarchical method in order to improve the resulting cluster solution.

Nonhierarchical clustering (NHC) methods, also called partitioning clustering, refers to the case where the number of partitions is known a priori [19]. Usually, the data is divided into $k$ clusters initially and the NHC algorithm iterates for all possible movements of data points between the formed clusters until a stopping criterion is met. In these methods, each cluster can be represented by the center of the cluster ($K$-Means) or by one instance located in the cluster center ($K$-Medoids). The NHC algorithms are sensitive to initial partitions and due to this fact, there exist too many local minima [18].

On the other hand, there are also NHC algorithms that build the clusters incrementally, such as the heuristic global $K$-Means method of Likas et al. [1]. They state that the optimal $k$ cluster solution can be reached from the optimal solution of $(k - 1)$ cluster problem by utilizing a local search over all data set by running the $K$-Means algorithm repetitively to come up with the best possible $k$th cluster center. Experimentally, the algorithm is shown to be superior to the classical $K$-Means algorithm. Although the algorithm may lead to promising solutions, it does not guarantee to reach to the global optimal solution.

Perhaps the most widely used clustering algorithm is the $K$-Means algorithm. $K$-Means is reported to work well in practice, although its worst case time complexity is exponential [19]. An important problem faced with $K$-Means algorithm is that the resulting solution is a local minimum which is sensitive to the initially selected cluster centers [17]. Thus, in order to find good minima, repeating the algorithm several times with different starting points is required, without guaranteeing global optimality. There are various studies conducted on the topic of finding good seeds, but none of them promises to find out the initial seeds that will achieve the global optimal solution.

Researchers have been studying the clustering problem with mathematical programming approaches for many years. In an early study, Rao [13] proposed the minimization of the maximum within-cluster distance as an objective function, and he gave a solution approach for the 2-cluster problem. Later on, Brusco [12] studied the same criterion in a branch and bound algorithm and implemented the algorithm by deriving upper bounds with utilizing a randomized complete-link clustering algorithm. Also recently, Bradley et al. [16] investigated integer programming formulations with the objective of minimizing the sum of distances between each data point and its corresponding cluster center, in other words implementation of $K$-Means algorithm within an optimization model. In their studies, they divided the clustering methods into three classes: metric-based methods, model-based methods and partition-based methods and focused on mathematical programming formulations of two popular nonhierarchical clustering techniques, $K$-Means
and \( K \)-Median. Another objective function that has been of interest is the minimization of the sum of within cluster distances. Koontz et al. [21] propose a branch and bound algorithm for an optimization problem with this objective. Diehr [6] studies the algorithm of Koontz et al. [21] for effectiveness and efficiency. Also, Brusco [12] studies the same clustering criterion in a branch and bound algorithm in addition to the proposed branch and bound algorithm with the criterion of the minimization of the maximum diameter. The branch-and-bound methods reported in these studies are not applicable on large data sets due to concerns on computational time. Moreover, the efficiency and performance of the algorithms are reported to be very sensitive to the number of clusters. Still, integer programming models for clustering problem, such as those developed by Vinod [8] and Rao [13], can be preferred as they add more flexibility to the model, and they consume less computing time than previous approaches such as dynamic programming.

In the next section, we present our mathematical programming model for the clustering problem.

3. A mathematical programming model for clustering

We now present a mathematical formulation for the clustering problem with the objective of minimizing the maximum cluster diameter. Similar formulations have been studied by Rao [13] and Brusco [12].

Given a data set of \( n \) data items in \( m \)-dimensions, i.e. a set of \( n \) points in \( \mathbb{R}^m \), the goal of the proposed mathematical model is to find the optimal partitioning of the data set into \( k \) exclusive clusters assuming that the number of desired clusters is known a priori. In the model, the objective function is to minimize the maximum diameter of the generated clusters. The parameter \( d_{ij} \) denotes the distance between two data points \( i \) and \( j \) in \( \mathbb{R}^m \) and can be calculated by any desired norm on \( \mathbb{R}^m \) such as the Euclidean or the Tchebycheff.

