A Hierarchical Approach to Represent Relational Data Applied to Clustering Tasks


Abstract— Nowadays, the representation of many real word problems needs to use some type of relational model. As a consequence, information used by a wide range of systems has been stored in multi relational tables. However, from a data mining point of view, it has been a problem, since most of the traditional data mining algorithms have not been originally proposed to handle this type of data without discarding relationship information. Aiming to ameliorate this problem, we propose a hierarchical approach for handling relational data. In this approach the relational data is converted into a hierarchical structure (the main table as the root and the relations as the nodes). This hierarchical way to represent relational data can be used either for classification or clustering purposes. In this paper, we will use it in clustering algorithms. In order to do so, we propose a hierarchical distance metric to compute the similarity between the tables. In the empirical analysis, we will apply the proposed approach in two well-known clustering algorithms (k-means and agglomerative hierarchical). Finally, this paper also compares the effectiveness of our approach with one existing relational approach.

I. INTRODUCTION

The relational databases are powerful because they require few assumptions about how data is related or how it will be extracted from the database. As a result, the same database can be viewed in many different ways [21]. These characteristics have made it a popular choice of data storage. As a consequence, different types of systems have used relational database. In a relational database, all data is stored in tables. They have the same structure repeated in each row (like a spreadsheet), and also have links (foreign keys) that are used to perform the relation between them [19].

The data representation of a relational model differs from the traditional feature-vector (single table or spreadsheet) representation used in traditional data mining tasks. A relational model is more expressive than attribute value model in capturing and describing complex structures and relationships between multiple tables [1], [2]. Furthermore, in a dataset stored in a relational database with one-to-many associations among records, each table record can form many associations with others records from other tables, making the processing of these data a difficult task.

As a consequence of the widespread use of relational databases, the use of data mining methods to discover knowledge in this type of database has become an interesting issue. However, conventional data clustering algorithms, for instance, identify similar instances of a dataset based on the similarity of their attribute values. In order to do this, the attributes need to be placed on a single table or worksheet format file. Relatively few researches have addressed the particularities of the relational data to discover unknown patterns within a relational database [3], [4].

In this paper we propose a hierarchical approach for handling relational data. The data is processed in a way that the relations (tables) of the database can be placed in hierarchical structure. The outcome of this processing is a number of instances (records) that have different lengths according to their position in the hierarchy. This paper will also evaluate the effectiveness of the proposed approach in two well-known clustering algorithms (Agglomerative hierarchical and k-means). This analysis aims to evaluate the performance of three different ways to handle relational data, including our approach. Therefore, we propose three different scenarios. In the first scenario (called baseline scenario or, simply, scenario 1), we retrieve all the data from the database and placed it in single table.

For this scenario, only the values of the attributes were used as input for the distance metric of the clustering algorithm. No relationship attributes were used in this scenario.

In the second scenario (called hierarchical scenario or, simply, scenario 2), we place the instances (records) in a hierarchical structure. In this case, one of the attributes stores the hierarchy of the data (the main table is the root and the relations are the nodes). For this scenario, we propose a hierarchical distance metric to be used specifically on the attribute that stores the hierarchy. In both scenarios, we use two well-known type of clustering algorithms (Agglomerative hierarchical and k-means). Nevertheless, any clustering algorithm could be used for the aforementioned scenarios. Diversity was the only criterion used to choose these two algorithms since the first one is deterministic and the other one is probabilistic.

In the third scenario, we use TSMRC, which is a method for clustering relational data proposed in [6], [7]. According to the authors, this relational clustering method called TSMRC uses two stages in order to improve the efficiency of clustering. In the first stage, each object type (table) is clustered individually, considering attribute and intra-relationship information. In the second stage, each cluster obtained as a result of the first stage is merged according to

---

Manuscript received February 10, 2011. This work has the financial support of CAPES and CNPq (Brazilian Research Councils), under processes numbers BEX 2481/09-0, 550810/2007-2 and 140239/2011-1.

