A Complexity Guided Algorithm for Association Rule Extraction on Fuzzy DataCubes

Nicolas Marin, Carlos Molina, José M. Serrano, and M. Amparo Vila

***Abstract*—The use of online analytical processing (OLAP) systems as data sources for data mining techniques has been widely studied and has resulted in what is known as online analytical mining (OLAM). As a result of both the use of OLAP technology in new fields of knowledge and the merging of data from different sources, it has become necessary for models to support imprecision. We, therefore, need OLAM methods which are able to deal with this imprecision. Association rules are one of the most used data mining techniques. There are several proposals that enable the extraction of association rules on DataCubes but few of these deal with imprecision in the process. The main problem observed in these proposals is the complexity of the rule set obtained. In this paper, we present a novel association rule extraction method that works over a fuzzy multidimensional model which is capable of representing and managing imprecise data. Our method deals with the problem of reducing the complexity of the result obtained by using fuzzy concepts and a hierarchical relation between them.**

***Index Terms*—Association rules extraction, complexity reduction, fuzzy multidimensional model, on-line analytical processing (OLAP).**

I. Introduction

AS DEFINED BY the OLAP Council [1] "*on-line analytical processing (OLAP) is a category of software technology that enables analysts, managers, and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user*." According to Han [2], the use of OLAP systems in data mining is interesting for the following three main reasons.

- Data mining techniques need integrated, consistent, and clean data to work with [3]. The data processing performed when building a data warehouse guarantees these qualities in data and converts data warehouses into good data sources for data mining.
- Users frequently need to explore the stored data, selecting only a portion of them, and might want to analyze data at different abstraction levels (different levels of granularity). OLAP systems are designed to ease these operations in a flexible way. The integration of data mining techniques with OLAP provides the user with even more flexibility.
- It is difficult to predict what knowledge is required *a priori*. The integrated use of OLAP and suitable data mining methods allows the user to obtain this knowledge using different approaches and representations.

Information in decision support systems usually has an ill-defined nature. The use of data from human interaction may enrich the analysis [4] and, nowadays, it is common for companies to require external data for strategic decisions. These external data are not always compatible with the format of internal information and even if they are, they are not as reliable as internal data. Moreover, information may also be obtained from semi-structured or nonstructured sources.

In addition, OLAP systems are now being used in new fields of knowledge (e.g., medical data) that present complex domains which are difficult to represent using crisp structures [5]. In all these cases, flexible models and query languages are needed to manage this information. These reasons, among many others, justify the search for multidimensional models which are able to represent and manage imprecision. Some significant proposals in this direction can be found in the literature [6]–[9]. These proposals support imprecision from different perspectives. In [9], we propose a fuzzy multidimensional model that manages imprecision both in facts and in the definition of hierarchical relationships. These proposals organize imprecise data using DataCubes (imprecise DataCubes) and it is, therefore, necessary to develop data mining techniques that can work over these imprecise DataCube models.

Our aim in this paper is to propose a new method to extract association rules from a fuzzy multidimensional model that can represent and manage imprecision in different aspects: COGARE. As we have already mentioned, previous proposals in the literature are directed towards obtaining as many associations as possible. However, they produce complex results (e.g., a high number of rules, rules that represent the same knowledge at different detail levels, etc.). In contrast, our proposal has the following two main goals:

1) to manage data imprecision throughout the entire process;
2) to reduce the complexity of the final result using both the fuzzy concepts and the hierarchical relation between elements, without reducing the quality of the rule set.

Section II of this paper contains a brief introduction to previously published proposals. Section III presents the main concepts of the fuzzy multidimensional model used as the data source in the process. Sections IV and V describe the complexity and quality measures that we use, respectively. The algorithm is then introduced in Section VI, which includes experiments over different domains to test the complexity reduction capability of our approach. This paper ends with the main conclusions and the guidelines that will lead our future work.

Manuscript received August 22, 2006; revised December 7, 2006.

N. Marin and M. Amparo Vila are with the Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain.

C. Molina and J. M. Serrano are with the Department of Computer Science, University of Jaén, Jaén 23071, Spain (e-mail: carlosmo@ujaen.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2007.905909

II. Association Rule Extraction

In this section, we will briefly describe the main published approaches for association rule extraction. We will first discuss classical methods. As the multidimensional models usually define hierarchies, the multilevel association methods are interesting when studying the association rule extraction over them. Section II-C introduces the proposed method to work on both crisp and fuzzy DataCubes.

A. Association Rules

Agrawal *et al.* [10] formalized the problem of association rule extraction. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of literals called items and $\mathcal{D}$ be a set of transactions, where each transaction $\mathcal{T}$ is a set of items such that $\mathcal{T} \subseteq \mathcal{I}$. A transaction $\mathcal{T}$ contains the set of items X if $X \subseteq \mathcal{T}$.

Definition 1: Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of literals and $\mathcal{D}$ be a set of transactions defined over $\mathcal{I}$. An association rule is an implication $X \rightarrow Y$, where $X \subset \mathcal{I}, Y \subset \mathcal{I}$, and $X \bigcap Y = \emptyset$.

The quality of the rules is usually measured in terms of the *confidence* and the *support* of the rule. The confidence is computed as the percentage of transactions that contain both X and Y with respect to the transactions that contain X, while the support of the rule is the percentage of transactions that contain $X \bigcup Y$ in the entire dataset. Generally speaking, only the rules with a *confidence* and *support* greater than a threshold are interesting. These rules are called *strong rules*.

The association rule extraction process is divided into the following two phases.

1) Discover the *frequent item sets*, i.e., the sets of items with a *support* greater than a given threshold.
2) Build the association rules using the previously obtained frequent item sets.

Since the first step is the most time consuming, there are proposals which focus on the optimization of the frequent item set calculation [11]–[15].

B. Multiple-Level Association Rules

The use of taxonomies over the data is interesting because the desired associations may not appear at the most detailed level but at higher levels. Researchers have also paid attention to this approach, and a first proposal [16] applies a rule extraction process to all the levels. The authors define an *interest measure* that is used to prune the rule set. In this way, a rule is deleted if a rule defined at a higher level exists and the first does not give more information. This approach considers the same *support* threshold for all the levels.

When the taxonomy is complex or when a relatively high number of levels is considered, a high number of rules appears. Let us explain this with a naive example. Fig. 1 represents a possible taxonomy over data. Let us suppose that the item set {Age is 13, Date is 05/22/2006} is frequent; then all the items that group one of the values will be frequent, and all the combinations of these values will also be frequent.

This circumstance will imply that the following item sets will be frequent: {Young, Date is 05/22/2006}, {Age is 13, May}, {Young, May}, {No legal age, Date is 05/22/2006}, {No legal age, May}, {Age is 13, Year 2006}, {Young, Year 2006}, and {No legal age, 2006}.

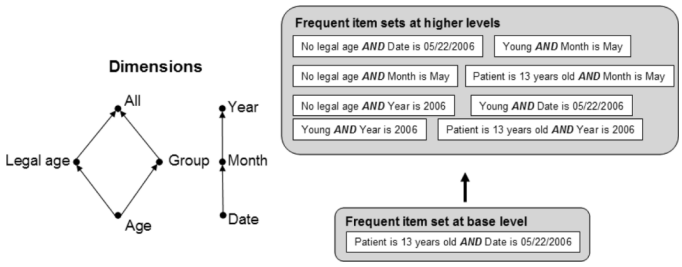


Fig. 1. Frequent item sets at different levels using the same support threshold.

Therefore, for a single item set we obtain another eight frequent item sets that represent exactly the same information at different abstraction levels. When the method finishes, it will produce a high number of rules that are redundant (i.e., they represent the same information at different abstraction levels). This fact only increases the complexity for the user.

The method uses an interesting measure to reduce redundant rules if other rules at a higher level give at least the same information. However, it allows redundant rules if the concrete ones are of a higher quality.

Han and Fu [17] proposed a *top-down* approach using a single taxonomy: an item set is considered to be frequent if the support is greater than a threshold and all the ancestors are also frequent. The items belonging to an item set are all defined at the same detail level. The authors used different support thresholds for each level that must be established by the user (if the taxonomy is complex, this involves a large number of parameters), and do not consider multiple taxonomies over the items. Thus, if the domain is complex, it may not be modeled well.

Shen and Shen [18] proposed another method that extracts all the strong rules which are defined at all the combinations of detail levels using different taxonomies, considering the same threshold. Yen [19] used a graph structure to obtain the relationships between elements at all the detail levels. Both approaches present the same problems as those mentioned for Srikant and Agrawal's proposal: a large number of rules with redundant information.

The method proposed by Lui and Chung [20] uses a *bottom-up* approach. In this case, the method considers two infrequent item sets of the same size with common items, and generalizes them to a new one, which is a candidate frequent item set in a second stage. The *support* threshold for each item set is calculated according to a proposed *generality measure*. As the generalization process is applied only once, if the relationships appear at higher levels, the method will not discover them.

Another approach to association rule extraction using taxonomies is *attribute-oriented induction (AOI)*. In this case, the taxonomies are used to raise the abstraction of the items before the process is applied. In line with this idea, several methods have been proposed [21], [22] and these have recently been extended to use fuzzy hierarchies [23]. The idea is to reduce the number of rules decreasing the number of items to consider. The main problem of all these approaches is that since generalization is applied before rule extraction, information is lost in the process.