The mathematical programming model, \( MIP-Diameter \), is given below.

\[
\begin{align*}
\text{minimize} & \quad Z = D_{\text{max}} \\
\text{subject to} & \quad D_l \geq d_{ij} x_{il} x_{jl} \quad \forall i, j, l, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad l = 1, \ldots, k, \\
& \quad \sum_{l=1}^{k} x_{il} = 1 \quad \forall i, \quad i = 1, \ldots, n, \\
& \quad D_{\text{max}} \geq D_l \quad \forall l, \quad l = 1, \ldots, k, \\
& \quad x_{il} \in \{0, 1\} \quad \forall i, l, \quad i = 1, \ldots, n, \quad l = 1, \ldots, k, \\
& \quad D_l \geq 0 \quad \forall l, \quad l = 1, \ldots, k. 
\end{align*}
\]

In the model, the variable \( D_l \) denotes the diameter of the cluster \( l \) and the variable \( D_{\text{max}} \) denotes the maximum diameter among the generated clusters, which forms the objective function of the model. The variable \( x_{ij} \) is a binary decision variable that represents the assignment of an instance to a particular cluster; it takes a value of 1 if the instance \( i \) is assigned to cluster \( l \), and 0 otherwise. According to constraint (2), the diameter of cluster \( l \) is allowed to be at least the maximum distance between any two data points assigned to that cluster. Since the model aims to partition the data points into exclusive clusters, each instance should be assigned to only one cluster as given in Eq. (3). Constraint (4), in conjunction with the objective function, helps set variable \( D_{\text{max}} \) equal to the value of the maximum diameter. The model has \( kn \) binary variables, \( k + 1 \) continuous variables and \( O(kn^2) \) constraints.

The \( MIP-Diameter \) model is a nonconvex bilinear mixed-integer programming model and as such, it is almost impossible to solve this model in a reasonable amount of time even with small number of instances. Therefore, we linearized \( MIP-Diameter \) constraint (2) without increasing the size of the formulation as:

\[
D_l \geq d_{ij} (x_{il} + x_{jl} - 1) \quad \forall i, j, l, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad l = 1, \ldots, k.
\]
Here if both decision variables $x_i$ and $x_j$ are equal to 1, then the constraint will be active, meaning that data points $i$ and $j$ are assigned to cluster $l$ and the diameter of the cluster has to be at least as long as the distance between them. If one of the decision variables or both of them are equal to zero, the constraint will be redundant.

Although the linear MIP-Diameter model outperforms the nonlinear model in terms of CPU time and number of iterations, the experimental results given in Section 5 show that the computational performance of the model is not comparable with other suggested studies and algorithms in the literature. Moreover, our experiments on synthetic data sets where the global optimal solution is known a priori showed that the objective function of the model should be strengthened. We observed that there are numerous alternative optimal solutions because once the model minimizes the maximum diameter of the clusters generated, the objective function is insensitive to the assignments that take place in other clusters. Considering all these facts, we developed the proposed algorithm given in the following section.

4. An improved algorithm

The idea of fixing the assignment of some instances to certain clusters has been used in clustering algorithms before with the goal of improving computational efficiency. These fixed assignments typically improve the computational performance of the algorithm; however, a new question of how to best determine the instances to be fixed, the seeds, is raised. In particular, we wish to select an initial seed for each cluster in such a way to ensure that the seeds are separated well from each other. To achieve this, we used a graph-theoretic approach that utilizes the concept of a maximal independent set, which has been used in clustering studies [5].

Our seed finding algorithm works as follows. A graph $G_R = (V, E)$ is constructed where the set of vertices $V$ corresponds to instances. An edge between two vertices exists in $E$ if the distance between the two corresponding instances is less than $R$, a distance parameter that is initialized as the average of minimum and maximum of $d_{ij}$ values. Given this graph, an independent set $S$ is a subset of $V$ such that there is no edge among the members of $S$. A maximal independent set is constructed by starting initially with a randomly picked vertex $v$ from the set $V$ and eliminating the vertices adjacent to $v$. This step is repeated until no more vertices can be added to $S$. Then, the size of $S$ is checked, and this procedure is repeated by adjusting the distance parameter $R$ through a line search and by rebuilding the corresponding graph $G_R$ until the maximal independent set $S$ in the current graph consists of $k$ points. Once a maximal independent set of size $k$ is achieved, each vertex in the maximal independent set is assigned to a different cluster for initializing the MIP-Diameter model. Although the procedure is sensitive to the selection of the initial vertex, and different maximal independent sets can be constructed with different initial vertices, the random choice of the initial vertex is sufficient for the purposes of our algorithm.