João C Xavier-Junior is with Computing and Automation Engineering Department – Federal University of Rio Grande do Norte (UFRN), Natal, RN - BRAZIL, 59078-900, lxavier01@gmail.com

Anne M P Canuto is with Informatics and Applied Mathematics Department – Federal University of Rio Grande do Norte (UFRN), Natal, RN - BRAZIL, 59072-970, amcp@dimap.ufrn.br

Alex A Freitas is with School of Computing – University of Kent, Canterbury, Kent, UK, CT2 7NF, Tel: +441227-827220, A.A.Freitas@kent.ac.uk

Luiz M. G. Gonçalves is with Computing and Automation Engineering Department – Federal University of Rio Grande do Norte (UFRN), Natal, RN - BRAZIL, 59078-900 Tel.: +558432153771 lmarcos@dcia.ufrn.br

Carlos N. Silla Jr. is with School of Computing – University of Kent, Canterbury, Kent, UK, CT2 7NF, Tel: +441227-827220, cn2@kent.ac.uk
II. RELATED WORKS

The main aim of clustering algorithms is to use attribute information to group instances (records) that have similar attribute values. However, when working with relational data there are more types of information available that need to be used to distinguish groupings [3]. Clustering in relational data has been studied in some works [1], [2], [3], [4], [5], [6], [7], [11]. Neville et al [3], for instance, adapted graph-cutting algorithms to cluster only link attributes (relationships), only attributes, or both by using a hybrid approach (graph-partitioning algorithm with an attribute similarity metric). Yin et al [4] proposed a methodology called CrossClus (cross-relational clustering with user’s guidance) for choosing descriptive, cross-relational features in a data set to produce a single object type (table) that is a composed of features from other tables. When the feature selection algorithm halts, the CLARANS algorithm [8] is used to cluster the set of compound object types. This algorithm is relational in the sense that it takes relational data as input. Finally, Bhattacharya and Getoor [5] proposed a relational clustering method applied for multi-type entity resolution. This method involves a metric calculated as the linear combination of graph similarity and attribute similarity between two references.

Other important aspect which needs to be considered, when clustering relational data, is the similarity between two instances. In [6] and [7], the authors proposed a two-stage clustering method for multi-type relational data, called TSMRCC. To improve clustering quality, the authors proposed different similarity measures for both stages. In TSMRCC, only attributes values were considered when clustering the tables separately (first stage), and all relationships were considered during the second stage [6], [7]. The authors state that the method improves clustering efficiency and accuracy. However, it is not clear whether this method can cope with a considerably large number of tables and the consuming time for clustering them in two stages.

In another work, Long et al [11] proposed a probabilistic model for relational clustering, which also provides a principal framework to unify various important clustering tasks including traditional attributes-based clustering, semi-supervised clustering, co-clustering and graph clustering. The proposed model seeks to identify cluster structures for each type of data object types (tables) and interaction patterns between different object types.

Finally, in Alfred and Kazakov [1], [2], they proposed a modeling approach that is capable of learning classification-clustering hybrid model from a relational database directly. In this modeling approach, they presented a way of generalizing or mapping data with one-to-many relationship in learning from relational domain. The authors use DARA (Dynamic Aggregation of Relational Attributes) to convert the data from a relational model into a vector space model. DARA processes the relation data from tables converting it into tuples of binary values. It is important to emphasize that relational data, in most of the cases, have no class labels which makes classification tasks very unlikely.

Unlike the aforementioned works, this work uses a hierarchical approach for handling the properties of the relational data. In this approach, we maintain the attributes values of each table, but we structure them in a hierarchical model. By doing this, we are able to measure the distance between two instances of the dataset as the instances have different values according to the hierarchy. Different distance metrics are applied for normal attributes (categorical or numeric) and hierarchical ones in order to improve the effectiveness of the clustering algorithm.

III. PROBLEM FORMULATION

In relational data clustering problem, there are some collections of multi-type instances and their relationships. We represent them as a set of instances $X$ and a set of relationships $R=\{R_{\text{intra}}, R_{\text{inter}}\}$. $X$ can include $m$ different object types (tables), which are $X_1=\{x_{11}, x_{12}, \ldots, x_{1n}\}, \ldots, X_m=\{x_{m1}, x_{m2}, \ldots, x_{mn}\}$. Each table $X_i$ has its own attributes $X_i.A$. There are two different types of $R$: $R_{\text{intra}}$ and $R_{\text{inter}}$. $R_{\text{intra}}$ refers to intra-relationships in each table, and $R_{\text{inter}}$ refers to inter-relationships between different tables.

Some relational data do not have intra-relationships, when this is the case, only inter relationships are taken into account. This is the case of the datasets used in this paper. Furthermore, the inter relationships (foreign keys) are one-to-many. Given $X$ and $R$, one can cluster $X$ into $k$ clusters and this process is called relational data clustering.