C. Association Rules Over Datacubes

Let us now briefly describe some proposals for association rule extraction on DataCubes. The first approach can be found in [24]. The authors proposed a method that works over a very simple multidimensional model (there are neither hierarchies on the dimensions nor grouping mechanisms to change the granularity of the data) and which focuses on frequent item set calculation to speed up the process using DataCube operations.

Zhu [25] proposed a more complex approach, with the definition of the following three different associations.

1) *Intra-Dimensional Association:* The association is found between elements in the same dimension (*item dimension*), using another dimension to calculate the support (*transaction dimension*).
2) *Inter-Dimension Association:* In this case, the associations hold between elements in different dimensions.
3) *Hybrid Association:* This association is the result of merging the two previous types. The method first looks for intra-dimensional frequent item sets and then for inter-dimensional frequent item sets, merging the resulting sets in order to obtain the rules.

In all the cases, this method works over a single abstraction level, and the support threshold is, therefore, a single value. For multilevel association rules, the user must run the method for all the level combinations required, defining the same number of support thresholds as executions (something which may be very complex for the user) or using the same value for the entire process. As the final result is the union of all of the obtained rule sets, there may be a high number of rules and repeated associations expressed at different abstraction levels (as in Srikant and Agrawal's proposal as mentioned in Section II-B).

Finally, Kaya and Alhajj [7], [26] propose a method that works over a simple fuzzy multidimensional model. The proposed DataCube structure defines fuzzy relations at the lowest abstraction level and does not support imprecision in hierarchies or facts, as well as the normal operations over DataCubes (e.g., changing the detail level, reducing the dimensionality of the DataCube, etc.). Under these circumstances, users would have less flexibility since they cannot explore the data. The proposed method extracts association rules at all abstraction levels, obtaining intra-dimensional and inter-dimensional associations as previously presented. The user must establish a support threshold for each level in the DataCube and the threshold for an item set will be the minimum of the threshold established for each item. The authors use an *interesting measure* to reduce certain problems of *confidence* when measuring the quality of the rules, but do not control the redundant associations defined at different abstraction levels.

III. The Fuzzy Multidimensional Model

Although there is no standard multidimensional model, we shall briefly introduce the common characteristics of the first models proposed in literature. In classical multidimensional models, we can distinguish two different types of data: on the one hand, we have the facts being analyzed, and on the other, the dimensions that are the context for the facts. Hierarchies may be defined in the dimensions [27]–[30].

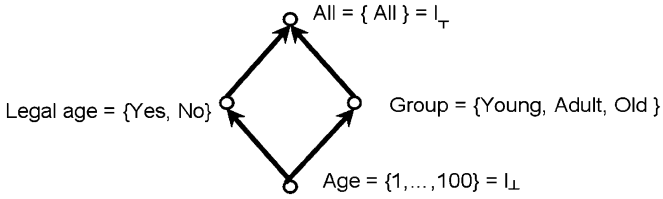


Fig. 2. Example of a hierarchy over ages.

The different levels of the dimensions allow us to access the facts at different levels of granularity. In order to do so, classical aggregation operators are needed (maximum, minimum, average, etc.). Other models, which do not define explicit hierarchies on the dimensions, use other mechanisms to change the detail level [31], [32]. The model proposed by Gray *et al.* [33] uses a different approach. This model defines two extensions of the relational *group by* (*rollup* and *cube*) that are used to group the values during the aggregation process.

As the models that define hierarchies usually use many-to-one relations, one element in a level can only be grouped by a single value of each upper level in the hierarchy. This makes the final structure of a DataCube rigid and well defined in the sense that given two values of the same level in a dimension, the set of facts relating to these values have an empty intersection. The normal operations (roll-up, drill-down, dice, slice, and pivot) are defined in almost all the models. Eventually, some of the models define other operations in order to provide the end user with additional functionality [27], [31], [32].

A. Fuzzy Multidimensional Structure

In this section, we will briefly introduce a fuzzy multidimensional model which we have already developed to manage data imprecision. The model is provided with explicit hierarchies that can use fuzzy relations between elements in two levels. A more detailed description can be found in [9]. Here, we only present the main concepts needed to understand the implemented model.

Definition 2: A *dimension* is a tuple $d = (l, \leq_d, l_\perp, l_\top)$, where $l = l_i$, $i = 1, \ldots, n$ so that each l_i is a set of values $l_i = \{c_{i1}, \ldots, c_{in}\}$ and $l_i \cap l_j = \emptyset$ if $i \neq j$, and $\leq_d$ is a partial order relation between the elements of l so that $l_i \leq_d l_k$ if $\forall\, c_{ij} \in l_i \Rightarrow \exists c_{kp} \in l_k | c_{ij} \subseteq c_{kp}$. $l_\perp$ and $l_\top$ are two elements of l so that $\forall\, l_i \in l\ l_\perp \leq_d l_i \leq_d l_\top$.

We use *level* to denote each element l_i. In order to identify level l of dimension d, we will use $d.l$. The two special levels $l_\perp$ and $l_\top$ will be called the *base level* and *top level*, respectively. The partial order relation in a dimension is what gives the hierarchical relation between the levels. An example of dimension on the ages can be found in Fig. 2.

The domain of a dimension will be the set of all the values that appear in all the defined levels.

Definition 3: For each dimension d, the domain is $\mathrm{dom}(d) = \bigcup_i l_i$

In the previous example, the domain of the dimension *Age* is $\mathrm{dom}(\text{Age}) = \{1, \ldots, 100, \text{Young, Adult, Old, Yes, No, All}\}$.

Definition 4: For each l_i, the set

$$H_i = \{l_j | l_j \neq l_i \wedge l_j \leq_d l_i \wedge \neq g \exists l_k\ l_j \leq_d l_k \leq_d l_i\} \quad (1)$$

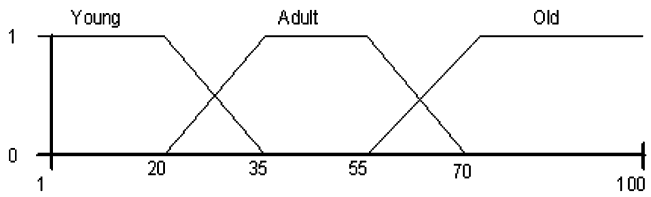


Fig. 3. Kinship relationship between levels Group and Age.

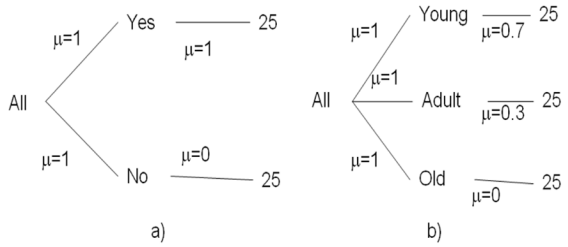


Fig. 4. Example of the calculation of the extended kinship relationship: (a) path *All—Legal age—Age* and (b) path *All—Group—Age*.

is called the *set of children of level* l_i.

This set defines the set of all the levels which are below a certain level (l_i) in the hierarchy. In addition, this set gives the set of levels whose values or labels are generalized by the ones included in l_i. Using the same example of the dimension on the ages, the set of children in level All is $H_{\text{All}} = \{\text{Group,Legal age}\}$. In all the dimensions defined for the *base level*, this set will always be empty (as the definition shows).

Definition 5: For each l_i the set

$$P_{l_i} = \{l_j | l_i \neq l_j \wedge l_i \leq_d l_j \wedge \neq g \exists l_k \; l_j \leq_d l_k \leq_d l_i\} \tag{2}$$

and we call this the *set of parents of level* l_i.

For a certain level, this set shall give all the levels that group or generalize the values of the level. In the hierarchy we have defined, the set of parents in level Age is $P_{\text{Age}} = \{\text{Legal age, Group}\}$. In the case of the *top level* of a dimension, this set shall always be empty.

In the case of fuzzy hierarchies, an element can be related to more than one element in the upper level and the degree of this relationship is in the interval $[0, 1]$. The *kinship relationship* defines this degree of relationship.

Definition 6: For each pair of levels l_i and l_j such that $l_j \in H_i$, we have the relation

$$\mu_{ij} : l_i \times l_j \to [0, 1] \tag{3}$$

and we call this the *kinship relationship*.

The degree of inclusion of the elements of a level in the elements of their parent levels can be defined using this relation. If we only use the values 0 and 1 and we only allow an element to be included with degree 1 in a unique element of its parent levels, this relation represents a crisp hierarchy.

If we relax these conditions and we allow values to be used in the interval $[0, 1]$ without any other limitation, we have a fuzzy hierarchical relationship. This allows several hierarchical relations to be represented in a more intuitive way. An example can be seen in Fig. 3, where we present the group of ages according to linguistic labels. Furthermore, this fuzzy relation allows hierarchies to be defined where there is imprecision in the relationship between elements of different levels. In this situation, the value in the interval shows the degree of confidence in the relation.

Using the relation between elements in two consecutive levels, we can define the relation between each pair of values in different levels in a dimension.

Definition 7: For each pair of levels l_i and l_j of dimension d such that $l_j \leq_d l_i \wedge l_j \neq l_i$, the relation $\eta_{ij} : l_i \times l_j \to [0, 1]$ is defined as

$$\eta_{ij}(a, b) = \begin{cases} \mu_{ij}(a, b), & \text{if } l_j \in H_{l_i} \\ \bigoplus_{l_k \in H_{l_i}} \bigoplus_{c \in l_k} (\mu_{ik}(a, c) \otimes \eta_{kj}(c, b)), & \text{otherwise} \end{cases} \tag{4}$$

where $\otimes$ and $\oplus$ are a t-norm and a t-conorm, respectively, or operators from the families MOM and MAM defined by Yager [34], which include the t-norms and t-conorms, respectively. This relationship is called the *extended kinship relationship*.