Once the $k$ seeds to be assigned to the $k$ clusters are determined as described above, we can solve the MIP-Diameter model with these instances fixed. Obviously, this constitutes a heuristic approach to solving the MIP-Diameter model. As we will report later, this heuristic approach is capable of obtaining near optimal solutions for the MIP-Diameter model with much smaller computational effort.

To address the problem of poor quality clusters due to the insensitivity of the MIP-Diameter model to the assignments that take place in the clusters that do not change the objective function, we propose a reassignment procedure. The reassignment procedure relies on improving two new criteria, the average of distance between each data point and other data points assigned to the same cluster and the sum of averages of within-cluster distances.

Let $C_l$ denote the set of all instances assigned to cluster $l$. Then for instance $i$ that belongs to cluster $l$, we compute the quantity
\[ APl_i = \sum_{j \in C_l \atop j \neq i} d_{ij} / |C_l|, \]  
(8)

where \( |C_l| \) denotes the number of elements.

At each step of the reassignment algorithm, we hypothesize that an assignment that yields a smaller \( APl_i \) is better than one with a larger \( APl_i \) since the instances within the clusters, on average, would be closer to each other. Once we check all the possible reassignments for all data points, we hypothesize that the whole reassignment procedure yields a better sum of within-cluster average distances calculated as \( AT \) where

\[ AT = \sum_{l=1}^{k} \left( \frac{\sum_{i \in C_l} \sum_{j \in C_l \atop j \neq i} d_{ij}}{|C_l|(|C_l| - 1)} \right), \]  
(9)

which is the sum of the average of within-cluster distances.

Our reassignment algorithm works as follows: Given the solution of the MIP-Diameter model obtained with fixed seeds, we consider each instance for a possible reassignment different than the one generated by the model. We pick an instance \( i \), and we tentatively consider a reassignment to each cluster \( l \). We compute the improvement, if any, that takes place in \( APl_i \), the average of distance between the point \( i \) and other points in cluster \( l \), if such a reassignment were to be conducted. In addition, we check that such a reassignment would not worsen the objective function of the MIP-Diameter model, \( D_{\text{max}} \), beyond an allowed range described by means of a multiplier \( \beta \) which has to be greater than or equal to 1. In other words, the reassignment procedure may allow for a deterioration of \( D_{\text{max}} \) to \( \beta D_{\text{max}} \) in order to achieve a better \( AT \) value eventually if the parameter \( \beta \) is set equal to a value that is strictly greater than 1. The instance under study is reassigned to a cluster that yields the most improvement in \( APl_i \) without worsening the maximum diameter beyond the allowed range. Once every instance is considered for reassignment, a second pass is conducted in the same way. The reassignment procedure stops when a complete pass with no changes is achieved. Finally, the new \( AT \) value is compared with the initial \( AT \) value that is obtained from the solution of the MIP-Diameter model with fixed seeds and if the new \( AT \) is less than the initial one, the reassignment of the data points is accepted. Otherwise, the solution of the MIP-Diameter model with seeds remains unchanged. We utilize this final control on \( AT \) value in order to eliminate undesired solutions which can be produced by the reassignment algorithm. We now present a formal description of our algorithm.

**Step 1: Seed finding**

**Step 1.0.** Set iteration number \( t = 0 \).

Set lower and upper bounds to \( l(t) = \min\{d_{ij}\}, \ u(t) = \max\{d_{ij}\} \).

**Step 1.1.** Set \( R(t) = \frac{l(t) + u(t)}{2} \).

**Step 1.2.** Construct the graph \( G_{R(t)} \) where instances correspond to vertices, and the edge \( (i, j) \in E \) if \( d_{ij} \leq R(t) \).

**Step 1.3.** Find a maximal independent set \( S \) in \( G_{R(t)} \).

If \( |S| = k \), then go to Step 2.

If \( |S| < k \), then set \( l(t + 1) = l(t), \ u(t + 1) = R(t), \ t = t + 1 \) and go to Step 1.1.