Table I shows the relational tables and attributes of a dataset with five different tables ($m=5$). In this example, the table main, as implied in the name, is the main table of this dataset. The other four tables are considered secondary tables and they have inter-relationship links (foreign key) with table main. This relation is made through information type, which is a categorical, having values fauna, flora, water and sediment. It can be noticed from Table I that the secondary tables have their own attributes. In the empirical analysis, we will use different versions of this dataset.

| TABLE I. THE RELATIONAL TABLES AND THEIR ATTRIBUTES, CATEGORICAL (C) OR NUMERIC (N). |
|-----------------------------------------------|-----------------------------------------------|-----------------------------------------------|
| **Table main Attributes**                     | **Table fauna Attributes**                    | **Table flora Attributes**                    |
| userType (c)                                   | faunaName (c)                                 | floraName (c)                                 |
| informationType (c)                           | faunaType (c)                                 | floraType (c)                                 |
|                                             | faunaSpecie (c)                               | floraSpecie (c)                               |
|                                             | faunaPresence (c)                             | floraPresence (c)                             |
| **Table sediment Attributes**                 |                                              |                                              |
| sedimentName (c)                              |                                              |                                              |
|                                             | sedimentPresence (c)                          |                                              |
| **Table water Attributes**                    |                                              |                                              |
| waterFeature (c)                              |                                              |                                              |
|                                             | Value (c)                                     |                                              |

IV. CLUSTERING ALGORITHM

In this paper, we use two well-known clustering algorithms. The first one is agglomerative hierarchical clustering algorithm, which generates clusters by iteratively merging similar instances into larger and larger clusters. Initially it uses each instance as a cluster, and computes the similarity between each pair of instances. Then it keeps merging the most similar pairs of clusters, until no clusters
are similar enough or a certain number of clusters remain [10].

The second one is k-means, which is one of the simplest unsupervised learning algorithms that solve clustering problems. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster [23].

A. Cluster Validity Measures

For measuring the results of a clustering algorithm, some validity indices have been proposed in the field of data mining [10]. These indices are used to measure the “goodness” of a clustering result comparing it to other results obtained by other clustering algorithms, or the same algorithm but varying different parameters.

In this paper, we use two internal validity indexes for measuring the clustering results, since our dataset has no class information (class label). The Davies-Bouldin (DB) index [12] and the Silhouette index [27] were chosen in this case. We now present formal definitions of these two quality indexes of clusters.

Let \( X_i = \{X_{i_1},...,X_{i_N}\} \) be the data set and let \( C = (C_1,...,C_k) \) be its clustering into \( K \) clusters. Let \( d(X_i,X_j) \) be the distance between \( X_i \) and \( X_j \). Let \( C_j = \{X_{j_1},...,X_{j_{m_j}}\} \) be the \( j \)-th cluster, \( j=1,...,K \), where \( m_j = |C_j| \). The average distance \( a_i \) between the \( i \)-th vector in the cluster \( C_i \) and the other vectors in the same cluster is given by the following expression [26], [27], [28]:

\[
a_i = \frac{1}{m_j-1} \sum_{k \neq i}^m d(X_i, X_k), \quad i = 1, ..., m_j \tag{1}
\]

The minimum average distance between the \( i \)-th vector in the cluster \( C_j \) and all the vectors clustered in the clusters \( C_k, k = 1, ..., K, k \neq j \) is given by the following expression:

\[
b_i = \min_{n=1, ..., K} \left\{ \frac{1}{m_n} \sum_{k=1}^m d(X_i, X_k) \right\}, \quad i = 1, ..., m_j \tag{2}
\]

Then the silhouette width of the \( i \)-th vector in the cluster \( C_j \) is defined in the following way:

\[
s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{3}
\]

From the equation (3), it follows that \(-1 \leq s_i \leq 1\). We can now define the silhouette of the cluster \( C_j \):

\[
S_j = \frac{1}{m_j} \sum_{i=1}^{m_j} s_i \tag{4}
\]

Finally, the global Silhouette index of the clustering is given by:

\[
S = \frac{1}{K} \sum_{j=1}^{K} s_j \tag{5}
\]

It is important to emphasize that both silhouette of the clusters and the global silhouette take values between -1 and 1. In this sense, the bigger the global silhouette index value, the better the clustering partition.