This relation gives us information about the degree of relationship between two values in different levels within the same dimension. In order to obtain this value, it considers all the possible paths between the elements in the hierarchy. Each one is calculated by aggregating the kinship relationship between elements in two consecutive levels using a t-norm. The final value is then the aggregation of the results of each path using a t-conorm.

By way of example, we will show how to calculate the value of $\eta_{\text{All,Age}}(\text{All}, 25)$. In this situation, we have the following two different paths.

1) *All—Legal age—Age*. In Fig. 4(a), it is possible to see the two ways of reaching 25 from All through the level legal age. The result of this path is $(1 \otimes 1) \oplus (1 \otimes 0)$.
2) *All—Group—Age*. This situation is very similar to the previous one. In Fig. 4(b), it is possible to see the three different paths going through the level Group. The result of this path is $(1 \otimes 0.7) \oplus (1 \otimes 0.3) \oplus (1 \otimes 0)$.

We must now aggregate these two values using a t-conorm in order to obtain the final result. If we use the maximum as t-conorm and the minimum as t-norm, the result is $((1 \otimes 1) \oplus (1 \otimes 0)) \oplus ((1 \otimes 0.7) \oplus (1 \otimes 0.3) \oplus (1 \otimes 0)) = (1 \oplus 0) \otimes (0.7 \oplus 0.3 \oplus 0) = 1 \oplus 0.7 = 1$

Thus, the value of $\eta_{\text{All,Age}}(\text{All}, 25)$ is 1, which means that the age 25 is grouped by *All* in level *All* with grade 1.

Definition 8: We say that any pair (h, α) is a *fact* when h is an m-tuple on the attribute domain we want to analyze, and $\alpha \in [0, 1]$.

The value α controls the influence of the fact in the analysis. The imprecision of the data is managed by assigning an α value that represents this imprecision. When we operate with the facts, the aggregation operators must manage these values in the computations. The arguments for the operator can be seen as a fuzzy bag [35], [36] since they are a set of values with a degree in the interval $[0, 1]$ that can be duplicated. The result of the aggregation must also be a fact. So, in the fuzzy case, the aggregation operators may be defined as follows.

Definition 9: Let $\tilde{B}(X)$ be all the possible fuzzy bags defined using elements in X, $\tilde{P}(X)$ be the fuzzy power set of X, and D_x be a numeric or natural domain. We define an *aggregation operator* G as a function $G : \tilde{B}(D_x) \to \tilde{P}(D_x) \times [0,1]$.

When we apply an aggregation operator, we summarize the information of a bag of values into a single value and it is not always possible to undo this operation. If we want to undo operations that reduce the level of detail in a DataCube, we, therefore, need something to prevent this problem and so we define the object *history* that stores a DataCube's aggregation states.

Definition 10: An object of type *history* is the recursive structure

$$\begin{aligned} H^0 &= \Omega \\ H^{n+1} &= (A, l_b, F, G, H^n) \end{aligned} \tag{5}$$

where

- Ω is the recursive clause;
- F is the fact set;
- l_b is a set of levels $(l_{1b}, \ldots, l_{nb})$;
- A is an application from l_b to F $(A : l_b \to F)$;
- G is an aggregation operator.

This structure enables detail levels of the DataCube to be stored while it is operated on so that it may be restored to a previous level of granularity.

We can now define the structure of a fuzzy DataCube. A DataCube can be considered to be the union of a set of facts (the variables to analyze) and a set of dimensions (the context of the analysis). In order to report the facts and dimensions, we need a correspondence which for each combination of values of the dimension gives us the fact related to these coordinates in the multidimensional space defined by the dimensions.

In addition to these DataCube features, we also need the levels that establish the detail level that the facts are defined with, and a *history*-type object that keeps the aggregation states during the operations. The DataCube is, therefore, defined in the following way.

Definition 11: A *DataCube* is a tuple $C = (D, l_b, F, A, H)$ such that it follows:

- $D = (d_1, \ldots, d_n)$ is a set of dimensions;
- $l_b = (l_{1b}, \ldots, l_{nb})$ is a set of levels such that l_{ib} belongs to d_i;
- $F = R \cup \emptyset$, where R is the set of facts and $\emptyset$ is a special symbol;
- H is a history-type object;
- A is an application defined as $A : l_{1b} \times \cdots \times l_{nb} \to F$ that gives the relation between the dimensions and the facts defined.

If for $\vec{a} = (a_1, \ldots, a_n)$, $A(\vec{a}) = \emptyset$, this means that no fact is defined for this combination of values. Normally, not all the combinations of level values have facts. This situation is shown by the symbol $\emptyset$ when application A is defined.

The basis of this analysis will be a DataCube defined at the most detailed level. We shall then refine the information while operating on the DataCube. This DataCube is *basic*.

Definition 12: We say that a DataCube is *basic* if $l_b = (l_{1\perp}, \ldots, l_{n\perp})$ and $H = \Omega$.

B. Operations

Once we have the structure of the multidimensional model, we need the operations to analyze the data in the DataCube. Over this structure, we have defined the usual operations of the following multidimensional model.

Roll-Up: Going up in the hierarchies to reduce the detail level. In this operation, we need to know the facts related with each value in the desired level. The set of facts is obtained using the kinship relationships as follows.

Definition 13: For each value c_{ij} belonging to l_r, we have the set

$$F_{c_{ij}} = \begin{cases} \bigcup\limits_{l_k \in H_{l_r}} F_{c_{kp}} | c_{kp} \in l_k \wedge \mu_{ik}(c_{ij}, c_{kp}) > 0, & \text{if } l_r \neq l_b \\ \{h | h \in F \wedge \exists \vec{c} A(\vec{c}) = h\}, & \text{if } l_r = l_b \end{cases} \tag{6}$$

where $\vec{c} = \left(c_b^1, \ldots, c_{ij}, \ldots, c_b^n\right)$.

Once we have the facts for each value, we must aggregate them to obtain a new fact according to the new detail level. The influence of each fact in the aggregation will depend on the relation of the fact with the value considered and the α value assigned to the fact. Fuzzy operators are needed for this process.

This operation may be defined in the following way.

Definition 14: The result of applying *roll-up* on dimension d_i, level l_{ir} $(l_{ir} \neq l_{i\perp})$, using the aggregation operator G on a DataCube $C = (D, l_b, F, A, H)$ is another DataCube $C' = (D, l_b', F', A', H')$, where it follows:

- $l_b' = (l_{1b}, \ldots, l_{ir}, \ldots, l_{nb})$;
- $A'(c_b^1, \ldots, c_r^i, \ldots, c_b^n) = G(\{h | h \in F_{c_r^i} \wedge \exists c_b^i A(c_b^1, \ldots c_b^i, \ldots, c_b^n) = h\})$;
- F' is the range of A';
- $H' = (A, l_b, F, G, H)$.

Drill-Down: Going down in the hierarchies to increase the detail level. In this operation, we use the *history*-type object. Since this structure stored the initial aggregation state when roll-up operations were applied, by using the information stored in this structure, we can, therefore, get to a previous detail level. The operations may, therefore, be defined as follows.

Definition 15: The result of applying *drill-down* on a DataCube $C = (D, l_b, F, A, H)$, where $H = (A', l_b', F', H')$ is another DataCube $C' = (D, l_b', F', A', H')$.

Dice: Project over the DataCube using a condition. In this operation, we must identify the values in the dimension that satisfy the condition or that are related with a value that satisfy the condition. This relation is obtained using the kinship relationship. Once we have reduced the values in the dimension, we must eliminate the facts for which the coordinates have been removed.

Definition 16: The result of applying *dice* with the condition β on level l_r of dimension d_i in a DataCube $C = (D, l_b, F, A, H)$ is another DataCube $C' = (D', l_b', F', A', \Omega)$, where it follows:

- $D' = d_1, \ldots, d_i', \ldots, d_n$ with $d_i' = (l_i', \leq_{d_i}, l_b, l_\top)$, where $l' = l_j | l_b \leq_d l_j$ and

$$d_i'.l_j' = \begin{cases} \{c_{jk} | c_{jk} \in l_j \wedge \beta(c_{jk})\}, & \text{if } l_j' = l_r \\ \{c_{jk} | c_{jk} \in d_i.l_j \wedge \delta_{rj}(c_{jk})\}, & \text{if } l_j' \leq_d l_r \\ \{c_{jk} | c_{jk} \in d_i.l_j \wedge \delta_{jr}(c_{jk})\}, & \text{if } l_r \leq_d l_j' \end{cases}$$

where $\delta_{ij}(c) = \exists c_r \in l_r \beta(c_r) \wedge \eta_{ij}(c_r, c) > 0$;

- $A'\left(c_b^1,\ldots,c_b^i,\ldots,c_b^n\right)=(h,\alpha)|c_b^1 \in d_1'.l_b' \wedge \cdots \wedge c_b^n \in d_n'.l_b' \wedge A\left(c_b^1,\ldots,c_b^n\right)=(h,\alpha)$;
- F' is the range of A'.

Slice: Reduce the dimensionality of the DataCube. When we apply this operation, we eliminate one of the DataCube's dimensions and so we must adapt the granularity of the facts using a fuzzy aggregation operator.