If \( |S| > k \), then set \( l(t + 1) = R(t), \ u(t + 1) = u(t), \ t = t + 1 \) and go to Step 1.1.

**Step 2: Linear MIP-Diameter model with fixed seeds**

Let \( S = \{v_1, v_2, \ldots, v_k\} \). Without loss of generality, set \( x_{ii} = 1 \) for each \( v_i \in S \). Solve the MIP-Diameter model with these assignments fixed. For \( i = 1 \ldots n \), define \( C(i) = l \) for \( l \) such that \( x_{il}^o = 1 \) where \( x_{il}^o \) denotes the optimal value of the assignment variable \( x_{il} \) according to the solution of MIP-Diameter model.
Step 3: Reassignment

Step 3.0. Set \( x^* = x^0 \), where \( x^0 \) is the solution obtained in Step 2.

For \( l = 1, \ldots, k \), let \( C_l(x^*) \) denote the instances assigned to cluster \( l \) in assignment matrix \( x^* \). Let \( AP(x^*) = \sum_{i,j \in C_l(x^*)} d_{ij} / |C_l(x^*)| \) denote the average distance between instance \( i \) and other data points in cluster \( l \), \( D_l(x^*) = \max_{i \in C_l(x^*)} \{ d_{ij} \} \) denote the diameter of cluster \( l \) and \( Initial_{AT}(x^*) = \sum_{i=1}^{k} \left( \sum_{j \in C_l(x^*)} \sum_{j \neq i} d_{ij} \right) / (|C_l(x^*)|( |C_l(x^*)| - 1 )) \) denote the sum of within cluster averages in solution \( x^* \).

Set \( R = \max_{l=1, \ldots, k} \{ D_l(x^*) \} \) and set \( \beta \geq 1 \).

Step 3.1. Set \( i = 1 \). Set \( update = 0 \).

Step 3.2. Set \( l = 1 \).

Step 3.3. If \( x^*_{il} = 1 \) and \( l < k \), set \( l = l + 1 \) and go to Step 3.4.

If \( x^*_{il} = 1 \) and \( l = k \), set \( i = i + 1 \) and go to Step 3.5.

Step 3.4. Let \( x_C^l \) denote the solution in which instance \( i \) is moved from cluster \( C(i) \) to cluster \( l \).

If \( AP(x_C^l) \leq AP(x^*) \) and \( D_{\text{max}}(x_C^l) \leq \beta R \), then set \( C(i) = l \) and \( x^* = x_C^l \), \( update = update + 1 \).

If \( l < k \), set \( l = l + 1 \) and go to Step 3.3. Otherwise go to Step 3.5.

Step 3.5. If \( i < n \), set \( i = i + 1 \) and go to Step 3.2.

If \( i = n \) and \( update = 0 \), set \( Final_{AT}(x^*) = \sum_{l=1}^{k} \left( \sum_{i \in C_l} \sum_{j \in C_l} d_{ij} \right) / (|C_l(x^*)|( |C_l(x^*)| - 1 )) \).

Go to Step 3.6.

If \( i = n \) and \( update > 0 \), go to Step 3.1.

Step 3.6. If \( Final_{AT}(x^*) < Initial_{AT}(x^*) \), then STOP.

If \( Initial_{AT}(x^*) < Final_{AT}(x^*) \), then set \( x^* = x^0 \), i.e. revert to the clustering solution at the end of Step 2.

5. Illustrative example

In this part of the study, we applied the proposed algorithm on a set of 81 data points given in Fig. 1. For the purpose of illustration, data points are represented in a 2-dimensional space and the distance between any two points is calculated by the Euclidean distance measure.

We performed experiments on this synthetic data set which consists of four distinct clusters to show how MIP-Diameter model may fail to reach an acceptable solution, to evaluate the accuracy and the performance of the proposed algorithm and to compare the results of our algorithm with the results of the well-known K-Means algorithm.

For the data set given in Fig. 1, the minimum value of the maximum diameter of the generated clusters is 8 since the horizontal and vertical distance between any two neighbor data points is equal to 1. We obtain the optimal solution to the linearized MIP-Diameter model using the CPLEX 8.1 solver in almost three hours of CPU time on a PC with Pentium IV 3.06 GHz processor, 512 MB memory although the data set is fairly small. There are 524,954 branch & bound nodes enumerated with respect to the 330 variables and 26,331 constraints in this illustrative example. These computational findings indicate the difficulty of solving the proposed MIP-Diameter model exactly. Furthermore, in Fig. 2, we see that in the optimal solution there are 10 data points that could possibly be assigned better.