Let \( X_r = \{X_{r_1},...,X_{r_N}\} \) be the data set and let \( C = (C_1,...,C_k) \) be its clustering into \( K \) clusters. Let \( d(X_i,X_j) \) be the distance between \( X_i \) and \( X_j \). Then the Davies-Bouldin index is defined in the following way [12]:

\[
DB(C) = \frac{1}{k} \max_{i \neq j} \left\{ \frac{\Delta(C_i) + \Delta(C_j)}{\delta(C_i,C_j)} \right\} \tag{6}
\]

where \( \Delta(C_i) \) is the intra-cluster distance and \( \delta(C_i,C_j) \) is the inter-cluster distance. The centroid diameter is used for \( \Delta(C_i) \). It is defined in the following way:

\[
\Delta(C_j) = 2 \left( \frac{\sum_{X_k \in C_j} d(X_k, sC_j)}{|C_j|} \right), \quad i = 1, ..., K \tag{7}
\]

where \( sC_j = \frac{1}{|C_j|} \sum_{X_k \in C_j} X_k \).

The centroid linkage inter-cluster distance is used for \( \delta(C_i,C_j) \). It is defined in the following way:

\[
\delta(C_i,C_j) = d(sC_i, sC_j), \tag{8}
\]

where \( sC_i = \frac{1}{|C_i|} \sum_{X_k \in C_i} X_k \) and \( sC_j = \frac{1}{|C_j|} \sum_{X_k \in C_j} X_k \).

The Davies-Bouldin index measures the average similarity between each cluster and its most similar one. As the clusters need to be compact and separated, the smaller the DB index value, the better the clustering partition. Note that, as the NatalGIS dataset used in this work has no classes, no external index was used.

V. SIMILARITY MEASURES

Measuring the distance between numeric values of attributes belonging to two different instances in the same table is straightforward. Generally, the distance between two records is considered in the Euclidean space \( R^2 \).

The problem of measuring similarity becomes more complicated with categorical attributes, because there is no inherent distance measure between them [13], [15], [18]. Various similarity measures for categorical attributes have been proposed in literature. However, in general, they use the similarity concept of dichotomy, because of its simplicity and convenience for use. They adopted similarity measures like Simple Matching Coefficient and Jaccard Coefficient [14].

When dealing with relational data, in many cases, the data is composed of numeric and categorical attributes. In this sense, similarity measures need to cope with both types of attributes. Some studies have proposed similarities measures for both numeric and categorical attributes [6], [7], [16]. In [6], for instance, they define their similarity measure as follows:

\[
Sim_A(x_{ij}, x_{ik}) = \sum_{p=1}^{P} |x_{ij}^p - x_{ik}^p| + \sum_{r=p+1}^{N} \delta(x_{ij}^r, x_{ik}^r) \tag{9}
\]
Simₐ is the sum of differences of standardized numeric and categorical attributes and xᵢⱼ and xᵢⱼ are two instances in Xᵢ. In this equation, xᵢⱼ, xᵢⱼ is the r-th attribute of xᵢⱼ, xᵢⱼ, and N is the number of attributes, p is number of numeric attributes and (N-p) is the number of categorical attributes. The difference function possible results for δ(a,b) are 0 or 1. If a and b are equal, the result is 0, otherwise, 1.

The similarity measure described in Eq.9 will be used in three scenarios as base to a distance metric for both agglomerative hierarchical and k-means clustering algorithms.

A. The Adjusted Similarity Procedure

As already mentioned, we store relational data in multiple tables and each one of these tables has a different number of attributes. Due to this fact, measuring the similarity between two instances from different tables becomes a difficult task. In this sense, it is necessary to address the problem in a more specific way.

In order to cope with the mentioned relational data particularity, the standard similarity procedure needs to be adjusted. In scenario 3, we use the method proposed by Gao et al [6]. For scenarios 1 and 2, we have adjusted the similarity procedure and they will be explained in the next sections (in the description of the scenarios).

VI. OUR APPROACH

In this section, we describe the proposed approach to handle relational data. It is clear that many of the relations between the different tables have the “is-a” property. For example, Octopus is one of the specializations of Cephalopods which is one of the specializations of Mollusk that is one of the specializations of Fauna. Fauna is one of the types of InformationType attribute. It is possible, therefore, to transform the relational attributes into a hierarchy. This transformation can be described as follows.
1. Get the main table of the relational dataset;
2. Create a tree in which the root can be represented by its name (main table);
3. The first level of this tree will be composed by the attributes of the main table;
a. The attributes with no “is-a” are considered seen leaf nodes and no further action will be taken;
4. Go through all unseen leaf nodes, verifying the existence of “is-a” relationships;
a. For all attributes with “is-a” relation, create the following level of the tree and allocate all the attributes of the linked table as unseen nodes of this level;
5. Finish when all nodes are defined as seen leaf nodes.