Definition 17: The result of applying *slice* on dimension d_i using the aggregation operator G in a DataCube $C=(D,l_b,F,A,H)$ is another DataCube $C'=(D',l_b',F',A',\Omega)$, where it follows:

- $D'=(d_1,\ldots,d_{i-1},d_{i+1},\ldots,d_n)$;
- $l_b'=(l_{ib},\ldots,l_{i-1b},l_{i+1b},\ldots,l_{nb})$;
- $A'(c_b^1,\ldots,c_b^{i-1},c_b^{i+1},\ldots,c_b^n) = G(\{h|\exists c_b^i A(c_b^1,\ldots,c_b^{i-1},c_b^i,c_b^{i+1},\ldots,c_b^n)=h\})$;
- F' is the range of A'.

Pivot: Change the order of the dimensions. This operation does not affect the facts, only the order of the coordinates that defined them.

Definition 18: The result of applying *pivot* on dimensions d_i and d_j in a DataCube $C=(D,l_b,F,A,H)$ is another DataCube $C'=(D',l_b',F,A',\Omega)$, where it follows:

- $D'=(d_1,\ldots,d_{i-1},d_j,d_{i+1},\ldots,d_{j-1},d_i,d_{j+1},\ldots,d_n)$;
- $l_b'=(l_{1b},\ldots,l_{i-1b},l_{jb},l_{i+1b},\ldots,l_{j-1b},l_{ib},l_{j+1b},\ldots,l_{nb})$;
- $A'\left(c_b^1,\ldots,c_b^{i-1},a_b^i,a_b^{i+1},\ldots,c_b^{j-1},c_b^j,c_b^{j+1},\ldots,c_b^n\right)=A\left(c_b^1,\ldots,c_b^{i-1},a_b^j,c_b^{i+1},\ldots,c_b^{j-1},a_b^i,a_b^{j+1},\ldots,c_b^n\right)$.

The properties of these operations have been studied in [9].

IV. Complexity Measure

Since our approach is supposedly driven by the desire to reduce the complexity of the obtained results, we therefore need to measure a rule sets complexity in order to compare different results and decide which is the least complex. We follow a similar approach to Atzmueller *et al.* [37] by considering the following two factors for the complexity.

- *Number of Rules:* The greater the number of rules in the results, the greater the complexity for the user. Section IV-A will describe a function to measure this factor.
- *Complexity of the Rule Elements:* Very specific values (e.g., dates) result in more specific information but are more difficult for the user to understand than elements in higher abstraction levels (e.g., months instead of specific dates). Section IV-B presents the functions for measuring the abstraction of a rule and a set of rules.

A. Number of Rules

As we have already mentioned, a large number of rules will increase the complexity and make the rule set harder to understand. We want to measure the complexity as a value in the $[0,1]$ interval. A rule set with a complexity value which is close to 0 will have very few rules, while a value which is close to 1 will correspond to a set with a high cardinality. Under these circumstances, a function can be considered to measure the complexity if it satisfies the following definition.

Definition 19: A function C_{NR} defined as

$$C_{NR}:\mathbb{N}\rightarrow[0,1] \tag{7}$$

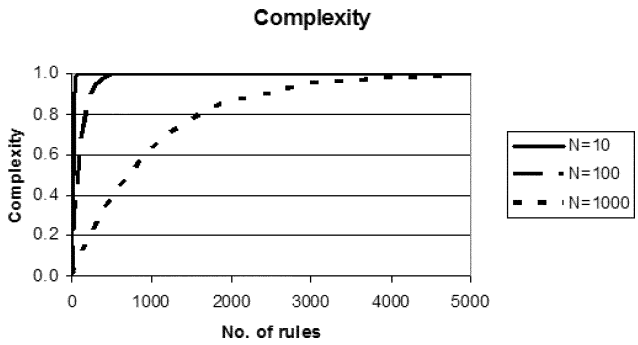


Fig. 5. Complexity function due to number of rules.

is a *complexity function based on the number of rules* when $C_{NR}(x)\geq C_{NR}(y)$, for all x and y such that $x>y$.

All the functions with this behavior can be used to measure the complexity produced by the number of rules. Nevertheless, this definition does not take into account the size of the problem, i.e., the number of items. If we get a result with 100 rules for a problem that involves relations among 100 items, we can intuitively conclude that this set presents less complexity than another with the same number of rules for a problem with 10 items. This is why we think that the complexity function should also depend on the size of the problem.

Similarly, two result sets for the same problem with either 4000 or 5000 rules will be about as difficult to understand. If, however, the sets have either 10 or 100 rules, although the difference in cardinality is less than in the other case, there will be a greater difference in complexity from the user's point of view. According to this intuitive behavior, the function should not present a linear behavior. Taking this discussion into account, the following function is proposed.

Definition 20: Let N be the number of items in the dimensions of the DataCube C. The complexity of the rule set C_R over the DataCube C is a function $C_{NR}:\mathbb{N}\rightarrow[0,1]$ with the value

$$C_{NR}(C_R)=1-e^{-(|C_R|/N)}. \tag{8}$$

Fig. 5 shows the behavior of the function for three different problem sizes.

B. Abstraction

The abstraction of an item will depend on the level defined. In a DataCube, elements at higher levels will present a higher abstraction than elements at lower ones, since the first ones group the second ones. Thus, intuitively, an abstraction function would behave in the following way.

Definition 21: Let D be a dimension. A function A defined as

$$A:\mathrm{dom}(D)\rightarrow[0,1] \tag{9}$$

is an *abstraction function* if it satisfies the following properties.

- If $c_i\in l_i \wedge c_j\in l_j \wedge l_j\in P_{l_i}$, then $A(c_j)\geq A(C_i)$ (the abstraction increases if we go up through the hierarchies defined in the dimension).
- If $c_i\in l_\perp$, then $A(c_i)=0$ (all the elements at the most detailed level (the base level of the dimension) have the lowest possible abstraction).

- If for $c_i \in l_i$ we get $\forall\, c_\perp \in l_\perp : \eta_{i\perp}(c_i, c_\perp) = 1$, then $A(c_i) = 1$ (an element that groups all the elements in the base level has the highest possible abstraction).

In view of the established properties, the abstraction function must take into account the granularity of the elements. One possibility is to define the abstraction according to the levels in the hierarchy. In this case, all the items in a level will share the abstraction value. This situation, however, can present certain problems because elements at the same level will not always have the same granularity. For example, if we consider a level to define *Legal age*, this level has two values: *Yes* and *No*. In Spain, both values will group different numbers of ages (the value *No* groups the ages $\{1, \ldots, 17\}$ and *Yes* the remaining values $\{18, \ldots 100, \ldots\}$) so both have different levels of granularity, and naturally, different levels of (*Yes* abstraction group more values so they appear to present higher abstraction than *No*). Therefore, the proposed abstraction function considers each element independently of its level but measures its granularity. The following definition presents the abstraction function we have chosen.

Definition 22: A is an abstraction function defined as $A : \mathrm{dom}(D) \rightarrow [0,1]$ when for an element $c_i \in l_i$ the value is

$$A(c_i) = \frac{V_{l_i}(c_i)}{|l_\perp|} \tag{10}$$

where $|l_\perp|$ represents the number of elements in the base level of the dimension and

$$V_{l_i}(c_i) = \begin{cases} \sum\limits_{\forall\, c_\perp \in l_\perp} \eta_{i\perp}(c_i, c_\perp), & \text{if } l_i \neq l_\perp \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

It should be noted that we consider the number of elements in the base level grouped by value in order to define the abstraction. This approach is similar to the one proposed by Lui and Chung [20] but considering fuzzy hierarchical relations.

The abstraction of a rule would depend on the items that appear in the rule. Once we know the abstraction of each of the items, the abstraction of the whole rule is defined as the average abstraction of the items that define the rule.

Definition 23: Let R be a rule with the elements $I_1, \ldots, I_N$ and A an abstraction function. The abstraction of the rule is

$$A_R = \frac{\sum\limits_{i=1}^{N} A(I_i)}{N}. \tag{12}$$

In order to measure the abstraction of a rule set, we consider the abstraction of each rule that appears in the set. Not all the rules, however, have the same importance and some may be more representative of the data set according to their support. In order to measure the abstraction of the set, we consider the abstraction of each rule weighted by the support of the rule. Under these considerations, the abstraction of a rule set is defined as follows.

Definition 24: Let $C_R = \{R_1, \ldots, R_N\}$ be a rule set with associated support $\mathrm{sop}(R_1), \ldots, \mathrm{sop}(R_N)$ and A be an abstraction function. The abstraction of C_R would be

$$A_{C_R} = \frac{\sum\limits_{i=1}^{N} A_{R_i} \times \mathrm{sop}(R_i)}{\sum\limits_{i=1}^{N} \mathrm{sop}(R_i)}. \tag{13}$$

C. Global Measure

In previous sections, we have defined two functions which are useful for measuring the complexity due to the number of rules and to the abstraction of a result rule set. In order to define a global measure, we now need to combine both functions to obtain a value in $[0, 1]$ that represents the complexity of the set according to both factors.

Definition 25: Let $\alpha \in [0,1]$. We define the global complexity of a rule set C_R as

$$C_{\mathrm{global}}(C_R) = \alpha \times C_{NR}(C_R) + (1 - \alpha) \times (1 - A_{C_R}). \tag{14}$$

Depending on the value of α, the function controls the relative importance of each complexity measure in the final value. The abstraction of the items will help in the comprehension of the rules but the number of rules may have a greater influence on the complexity for the final user (intuitively, a low number of rules with concrete values will be easier to understand than a high number of rules defined at high abstraction levels). Therefore, we suggest a value of $\alpha = 0.7$.