Next, we apply the proposed algorithm on the illustrative data set. Initially, we start by the seed finding algorithm to determine 4 seeds and then solve the MIP-Diameter model with these determined seeds. We
see that the MIP-Diameter solved with seeds also reaches the optimal solution as maximum diameter value is again found to be 8. However, by applying the seed finding algorithm the running time of the model is decreased from almost three hours to only 5.81 seconds while preserving the optimality of the resulting solution.

As expected, when we look at Fig. 3, it is seen that there are still assignments of data points that seem to be open to improvement. We then apply the reassignment algorithm with \( \beta = 1 \). The solution of the proposed algorithm is given below in Fig. 4.

We also compare the solution of our algorithm with the solution of the well-known K-Means algorithm according to the measure \( AT \), the sum of average distances of the generated clusters. In Fig. 5, the
assignment pattern of the $K$-Means algorithm is illustrated and in Table 1 the numerical values for the exact solution, $MIP$-$Diameter$ model’s solution, $K$-Means algorithm’s solution and proposed algorithm’s solution are given. Based on the values given in Table 1, the proposed algorithm solution is only 0.2% worse than the optimal solution of the $MIP$-$Diameter$ model with respect to the sum of averages of within-cluster distances.

6. Evaluation of proposed models on a real data set

In this part of the study, the performance and accuracy of the proposed mathematical programming model and the proposed clustering algorithm are examined on a real data set. Some interpretations derived
from the data set are given and the solution of the proposed algorithm is compared with the solution of the
K-Means algorithm with respect to discussed clustering indicators and interpretability of the formed clus-
ters. In later parts of this section, we report results of various computational experiments carried on data
sets drawn from the Digiturk database.

6.1. Experiments with the MIP-Diameter model

On the real data set consisting of 18 attributes, several sets of experiments are performed in order to ana-
lyze the performance and the accuracy of the proposed MIP-Diameter model.

First we performed a set of experiments on the linearized MIP-Diameter model without seeds to com-
pare its results with the one initiated with seeds. One can observe the positive effect of the seed finding algo-
rithm on the solution of MIP-Diameter model, in terms of improvement in the number of iterations, the
number of nodes and the CPU time required to partition the data set into 2 clusters by comparing the
results given in Table 2.
In Table 2, \(k\) is the number of clusters, \(N\) is the number of instances, Var. is the number of variables and Cons. is the number of constraints in the model, Iter. is the number of iterations the model performs, Solv. (MB) is the required memory size, \(t\) (seconds) denotes CPU time, and \(D_{\text{max}}\) is the objective function value of the MIP-Diameter model.

When we analyze the above table, we see that CPU times, number of nodes and iterations decrease significantly while the optimal solution of the model is preserved. We performed several experiments with varying size of data sets and number of clusters and from all those experiments the positive effect of the seed finding algorithm is approved. To partition the same data set with 100 instances into four clusters without seeds takes more than 2 weeks of time and by applying the seed finding algorithm we partition the same data set into four clusters only in 12.5 seconds.

### 6.2. Experiments with the proposed algorithm

In Tables 3–5, we give our experimental findings with the proposed algorithm with the goal of illustrating the benefit of the reassignment procedure. For this part of the study, data sets with 100, 200 and 300