As it can be seen, it is a simple procedure in which the inclusion of one level in the hierarchical data is defined by an “is-a” relation. The representation of hierarchical attribute follows the standard representation of hierarchical attribute values, in which R represents the root and values of different levels are separated by point. The first two levels of the hierarchical model in Figure 1 represent the transformation of the dataset represented in Table I. The first level of Figure 1 represents the attributes of the main table. As InformationType attribute has “is-a” relation with secondary tables, a second level of the hierarchical model was created and has four nodes (flora, fauna, sediment and water). In fact, Table I represents a simplification of the relational dataset used to create Figure 1.

In the complete relational dataset, all four secondary tables (flora, fauna, sediment and water) have attributes with “is-a” relations with other tables. In the fauna table, for instance, attributes can have up to five levels (four levels of Figure 1 plus the values associated to these attributes). In the empirical analysis of this paper, we will use different versions of the dataset represented in Figure 1.

VII. EXPERIMENTAL SETTING UP

For experimental purposes, as already mentioned, we proposed three different scenarios and they represent three different ways to handle hierarchical data. In this section, we describe in details these different scenarios. However, first of all, we describe the datasets used in this empirical analysis.

A. Datasets

In this analysis, we use four different versions of the NatalGIS dataset [9]. This dataset is used to store the accesses (logs) of different users within a Geographic Information System (GIS). The NatalGIS provides geographic information only for a group of registered users. This system is responsible for environmental management of an area of coral reefs (9 kilometers long by 3 kilometers wide) located in the state of Rio Grande do Norte, Brazil [9]. In Table II, we described the relational structure of the NatalGIS dataset.

The different versions of the NatalGIS were defined by the number of instances as well as the proportion of instances with links to other tables (attributes which are foreign key in other tables). Table II shows a description of these different versions (called Set1, Set2, Set3 and Set4). The first column represents the total number of instances and the following columns represent the number of instances with relation to four secondary tables (Fauna, Flora, water and sediment). For instance, in set1, the main table has 300 instances and 100 instances are linked to table Fauna.

For Fauna and Flora tables, as they have further links, the corresponding columns have number within brackets and they represent the proportion of instances of the corresponding table with links to other tables. We created these four different datasets aiming to investigate the performance of the scenarios in handling data with a varied proportion of relational data.

<table>
<thead>
<tr>
<th>TABLE II. NUMBER OF INSTANCES OF DIFFERENT SETS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total of instances</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>Set 1</td>
</tr>
<tr>
<td>Set 2</td>
</tr>
<tr>
<td>Set 3</td>
</tr>
<tr>
<td>Set 4</td>
</tr>
</tbody>
</table>
As already mentioned, all three scenarios use the similarity measure defined in Eq.(9). For attributes latitude and longitude, we compute the distance between two instances by using the Pythagorean Theorem [17]. After applying the Pythagorean Equation, we normalize the result which will always range between \([0..1]\).

B. Establishing a baseline approach (Scenario 1)

The main purpose of having scenario 1 (very common in literature) is to establish a baseline approach necessary for comparison. For this scenario, a single table containing all the attributes of all tables was created. In this global table, the instances that do not have links to the tables represented by one attribute, its value will be "N/A" (non-applicable). In this case, when an instance in the main table has a link to table Water, for instance, this instance will have values in the attributes that represent tables main and water. The attributes which represent tables fauna, flora and sediment will have "N/A".

Table III presents names, types and source table for all attributes placed in the global table used as dataset for scenario 1. In addition, Figure 2 represents a typical representation for scenario 1.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Source Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>userType</td>
<td>Categorical</td>
<td>Main</td>
</tr>
<tr>
<td>pointLatitude</td>
<td>Numeric</td>
<td>Main</td>
</tr>
<tr>
<td>pointLongitude</td>
<td>Numeric</td>
<td>Main</td>
</tr>
<tr>
<td>informationType</td>
<td>Categorical</td>
<td>Main</td>
</tr>
<tr>
<td>faunaName</td>
<td>Categorical</td>
<td>Fauna</td>
</tr>
<tr>
<td>faunaType</td>
<td>Categorical</td>
<td>Fauna</td>
</tr>
<tr>
<td>faunaSpecie</td>
<td>Categorical</td>
<td>Fauna</td>
</tr>
<tr>
<td>faunaPresence</td>
<td>Categorical</td>
<td>Fauna</td>
</tr>
<tr>
<td>floraName</td>
<td>Categorical</td>
<td>Flora</td>
</tr>
<tr>
<td>floraType</td>
<td>Categorical</td>
<td>Flora</td>
</tr>
<tr>
<td>floraPresence</td>
<td>Categorical</td>
<td>Flora</td>
</tr>
<tr>
<td>sedimentName</td>
<td>Categorical</td>
<td>Sediment</td>
</tr>
<tr>
<td>sedimentPresence</td>
<td>Categorical</td>
<td>Sediment</td>
</tr>
<tr>
<td>waterFeature</td>
<td>Categorical</td>
<td>Water</td>
</tr>
<tr>
<td>value</td>
<td>Numeric</td>
<td>Water</td>
</tr>
</tbody>
</table>