V. Quality Measures

The method we propose is based on the complexity of the result obtained, but controlling the quality loss of the rule set. Thus, the method must use a quality measure. We first describe the measures that are conventionally used. Later, a new way of computing the quality of a rule set based on these measures is introduced.

A. Classical Measures

In this section, we will present some of the main quality measures used in the literature. We will only introduce the expression and briefly comment on the characteristics of these measures. For a deeper study, the reader can consult comparative studies about the performances of these measures in [38]–[41]. All the measures can be expressed in terms of relative frequencies. If $R : A \rightarrow C$ is a rule, Table I shows the contingency table with relative values.

Consistency is the normal quality measure used in association rule extraction, called in this field the *Confidence* of the rule. Its aim is to measure specificity, but various problems arise when very frequent items appear. The *Coverage* measures the extent to which the domain of the consequent is covered by the rule (the maximum value is reached when all the elements that satisfy C are covered by the rule). Both the *Confidence* and *Coverage*, measure two important factors for the rule quality, but if we use them separately we can reach bad conclusions (rules that cover few elements in the case of *Consistency*, or a high

TABLE I
CONTINGENCY RELATIVE VALUES

	Satisfy C	Not satisfy C	
Covered by R	f_{rc}	$f_{r\bar{c}}$	f_r
Not covered by R	$f_{\bar{r}c}$	$f_{\bar{r}\bar{c}}$	$f_{\bar{r}}$
	f_c	$f_{\bar{c}}$	1

TABLE II
QUALITY MEASURES

Measure	Expression
Consistency	$Cons(R) = \frac{f_{rc}}{f_r}$
Coverage	$Cov(R) = \frac{f_{rc}}{f_c}$
Michalski ([42])	$M(R) = w_1 \times Cons(R) + w_2 \times Cov(R)$ where $w_1 = 0.5 + \frac{1}{4}Cons(R)$ and $w_2 = 0.5 - \frac{1}{4}Cons(R)$
Brazdil and Torgo ([43])	$BT(R) = Cons(R) \times e^{Cov(R)-1}$
Certainty factor	$CF(R) = \begin{cases} \frac{Cons(R)-f_c}{1-f_c} & \text{if } Cons(R) > f_c \\ \frac{Cons(R)-f_c}{f_c} & \text{if } Cons(R) < f_c \\ 0 & \text{in other case} \end{cases}$
Cohen	$Cohen(R) = \frac{Cons(R)-f_c}{\frac{1}{2}(1+\frac{Cons(R)}{Cov(R)})-f_c}$
Coleman	$Coleman(R) = \frac{Cons(R)-f_c}{1-f_c}$
Bruha ([46])	$C1(R) = Coleman(R) \times \frac{2+Cohen(R)}{3}$ $C2(R) = Coleman(R) \times \frac{1+Cov(R)}{2}$
Information gain	$IG(R) = -log(f_c) + log\frac{f_{rc}}{f_r}$
Logical sufficiency	$LS(R) = \frac{f_{rc}/f_c}{f_{r\bar{c}}/f_{\bar{c}}}$
Discrimination	$D(R) = log\frac{f_{rc}/f_{\bar{r}c}}{f_{r\bar{c}}/f_{\bar{r}\bar{c}}}$
IMAFO	$IMAFO(R) = (AC \times E_C) \times 10$ where $AC = f_{rc} + f_{\bar{r}c}$ and $E_C = e^{\frac{f_{rc}}{f_c}-1}$

number of false positives when using the *Coverage*). To improve the performance, certain authors have proposed a combination of both measures: Michalski [42] uses a weighted combination which concedes greater importance to *Consistency* when it reaches high values, and Brazdil and Torgo [43] propose a combination that adjusts the *Consistency* according to the *Coverage*.

Another classical measure is the *Certainty factor*, proposed by Shortliffe and Buchanan in 1975 [44]. This has been used in induction systems and measures both direct relations (antecedent implies the consequent) and indirect relations (when the antecedent appears, it implies no occurrence of the consequent). This measure has also been used in association rules [45] because it does not present some of the problems of the *Confidence*.

Agreement measures use the main diagonal of the contingency table, and *Cohen* and *Coleman*'s measures are defined in this way. Bruha [46] proposed two measures that attempt to combine the best characteristics of both. Measures from other knowledge fields have also been used to compute the quality of the rules, such as *Information Gain*, *Logical Sufficiency*, and *Discrimination*. In some situations, the measures lack a formal analysis of the dependencies and are empirically defined. An example of these measures is *IMAFO* [47] which combines two measures for the accuracy and coverage of the rules.

Table II gathers the expressions of the measures based on the contingency values.

B. Quality Measure for a Rule Set

Although all of the previously presented quality measures compute the quality of a given rule, we need to measure the quality of an entire rule set. As we have previously done with abstraction, we now propose a general measure that takes into account the importance of each rule. We propose the use of the weighted arithmetic average. The following definition shows the quality measure for a rule set.

Definition 26: Let $C_R = \{R_1, \ldots, R_N\}$ be a set of rules, $\text{sop}(R_i)$ be the support for rule R_i, and Q_R be a quality measure for the rules. The quality of the rule set is defined as

$$Q_{C_R} = \frac{\sum_{i=1}^{|C_R|} Q_R(R_i) \times \text{sop}(R_i)}{\sum_{i=1}^{|C_R|} \text{sop}(R_i)}. \quad (15)$$

VI. COGARE ALGORITHM

As we have already mentioned, one of the main problem of previous rule extraction methods is the complexity of the results. Normally, the number of rules obtained is high and this complicates their interpretation. In addition, if the elements used to define the rules have a high level of detail, they will be even more complex for the user.

In this section, we will propose a new method to accomplish this task based on fuzzy DataCubes: complexity guided association rule extraction (COGARE). This method extracts inter-dimensional association rules and tries to reduce the complexity of the obtained rules using the fuzzy concepts defined in the dimensions and hierarchies. The use of fuzzy logic allows concepts to be defined more naturally from the user's point of view. If the rules are defined using these concepts, they will be more understandable for the user due to the use of concepts nearer to the user's language. The hierarchies are helpful in the following two ways.

1) It is possible that a relation does not appear in a detailed level but can be found at higher detail levels. Thus, by using hierarchies we can extract rules at different abstraction levels and get information that does not appear at lower levels.
2) According to the hierarchical relation between elements, the number of rules can be reduced because some rules can be generalized to a single rule using elements which group the elements that appear in the rules we want to reduce.

COGARE is based on these ideas, and two main steps can be identified in the following method.

- *Rule Generation*: The extraction begins by obtaining rules at the most detailed possible level. It attempts to calculate the frequent item sets at base levels of the dimensions. If an item set is not frequent at this level, the method generalizes the items using the hierarchies. This process is repeated until the item set is frequent or the elements can no longer be generalized. The rules are generated using these frequent item sets.
- *Generalization Process*: The result of the previous step is then generalized using the hierarchical relations. In this case, the method tries to reduce the complexity of the result, using more abstract elements in the definition of the rules and reducing the cardinality. In this step, the quality loss is also controlled.

Since the method is developed to work over fuzzy DataCubes, COGARE manages fuzzy concepts in both steps. The following sections will explain each phase.

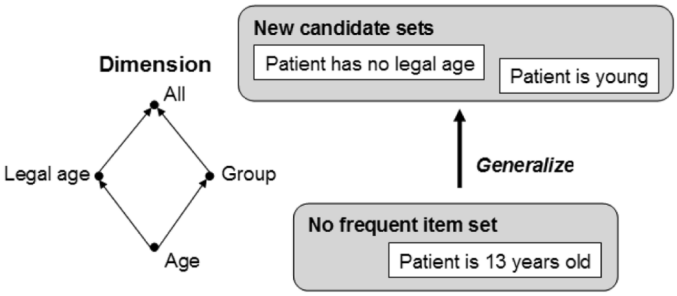


Fig. 6. Example of generalization of nonfrequent one-itemset.

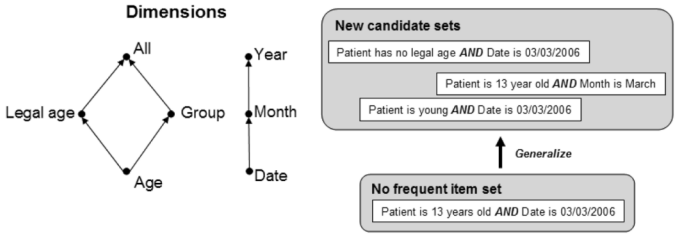


Fig. 7. Example of generalization of nonfrequent two-item sets.

A. Rule Generation

In this phase, the algorithm extracts association rules between elements at different dimensions and multiple levels. We can differentiate the following two steps:

1) obtain the frequent item sets;
2) generate rules using the item sets found in the previous step.

COGARE uses an extension of the Apriori algorithm [11]. Candidates to obtain the frequent one-itemset (item sets that only have one element) are all the elements defined at the base level of all the dimensions

$$C_1 = \bigcup_{\forall\, D_i \in C} l_{i\perp} \tag{16}$$

where C is a DataCube. An item set will be frequent if its support is equal to or greater than a given threshold. For the base level, the process uses a value given by the user ($\mathrm{threshold_{SUP}}$). If the item set is not frequent, then it is generalized, considering all the elements in parent levels that are directly connected and that group the item (see Fig. 6). The new item sets obtained are considered as candidates. This process is repeated until the item set is accepted as frequent or we can no longer generalize. We follow a similar strategy to Lui and Chung's proposal [20].