<table>
<thead>
<tr>
<th>Experiment number</th>
<th>(k)</th>
<th>Var.</th>
<th>Cons.</th>
<th>Without seeds (D_{\text{max}})</th>
<th>Iter.</th>
<th>t (seconds)</th>
<th>Node</th>
<th>Solv. (MB)</th>
<th>With seeds (D_{\text{max}})</th>
<th>Iter.</th>
<th>t (seconds)</th>
<th>Node</th>
<th>Solv. (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>100</td>
<td>203</td>
<td>20102</td>
<td>61.1</td>
<td>2167</td>
<td>22</td>
<td>16.5</td>
<td>9.2</td>
<td>61.1</td>
<td>322</td>
<td>0</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>200</td>
<td>403</td>
<td>80202</td>
<td>68.1</td>
<td>9331</td>
<td>36</td>
<td>215.5</td>
<td>35.4</td>
<td>68.1</td>
<td>982</td>
<td>1</td>
<td>14.9</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>300</td>
<td>603</td>
<td>180302</td>
<td>70.6</td>
<td>12384</td>
<td>13</td>
<td>2166</td>
<td>79</td>
<td>70.6</td>
<td>1036</td>
<td>0</td>
<td>46.5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>400</td>
<td>803</td>
<td>320402</td>
<td>77.9</td>
<td>14613</td>
<td>9</td>
<td>4650.8</td>
<td>140</td>
<td>77.9</td>
<td>1170</td>
<td>0</td>
<td>91.1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>500</td>
<td>1003</td>
<td>500502</td>
<td>77.9</td>
<td>29551</td>
<td>12</td>
<td>7266.4</td>
<td>218</td>
<td>77.9</td>
<td>1411</td>
<td>0</td>
<td>213.3</td>
</tr>
</tbody>
</table>

In Table 2, \(k\) is the number of clusters, \(N\) is the number of instances, Var. is the number of variables and Cons. is the number of constraints in the model, Iter. is the number of iterations the model performs, Solv. (MB) is the required memory size, \(t\) (seconds) denotes CPU time, and \(D_{\text{max}}\) is the objective function value of the MIP-Diameter model.

<table>
<thead>
<tr>
<th>Experiment number</th>
<th>(k)</th>
<th>Var.</th>
<th>Cons.</th>
<th>MIP-Diameter with seeds</th>
<th>Iter.</th>
<th>(D_{\text{max}})</th>
<th>AT (t) (seconds)</th>
<th>Node</th>
<th>Solv. (MB)</th>
<th>After reassignment</th>
<th>RIter.</th>
<th>(D_{\text{max}})</th>
<th>AT (t) (seconds)</th>
<th>Total (t) (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>203</td>
<td>20102</td>
<td>322</td>
<td>61.1</td>
<td>46</td>
<td>1.5</td>
<td>0</td>
<td>9.2</td>
<td>1</td>
<td>61</td>
<td>46</td>
<td>0.6</td>
<td>2.2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>304</td>
<td>30103</td>
<td>962</td>
<td>52</td>
<td>83</td>
<td>5.5</td>
<td>26</td>
<td>13.6</td>
<td>1</td>
<td>52</td>
<td>54</td>
<td>0.8</td>
<td>8.3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>405</td>
<td>40108</td>
<td>1051</td>
<td>83</td>
<td>12.5</td>
<td>18</td>
<td>35</td>
<td>18</td>
<td>2</td>
<td>50</td>
<td>67</td>
<td>1.8</td>
<td>14.3</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>506</td>
<td>50110</td>
<td>3495</td>
<td>12.5</td>
<td>99</td>
<td>37.5</td>
<td>313</td>
<td>22.4</td>
<td>3</td>
<td>44</td>
<td>81</td>
<td>2.8</td>
<td>40.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experiment number</th>
<th>(k)</th>
<th>Var.</th>
<th>Cons.</th>
<th>MIP-Diameter with seeds</th>
<th>Iter.</th>
<th>(D_{\text{max}})</th>
<th>AT (t) (seconds)</th>
<th>Node</th>
<th>Solv. (MB)</th>
<th>After reassignment</th>
<th>RIter.</th>
<th>(D_{\text{max}})</th>
<th>AT (t) (seconds)</th>
<th>Total (t) (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>403</td>
<td>80102</td>
<td>982</td>
<td>68</td>
<td>45</td>
<td>14.9</td>
<td>1</td>
<td>35.4</td>
<td>1</td>
<td>68</td>
<td>45</td>
<td>4.5</td>
<td>19.4</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>604</td>
<td>120206</td>
<td>5659</td>
<td>77</td>
<td>147</td>
<td>186</td>
<td>186</td>
<td>52.9</td>
<td>2</td>
<td>57</td>
<td>57</td>
<td>9.6</td>
<td>156.5</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>805</td>
<td>160208</td>
<td>12302</td>
<td>52</td>
<td>365</td>
<td>1396</td>
<td>1396</td>
<td>70.3</td>
<td>3</td>
<td>52</td>
<td>66</td>
<td>16.4</td>
<td>381</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>1006</td>
<td>200210</td>
<td>13069</td>
<td>46</td>
<td>467</td>
<td>1054</td>
<td>1054</td>
<td>87.8</td>
<td>2</td>
<td>44</td>
<td>78</td>
<td>16.9</td>
<td>484.2</td>
</tr>
</tbody>
</table>
Instances are partitioned into two to five clusters. In addition to the explanations in Section 6.1, the term $AT$ in below tables is the sum of averages of within-cluster distances.