The similarity measure used as a distance metric module of both agglomerative hierarchical and \(k\)-means clustering algorithms for this scenario was based on the equation defined in Eq. (9). However, in order to optimize and make a more realistic comparison, we dynamically select the attributes to be used in the calculation of the similarity. This selection is performed according to the attributes that have "is-a" relation. The idea is to calculate the similarity of the attributes that the instances have in common. Basically, taking as example informationType attribute, the selection process is defined as follows.

- If the value for attribute informationType is the same for both instances, they are considered as having the same links. Then, select all attributes referring to the corresponding tables.
- If the value for attribute informationType is different, select only the attributes of the main table.

The main aim of this selection process is to use as much target information as possible. In this case, we avoid comparing two instances with different attributes. Furthermore, performing in this way, we avoid the problem of processing missing values (described as "N/A" in the examples), during the clustering procedure.
A novel approach for clustering relational data (scenario 2)

As mentioned on the previous section, we applied our approach to transform the relational dataset in a hierarchical one and the resulting hierarchical model is shown in Figure 1.

Figure 3 shows a small sample of the dataset using the hierarchical modeling of the relational attributes. The fourth attribute of this figure indicates the hierarchical attribute. As already mentioned, the representation of data follows the standard for hierarchical data, in which R denotes the Root, and each full stop “.” denotes a hierarchy level of the data. The approach used in this scenario (scenario 2) was completely different from the one used in scenario 1. First, all the relational data was processed in order to model it in a hierarchical structure.

Algorithm: Compute Hierarchical Distance

INPUT: Inst1, Inst2
BEGIN
IF [attribute is Hierarchical] then
BEGIN
int level ← 1;
WHILE (Inst1’s value at level = Inst2’s value at level) level ← level + 1;
END
Dist ← 2^(-level-1);
Return Dist;
END.

Figure 4. The hierarchical distance.

For this reason, we need to use a similarity measure that takes into account the distance between instances in the hierarchy, but also considers that the similarity at deeper levels is higher than the similarity at shallower levels. In using this distance, the distance value between a “Sea-urchin” and a “Sea-star” attributes needs to be smaller than the distance between a “Sea-urchin” and an “Octopus”.

D. A method for clustering relational data

In this scenario (scenario 3), we use a method for clustering relational data proposed in [6], [7]. The main idea of this method is to cluster the instances of the relational tables separately and then merge them together according to their relationships. Unlike the previous scenarios, we used the agglomerative hierarchical, because it is a deterministic algorithm and it was chosen for simplicity reasons.

In the first stage of this scenario, we clustered the instances of the table main (see Figure 2), varying the same parameters used in scenario 1 (k from 2 to 20). For the secondary tables (water, fauna, flora and sediment), we varied k from 2 to 6, as six is the maximum value for table sediment (sedimentName [3] x sedimentPresence [2]).

In the second stage, we merged clusters from table main with clusters from other tables by using the similarity measure proposed by the authors. For instance, we can merge the clustering results for k = 2 (table main) with the clustering results for k = 2, 3, 4, 5 and 6 (other tables). In this sense, we had to compute the average values for all possible combination (2k-2k, ..., 2k-6k, 3k-2k, ..., 3k-6k, ...).

VIII. EXPERIMENTAL RESULTS

In this section, we present the experimental results for scenarios 1, 2 and 3. In all three scenarios, for both clustering algorithms, the number of k varied from 2 to 20. In the agglomerative hierarchical clustering algorithm, we used the version using average link. Finally, k-means algorithm was performed 10 times, initialized with different seed, since it is a stochastic method. The results presented in this paper represent the average values for all 19 values of k and all 10 executions (for k-means).