These new item sets are defined using elements at a more abstract level. Each item may group more than one element at the base level. Then, to be considered interesting, the support threshold should be defined according to the abstraction level. All the elements at higher levels may group several values at the base level; the support threshold should, therefore, be greater than the one established for these. Under these circumstances, the algorithm should use different support thresholds for each abstraction level. Some approaches ask the user for a value for each level [7]. Depending on the number of dimensions and the level, this approach may imply asking the user for an excessive number of values. In order to avoid this problem, we propose that the abstraction of an item set be used in order to define the threshold as follows: for an item set I and an abstraction function A, the support threshold is defined as

$$\mathrm{threshold}_I = \mathrm{threshold_{SUP}} + (1 - \mathrm{threshold_{SUP}}) \times A(I) \tag{17}$$

where

- $\mathrm{threshold_{SUP}}$ is the support threshold established by the user for the basic levels;
- $A(I)$ is the abstraction of the item set I.

TABLE III
ALGORITHM TO OBTAIN FREQUENT ITEM SETS

Algorithm: $CalculateFrequentItemSets(threshold_{SUP}, C)$

- Input:
  - $threshold_{SUP}$: support threshold to consider an itemset as frequent
  - C: DataCube
- Output:
  - Set of frequent itemsets (C_F)

1) $C_F \leftarrow \emptyset$
2) $k \leftarrow 0$
3) Do
   a) $k \leftarrow k + 1$
   b) If $k = 1$ then
      i) $C_1 \leftarrow \bigcup_{\forall D_i \in C} l_{i\perp}$
   c) else
      i) $C_k \leftarrow$ Generate candidates using C_{Fk-1}
   d) $C_{Fk} \leftarrow \emptyset$
   e) While $C_k \neq \emptyset$ do
      i) $I \leftarrow$ First element of C_k
      ii) $C_k \leftarrow C_k - \{I\}$
      iii) $sop_I \leftarrow$ calculate the support of I
      iv) If $sop_I \geq threshold_{SUP} + (1 - threshold_{SUP}) \times A(I)$ then
         A) $C_{Fk} \leftarrow C_{Fk} \bigcup \{I\}$
      v) else
         A) $C_k \leftarrow C_k \bigcup Generalize(I)$
   f) End while
   g) If $k \neq 1$ then
      i) $C_F \leftarrow C_F \bigcup C_{Fk}$
4) While $C_{Fk} \neq \emptyset \wedge k <$ No. of dimensions
5) Return C_F

Once the process has all the frequent one-item sets, it applies an *a priori* strategy to obtain frequent item sets with more elements: to calculate the frequent k-item sets, it considers as candidates all the k-element sets that can be built using the frequent $(k-1)$-item sets, as long as all their subsets are frequent. In our case, the item sets must be defined using elements at different dimensions (we look for inter-dimensional relations).

The candidate k-item sets are considered frequent if their support is greater than the support threshold corresponding to their abstraction (using the previous formulation). As in the one-item sets case, if a set is not frequent, the algorithm considers as new candidates all the possible generalized item sets defined using elements at parent levels which group the elements of the set (see Fig. 7). The pseudo-code of the process is shown in Table III.

From the frequent item sets, the algorithm builds association rules using the same *a priori* method (considering a certainty factor threshold $\mathrm{threshold}_{CF}$ instead of a threshold over the rule consistency).

B. Generalization Process

At the end of the previous phase, the algorithm obtains a rule set, trying to represent as much information as possible about the DataCube. The method then tries to reduce the complexity of this set. The method must deal with the factors we have identified: the number of rules and the abstraction.

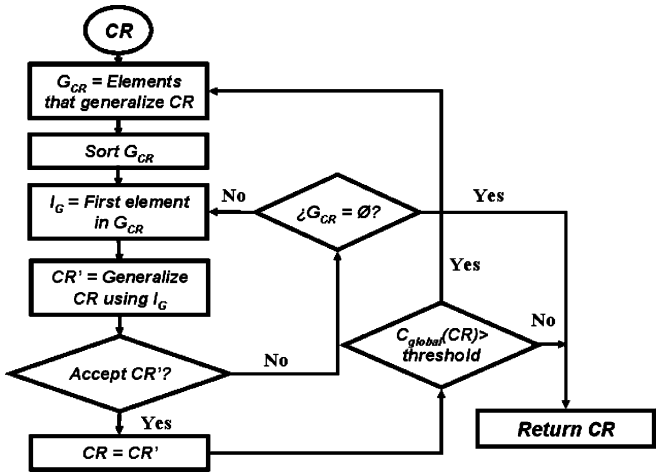


Fig. 8. Generalization process.

TABLE IV
LOSS-LESS GENERALIZATION ALGORITHM

Algorithm: $Generalize_{WL}(CR, threshold, \beta)$

- Input:
  - CR: Rule set
  - $threshold$: value in [0,1] with the complexity threshold
  - β: value in $[0, +\infty]$ with the penalty for failed generalization elements
- Output:
  - Rule set

1) While $Complexity(CR) > threshold$ do
   a) $G_{CR} \leftarrow$ Set of items than generalize CR order using β
   b) Do
      i) If $G_{CR} \leftarrow \emptyset$ then
         A) Return CR
      ii) $I_G \leftarrow$ first element in G_R
      iii) $G_{CR} \leftarrow G_{CR} - \{I_G\}$
      iv) $CR' \leftarrow$ generalize CR using I_G
   c) While $(Quality(CR') < Quality(CR)) \vee (Complexity(CR') > Complexity(CR))$
   d) $CR \leftarrow CR'$
2) End while
3) Return CR

The method applies a generalization process to reduce the complexity. This approach works directly on the abstraction and indirectly on the number of rules. We shall explain this by means of an example.

Let us suppose we have the following two rules.

1) If *[Patient is 13 years old]*, then *[Severity is low]*.
2) If *[Patient is 20 years old]*, then *[Severity is low]*.

We can generalize both antecedents, replacing *13 years old* and *20 years old* with the value *Young* that groups both elements. The abstraction of the rules will increase because new rules are defined using a higher level concept. However, the number of rules also decreases because both rules will be translated into the same one as in the following generalization:

If *[Patient is Young]*, then *[Severity is low]*.

In view of this, the generalization process is expected to reduce the complexity due to the number of rules and abstraction. This process will be applied until the complexity of the result is below a threshold established by the user $(\text{threshold}_{\text{Complexity}})$ without disregarding the loss of quality.

The generalization process has two steps. First, it tries to reduce the complexity through generalization but without allowing loss of quality. Then, if the method does not obtain a result set below the threshold, it applies a generalization allowing the decrease of quality.

1) Loss-Less Generalization: This first approach applies an iterative generalization but only accepts a new rule set if the quality of the new set is greater than or equal to the previous one. The scheme of the process is shown in Fig. 8.

The first step in the process is to find the elements that generalize the rule set (CR). The method looks at each item in each rule and obtains the elements in the DataCube which group them with a kinship relationship which is greater than 0 $(\mu_{ij} > 0)$. Under these circumstances, the method only looks for generalization elements at parent levels which are directly connected to the considered item level.

One element must then be chosen to generalize the rule set. In order to select the element, all the items are sorted using a heuristic: an element that generalizes more elements would be better if it is supposed to greatly increase the abstraction of the result. Then, the method selects the first element. If the method generalizes the rule set and obtains a new set with an unacceptable quality (it is lower than the previous one), then it could be very expensive in the sense that the method had to recalculate the quality of all the generalized rules. In order to sort the elements, the method therefore takes into account the number of times an item has been used unsuccessfully in the generalization process.

In this way, the weight of an item I will be calculated as

$$\text{Weight}_I = \frac{N_{RG}}{N_F \times \beta + 1} \tag{18}$$

where N_{RG} represents the number of rules that item I generalizes, N_F the times that I has been used and the result set was not accepted, and $\beta \in [0, \infty)$ measures the penalty for each failed generalization. Taking this into account, the method decreasingly sorts the elements according to their weights.

Once we have the generalized rule set (CR'), we accept it if the complexity has decreased and the quality has not decreased

$$C_{\text{global}}(CR') < C_{\text{global}}(CR) \wedge Q(CR') \geq Q(CR). \tag{19}$$

If the new set satisfies the condition, this set becomes the new result, and if the complexity is above the threshold, the entire process is repeated. If the set is not accepted, the process takes the next element that generalizes the rule set and the process is repeated.

The process finishes if the obtained rule set satisfies the complexity threshold or there are no elements to generalize. The pseudo-code is shown in Table IV.

2) Lossy Generalization: If the previous process fails to obtain a rule set with a complexity below the threshold, then we apply another generalization process but allowing quality loss. The general process is the same as the one shown in Fig. 8, but we change the new set acceptance criteria. In this case, for a new rule set to be accepted, it must satisfy the following two constraints.

1) The process compares the reduced complexity and the quality loss to decide if the generalization is good enough to accept the new rule set. The condition can be written as

$$\text{QualityLoss} < \gamma \times \text{ComplexityLoss} \tag{20}$$

where $\gamma \in [0, +\infty)$ and establishes when the complexity reduction is good enough compared to the quality lost.