In all of these experiments, the coefficient $b$ is taken as 1. We note that the sum of average within cluster distances almost always improves with reassignment and the improvement is more significant as the number of clusters increases.

6.3. The effect of parameter $\beta$

In order to analyze the effect of the coefficient $\beta$ on the reassignment algorithm, we experiment on the 100 instance data set with its partitioning into 5 cluster solution. The data points marked with diamonds represent the value of the maximum of the diameters and the data points marked with squares represent the value of the sum of the averages of distances within clusters.

In Fig. 6, the values for both the maximum diameter and the sum of averages are reported as a function of $\beta$. We observe that with $\beta = 1$, we are able to obtain a sum of averages of 81. This figure is 18% better than the sum of averages measure before reassignment, which happens to be 98.8. We note that a 12% increase in the maximum diameter may lead to a 34% improvement in the value of sum of averages measure by letting $\beta$ take a value of 1.2.

6.4. Interpretation of the results

In this part of the study, we compare the results of the proposed algorithm with the results of the $K$-Means algorithm. For this purpose, we used 300 instances dataset clustered into four classes by two methods.

![Image](image_url)
We observe that the clusters formed by our proposed algorithm are more interpretable than the clusters formed by the \textit{K}-Means algorithm. According to the results of our algorithm, we were able to identify two clusters of active users, one cluster of inactive users and one cluster of suspended users. Furthermore, it is possible to differentiate the two active user clusters from each other by the attributes age, pay-per-view services and membership durations. Statistical tests indicate that the differentiation in attribute values is significant. When we look at the results of the \textit{K}-Means algorithm, we can not easily interpret the clusters obtained. Moreover, the test statistics calculated for the clusters of our algorithm are more promising for the ones calculated for the \textit{K}-Means clusters. Based on our intuitive observations, our algorithm leads to a better solution than the \textit{K}-Means algorithm. However, we note that \textit{K}-Means might deliver different clusters when different initial points are used and it may be possible to obtain interpretable clusters with \textit{K}-Means as well. Still, the fact that our algorithm does not require trial and error experimentation constitutes an advantage.

7. Conclusions

In this paper, we presented one mathematical programming based segmentation model and a heuristic clustering algorithm that are applied to a digital platform company’s customer database. The \textit{MIP-Diameter} model forms clusters by minimizing the maximum diameter of the generated clusters. The model is nonhierarchical in the sense that the number of the clusters is assumed to be known a priori. The accuracy of the proposed algorithm is compared with the results of the \textit{K}-Means algorithm on an illustrative example and real data set and it is shown that the proposed algorithm performs better than the \textit{K}-Means algorithm.

The run time of the proposed \textit{MIP-Diameter} model is improved drastically with linearization and the proposed seed finding algorithm. In addition, the reassignment of the instances according to the measure sum of averages of within-cluster distances is shown to lead to better solutions. Although the \textit{MIP-Diameter} model has been formulated before, our proposed approach constitutes a viable solution to the clustering problem with the help of the seed finding and reassignment ideas.

Topics that remain to be explored in the future include improving the reassignment algorithm and the solution of the \textit{MIP-Diameter} model via generation of valid inequalities and branching rules. In addition, addressing the problem of minimizing sum of average within cluster distances directly via efficient formulations constitutes a future research problem. Finally, the study of the two alternative models within a combined framework could lead to interesting findings.

Acknowledgements

We wish to thank Salih Eren, Devrim Melek Tunç and Kemal Özden at the Information Technologies Department of Digiturk for providing the data for the experiments. We also appreciate the anonymous referees for their many helpful suggestions.

References