As already mentioned, the clustering algorithms will be evaluated in terms of two internal indexes, DB and Silhouette, and they measure the compactness of the generated clusters throughout the experiments.

Table IV presents the DB and the Silhouette indexes for datasets 1, 2, 3 and 4. For all four datasets, the values represent average index value and standard deviation. In order to compare the performance of the clustering algorithms in the proposed scenarios, a statistical test will be applied, which is called hypothesis test (one-tailed student t-test), with a confidence level of 95% (α = 0.05) [24]. In this table, the best value for each index and each dataset is marked as bold. Then, the statistical test is applied, comparing the best result with the result of the other clustering algorithm. The best results that are statistically significant are underlined.

<table>
<thead>
<tr>
<th>Scenarios 1</th>
<th>Agglomerative Hierarchical</th>
<th>k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>2.46 ± 0.94</td>
<td>0.35 ± 0.05</td>
</tr>
<tr>
<td>Set 2</td>
<td>2.54 ± 0.54</td>
<td>0.34 ± 0.06</td>
</tr>
<tr>
<td>Set 3</td>
<td>2.25 ± 0.85</td>
<td>0.39 ± 0.05</td>
</tr>
<tr>
<td>Set 4</td>
<td>2.50 ± 0.42</td>
<td>0.33 ± 0.07</td>
</tr>
</tbody>
</table>

Note that, in overall perspective, the values obtained by the Agglomerative hierarchical clustering algorithm were
better than the results obtained by the \(k\)-means clustering algorithm in scenario 1. It is important to emphasize that for the DB index the lower the best and for the Silhouette index the bigger the best. Furthermore, the best overall results in scenario 1 were obtained by the Agglomerative hierarchical clustering algorithm when clustering subset 3 (750 instances), for both indexes. Although the Agglomerative hierarchical algorithm had better performance than the \(k\)-means algorithm according to the DB index for three datasets, only for subset 2, the difference of the results proved to be statistically significant. Nevertheless, according to the Silhouette index, the difference was statistically significant for all four.

Table V presents the DB and the Silhouette indexes average results for subsets 1, 2, 3 and 4 in scenario 2. From Table V, it can be observed that, once again, values obtained by the Agglomerative hierarchical clustering algorithm were better than the results obtained by the \(k\)-means algorithm for all four subsets and for both indexes in this scenario. Furthermore, the best average result in scenario 2 was obtained by the hierarchical clustering algorithm when clustering subset 2, for both indexes. The performance of the Agglomerative hierarchical was much better than the \(k\)-means in this scenario. The results obtained by the hierarchical algorithm were statically significant in 6 cases (bold and underline numbers). The only two exceptions were for the DB index with subsets 1 and 3.

**TABLE V. EXPERIMENTAL RESULTS FOR SCENARIO 2.**

<table>
<thead>
<tr>
<th>Scenario 2</th>
<th>Agglomerative Hierarchical</th>
<th>(k)-means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DB</td>
<td>Silhouette</td>
</tr>
<tr>
<td>Subset 1</td>
<td>1.19 ± 0.54</td>
<td>0.42 ± 0.06</td>
</tr>
<tr>
<td>Subset 2</td>
<td>1.17 ± 0.40</td>
<td>0.43 ± 0.05</td>
</tr>
<tr>
<td>Subset 3</td>
<td>1.48 ± 0.40</td>
<td>0.43 ± 0.05</td>
</tr>
<tr>
<td>Subset 4</td>
<td>1.26 ± 0.42</td>
<td>0.42 ± 0.05</td>
</tr>
</tbody>
</table>

Table VI presents the DB and the Silhouette indexes average results for subsets 1, 2, 3 and 4 in scenario 3. It is important to emphasize that, for this scenario, we only used the hierarchical algorithm since the authors [6], [7] have also used this clustering algorithm in their analysis, and because it has proved in the previous scenarios to have better performance than the \(k\)-means algorithm. For this reason, we have not applied the statistical test for this scenario.

In Table VI, the best DB index and Silhouette results obtained by the TSMRC occurred when clustering subsets 2 and 3, respectively.