TABLE V
LOSSY GENERALIZATION ALGORITHM

Algorithm: $Generalize_L(CR, threshold, \beta, \delta, \gamma)$

- Input:
  - CR: Rule set to generalize
  - $threshold$: value in [0,1] for the complexity threshold
  - β: value in $[0, +\infty]$ with the penalty for failed generalization elements
  - δ: value in [0,1] with the quality threshold to preserve
  - γ: value in [0,1] indicating the quality loss allowed to accept a new rule set
- Output:
  - Rule set

1) $BestQuality \leftarrow Quality(CR)$
2) While $Complexity(CR) > threshold$ do
   a) $G_{CR} \leftarrow$ Set of item than generalize CR ordered using β
   b) Do
      i) If $G_{CR} \leftarrow \emptyset$ then
         A) Return CR
      ii) $I_G \leftarrow$ first element in G_R
      iii) $G_{CR} \leftarrow G_{CR} - \{I_G\}$
      iv) $CR' \leftarrow$ generalize CR using I_G
      v) $QualityLoss \leftarrow \frac{Quality(CR) - Quality(CR')}{Quality(CR)}$
      vi) $ComplexityLoss \leftarrow \frac{Complexity(CR) - Complexity(CR')}{Complexity(CR)}$
   c) While $(QualityLoss > \gamma \times ComplexityLoss) \vee (Quality(CR') < \delta \times BestQuality)$
   d) $CR \leftarrow CR'$
   e) $BestQuality \leftarrow max\{BestQuality, Quality(CR)\}$
3) End while
4) Return CR

TABLE VI
COGARE ALGORITHM

Algorithm:
$COGARE(C, threshold_{Complexity}, threshold_{SUP}, threshold_{CF}, \beta, \delta, \gamma)$

- Input:
  - C: DataCube to apply the method
  - $threshold_{Complexity}$: value in [0,1] with the complexity threshold
  - $threshold_{SUP}$: value in [0,1] with the support threshold to accept frequent itemsets
  - $threshold_{CF}$: value in [-1,1] with the threshold to accept a rule
  - β: value in $[0, +\infty]$ with the penalty for failed generalization elements
  - δ: value in [0,1] with the quality threshold to preserve
  - γ: value in [0,1] indicating the quality loss allowed to accept a new rule set
- Output:
  - Rule set

1) $C_F \leftarrow CalculateFrequentItemSets(threshold_{SUP}, C)$
2) $CR \leftarrow AssociationRules(C_F, threshold_{CF})$
3) If $Complexity(CR) > threshold_{Complexity}$ then
   a) $CR \leftarrow Generalize_{WL}(CR, threshold_{Complexity}, \beta)$
4) If $Complexity(CR) > threshold_{Complexity}$ then
   a) $CR \leftarrow Generalize_L(CR, threshold_{Complexity}, \beta, \delta, \gamma)$
5) Return CR

2) In any case, the generalized rule set will not be accepted if its quality is below a threshold based on the best quality obtained throughout the entire process

$$Q(CR') \geq \delta \times \text{BestQuality} \tag{21}$$

where $\delta \in [0, 1]$. BestQuality will be at least the quality of the first rule set generated, but if when applying the loss-less generalization we obtained a higher quality rule set and it is accepted, then this new quality will be used.

If we set $\gamma = 0$ or $\delta = 1$, then quality loss is not allowed, so the process performs in exactly the same way as the lossless generalization. The pseudo-code of the process is presented in Table V.

Algorithm: In Table VI, the main function of COGARE is presented. Let us comment on all the parameters needed by the method.

- C: DataCube to apply the method.
- $\text{threshold}_{\text{Complexity}}$: Value in $[0, 1]$ with the complexity threshold.
- $\text{threshold}_{\text{SUP}}$: Value in $[0, 1]$ with the support threshold to accept frequent item sets. This value will be used for items at base levels. For items at other levels, the support threshold is calculated according to their abstraction as shown before.
- threshold_{CF}: Value in $[-1, 1]$ with the threshold to accept a rule.
- β: Value in $[0, +\infty]$ with the penalty for failed generalization elements. This value will have more influence on the time taken by the algorithm than on the quality of the results.
- δ: Value in $[0, 1]$ with the quality threshold to preserve.
- γ: Value in $[0, 1]$ indicating the quality loss allowed in order to accept a new rule set.

In Section VII-C1, we present a set of parameters obtained empirically that produce good results for three different domains. This set can be considered as the default values for the parameters.

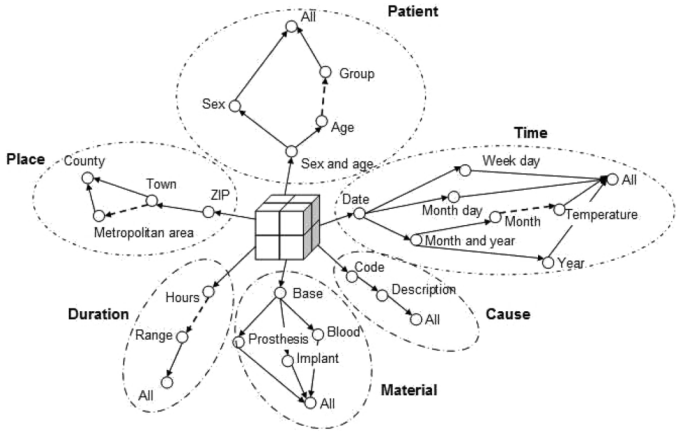


Fig. 9. Multidimensional schema over medical data.

VII. EXPERIMENTS

In this section, we will present the experimentation we have carried out in order to test the algorithm introduced in previous sections: first, the DataCubes used to extract rules are described, and then the support computation for fuzzy DataCubes is introduced.

We propose two experiments to test COGARE. We will start by focusing on the generalization process in order to test the complexity reduction capabilities of COGARE. We will then compare our algorithm with other methods. In this latter case, the obtained rule sets and the time taken are studied. The results of each of these two experiments are presented in Section VII-C and VII-D, respectively.

A. DataCubes

We have used DataCubes defined over three different domains: medical, financial, and census data. For each domain, we have defined two multidimensional schemata that model the same information from both crisp and fuzzy perspectives. The reason for this choice is to test whether the fuzzy approach helps reduce complexity.

Figs. 9–11 show the three multidimensional schemata. Fuzzy relations are represented by means of a dotted line connecting two levels. Crisp schemata are defined in the same way, translating the fuzzy relations into crisp ones (an element would be grouped by a value in the parent level—the one with the greatest kinship relationship value in the fuzzy case).

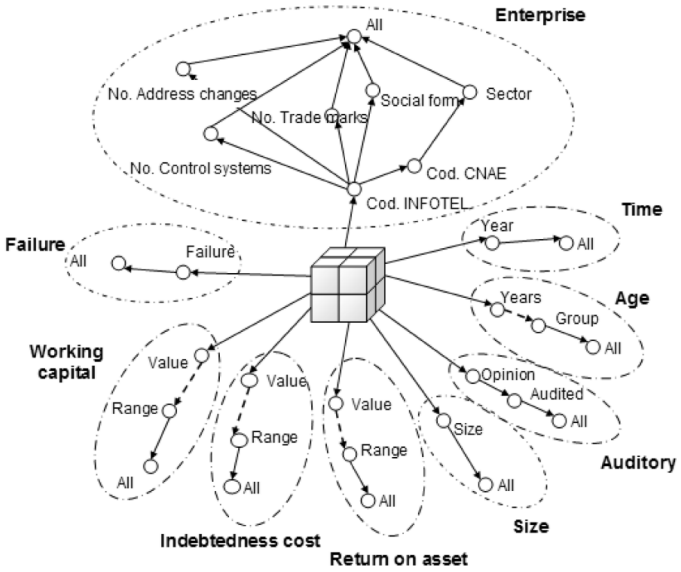


Fig. 10. Multidimensional schema over financial data.

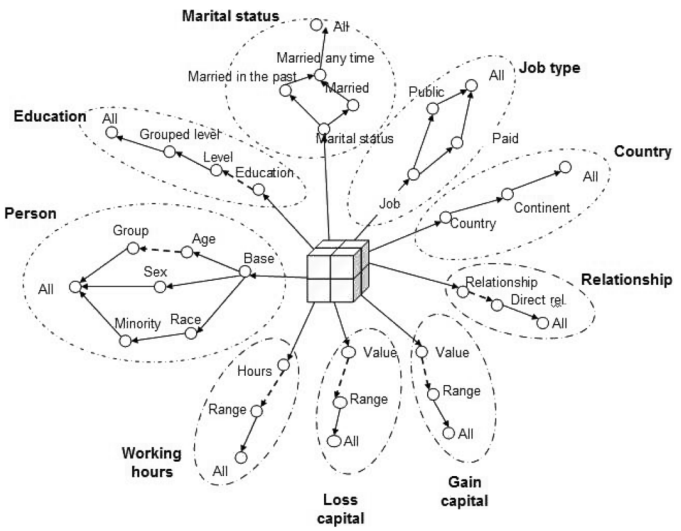


Fig. 11. Multidimensional schema over census data.

We will briefly explain the structures of the multidimensional schemata in the following.

1) C_{Medical}: This schema is defined over data collected for nonpostponed operations which were carried out in hospitals in Granada between 2002 and 2004. For the facts, we only consider the data when the patients are from Granada. There are 50 185 facts with one variable (*amount*) and six dimensions. Let us briefly explain each one.