**TABLE VI. EXPERIMENTAL RESULTS FOR SCENARIO 3.**

<table>
<thead>
<tr>
<th>TSMRC (Agglomerative Hierarchical)</th>
<th>DB</th>
<th>Silhouette</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset 1</td>
<td>4.19 ± 2.27</td>
<td>0.36 ± 0.11</td>
</tr>
<tr>
<td>Subset 2</td>
<td>2.76 ± 0.75</td>
<td>0.41 ± 0.07</td>
</tr>
<tr>
<td>Subset 3</td>
<td>2.80 ± 0.75</td>
<td>0.42 ± 0.04</td>
</tr>
<tr>
<td>Subset 4</td>
<td>3.79 ± 1.98</td>
<td>0.36 ± 0.13</td>
</tr>
</tbody>
</table>

**A. A Comparison of different scenarios**

After analyzing the different scenarios separately, we will make a comparison of these three ways to handle relational data. Table VII summarizes the results of all three scenarios, for both indexes. In this Table, each column represents a subset and every two or three lines represent the results of the one index applied to one clustering algorithm in the corresponding scenario. For instance, the first data line represents the results of DB indexes for the agglomerative algorithm for scenario 1. As already mentioned, we used only the agglomerative hierarchical algorithm for scenario 3. Hence, there is no scenario 3 for the \(k\)-means algorithm.

Once again the best result obtained by one scenario is marked as bold and its result is compared with the second best result, using the t-test statistical analysis. The best results which are statistically significant are underlined. In analyzing table VII, we can state that scenario 2 has obtained the best results, for both indexes and clustering algorithms. This is an important result since it shows the hierarchical way to handle relational data has obtained more compact and effective clusters. In comparing scenarios 1 and 3, we can see that scenario 1 has obtained better results than scenario 3 for DB index, while scenario 3 has obtained better results than scenario 1 for the silhouette index.

**TABLE VII. RESULTS FOR ALL THREE SCENARIOS.**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>DB - Agglomerative</th>
<th>Silhouette – Agglomerative</th>
<th>DB – k-means</th>
<th>Silhouette – k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sc1</td>
<td>2.46 ± 0.94</td>
<td>0.35 ± 0.05</td>
<td>2.50 ± 0.42</td>
<td>0.33 ± 0.06</td>
</tr>
<tr>
<td>Sc2</td>
<td>1.39 ± 0.54</td>
<td>0.43 ± 0.05</td>
<td>1.44 ± 0.47</td>
<td>0.42 ± 0.05</td>
</tr>
<tr>
<td>Sc3</td>
<td>4.19 ± 2.27</td>
<td>2.41 ± 0.47</td>
<td>2.60 ± 0.35</td>
<td>1.54 ± 0.37</td>
</tr>
<tr>
<td>Sc4</td>
<td>3.79 ± 1.98</td>
<td>1.19 ± 0.54</td>
<td>2.76 ± 0.75</td>
<td>0.33 ± 0.06</td>
</tr>
</tbody>
</table>

In analyzing the performance of both \(k\)-means and hierarchical algorithms in scenarios 1 and 2, we can see that the hierarchical algorithm has provided better results in 15 cases (out of 16). The only exception was for subset 1 applying the DB index. It shows the predominance of the hierarchical algorithm over \(k\)-means.

Finally, the experiments results (Table VII) have shown that the proposed hierarchical structure helped the well-known clustering algorithms (\(k\)-means and Agglomerative hierarchical) to perform better than an existing relational clustering method (TSMRC) as well as a standard way to handle relational data. In our experiments, the hierarchical structure proposed for the relational database has proved to be very helpful in measuring distances between instances. This fact, contributed enormously to the improvement of the performance of the \(k\)-means and Agglomerative hierarchical clustering algorithms.

**IX. CONCLUSION**

The main contribution of this paper is to propose a hierarchical approach in order to handle the real world relational database particularities. By modeling the relational tables and attribute values in a hierarchical structure, we were able to analyze properly the distances between instances within the hierarchy.

Secondly, a few studies have proposed methods and...
similarity measures which can handle the particularities of the relational data. However, our approach handles these particularities without changing attribute values or their relationship. The time consumed with pre-processing algorithms is very low compared to other studies [6], [7]. Furthermore, the number of relational tables does not seem to be a problem for this hierarchical approach.

Based on the experiments results obtained for scenarios 1, 2 and 3, we can conclude that the proposed hierarchical approach for clustering relational database had significant positive results on clustering compact and separated groups in our dataset. In addition, the hierarchical approach is more stable and less dependent on the dataset size (the lowest standard deviation of all scenarios).

Finally, as future work, an investigation needs to be conducted on comparing the performance with other relational clustering methods [5], [25] on clustering the NatalGIS relational database, and also different similarity measures, especially suitable for categorical attributes. Furthermore, another investigation needs to be performed on applying the hierarchical structure to different relational databases.

REFERENCES