Dimensions:

- *Patient*: In this dimension, we model patient data. The most detailed levels consider the different combinations of sex and age of each patient (the base level, therefore, has 2 sexes for 101 possible ages, totalling 202 values). Over this level, we group the patients according to their sex (level *sex*) and age (level *age*). Over this last one, we group the values more naturally for user (level *group*), and so we define what we can consider to be *young*, *adult*, and *old* patients using linguistic terms over the concrete values. The definition of these terms is the same as that shown in Fig. 2. The last level groups all the values so we have called it *all* with a single value (*all*). The structure of the dimension is as follows:

$$\text{Patient} = (\{\text{Sex and age, Sex, Age, Group, All}\}, \leq_{\text{Patient}}, \text{Sex and age,All}).$$

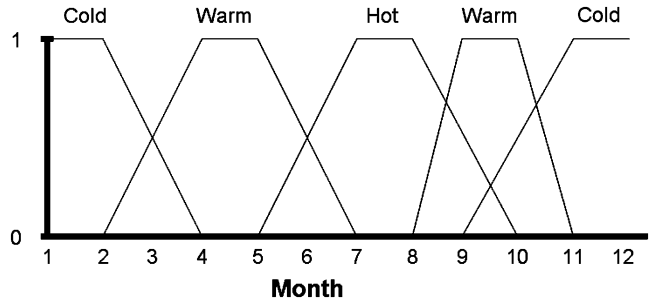


Fig. 12. Definition of level *Temperature* in dimension *Time* for C_{Medical}.

- *Time*: In this dimension, we consider the date when the operations took place. Over this level, we have defined a normal hierarchy over dates: weekday, month day, month, month and year, and year. The level *Temperature* represents information about the average temperature of each month in Granada using the values *cold*, *warm*, and *hot* to group the values. The relationships between the month and the temperature are not crisp because the user normally considers these concepts with imprecision. The definition of the relationships are shown in Fig. 12.
  The structure of the dimensions is as follows:

$$\text{Time} = (\{\text{Date, Weekday, Month day, month and year, Temperature, Year, All}\}, \leq_{\text{Time}}, \text{Date,All}).$$

- *Place*: This dimension stores information about where the patients live. Since the definition of the metropolitan area of Granada is not clear, we have used a fuzzy relation to establish the relationship between this level and the towns. The structure of the dimension is

$$\text{Place} = (\{\text{ZIP, Town, County, Metropolitan area, All}\}, \leq_{\text{Place}}, \text{ZIP,All}).$$

- *Duration*: We also consider the amount of time each operation lasted. The level *Range* groups this information according to three categories: *normal*, *long*, and *very long* duration. These groups have been defined imprecisely as shown in Fig. 13.
  The structure of the dimension is

$$\text{Duration} = (\{\text{Hours, Range, All}\}, \leq_{\text{Duration}}, \text{Hours,All}).$$

- *Material*: We want to analyze whether any materials were required for the operations, i.e., blood, prothesis, implants. The dimension *Material* models this information and has the following structure:

$$\text{Material} = (\{\text{Base, Blood, Implant, Prothesis, All}\}, \leq_{\text{Material}}, \text{Base,All}).$$

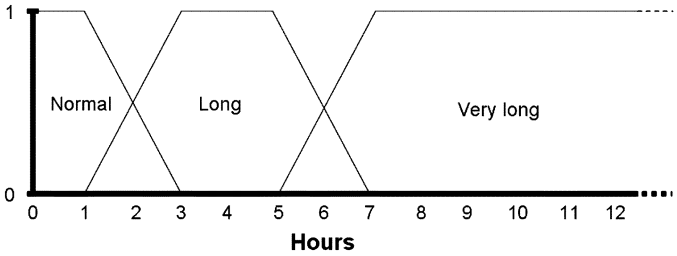


Fig. 13. Definition of level *Range* in dimension *Duration* for C_{Medical}.

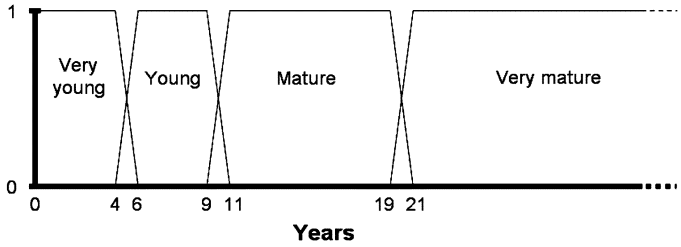


Fig. 14. Definition of level *Group* in dimension *Age* for $C_{\text{Financial}}$.

- *Cause*: In this dimension, we model the causes according to the codes established by the WHO. We consider the nine main categories as the base level and the description on them. The structure of the dimension is as follows:

$$\text{Cause} = (\{\text{Code, Description, All}\}, \leq_{\text{Cause}}, \text{Code,All}).$$

Measures: The only measure we consider is the number of operations with exactly the same values for all the dimensions we have built. This measure has been called the *amount*.

DataCube: The structure of the DataCube modeling the data is as follows:

$$C_{\text{Medical}} = \Big(\{\text{Duration, Time, Patient, Material, Place, Cause}\},\ \{\text{Amount}\} \bigcup \emptyset, \Omega, A\Big).$$

2) $C_{\text{Financial}}$: In this section, we present the structure of the DataCube built using the fuzzy multidimensional model presented. We have built a DataCube using the data obtained from Asexor about 872 companies from three sectors (service, commercial, and industrial) using the National Classification of Economic Activities (CNAE). In each sector, we differentiate between failed companies and those which have not in accordance with Spanish Law applied in 2001. We have considered three economic-financial variables: return on asset, working capital, and indebtedness cost, over the years 1998–2000.

Dimensions: We have defined five dimensions. In all of these, we have used the minimum and maximum operator as t-norm and t-conorm when calculating the extended kinship relationship. In the following sections, we will present the structure of each one.

- *Time*: The time dimension in this datacube is defined at a detail level of years. The structure of this dimension is

$$\text{Time} = (\{\text{Year, All}\}, \leq_{\text{Time}}, \text{Year,All})$$

where $\leq_{\text{Time}}$ defines the hierarchical relation as

$$\begin{aligned} \text{Year} &\leq_{\text{Time}} \text{Year} \\ \text{Year} &\leq_{\text{Time}} \text{All} \\ \text{All} &\leq_{\text{Time}} \text{All}. \end{aligned}$$

- *Failure*: We have mentioned that we study the companies differentiating between those which have failed and those which have not. This dimension gives this information. The basic level (*Fail*) only has two values, representing the failure of the company (value *Yes*) or not (value *No*), respectively. The following structure is associated to the dimension:

$$\text{Failure} = (\{\text{Fail, All}\}, \leq_{\text{Failure}}, \text{Fail,All}).$$

- *Company*: This dimension models information about a company. We have used the INFOTEL code as the base level. Over this, we have defined the CNAE codes to group the companies according to a detail sector classification. Over this, we define the sector level that groups the CNAE codes into service, commercial or industrial companies. The other levels represent the number of control systems used, the number of changes of social address, the number of trademarks obtained by the company, and the social form. This hierarchy translated into the fuzzy model proposed corresponds to the following structure:

$$\begin{aligned} \text{Company} = (\{&\text{INFOTEL, CNAE, No. control systems,} \\ &\text{No. changes, No. marks, social form,} \\ &\text{sector, all}\}, \leq_{\text{Company}}, \text{INFOTEL,All}) \end{aligned}$$

where $\leq_{\text{Company}}$ defines the hierarchical relation as shown in Fig. 10.

- *Age*: the base level of this dimension is the number of years that a company has been in operation. Over this level, we define another which groups this value in years to classify the companies depending on whether they are *very young*, *young*, *mature*, or *very mature*. This kind of concept is ill-defined, and they are normally defined using crisp intervals. This is not how people normally use these concepts and the previously mentioned edge problem *** may arise. The use of fuzzy logic in this situation is useful as it characterizes the concepts in a more intuitive way. The definition we have used is shown in Fig. 14.
  The structure of the dimension *Age* is:

$$\text{Age} = (\{\text{Years, Group, all}\}, \leq_{\text{Age}}, \text{Years,All}).$$

- *Return on Asset*, *Indebtness Cost*, and *Working Capital*: For the three dimensions over the economic financial variables, we have used the values observed in the data set for these variables to define the base level of the dimensions. Over these levels, we have defined another (*Range*) which groups the values into five categories to facilitate the analysis. For the user, the use of categorical values (e.g., average, low, high, etc.) is more intuitive than numeric values (e.g., a 6.51 return on asset). In order to avoid the edge

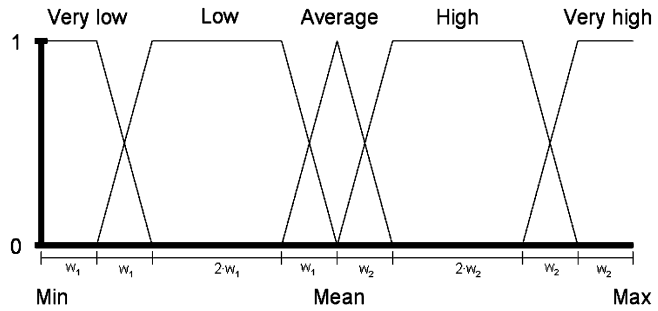


Fig. 15. Definition of ranges over the economic-financial variables for $C_{\text{Financial}}$.

problem, we fuzzify the intervals associated to each category.

We consider five categories according to the distance of the value to the mean of the variable and so we have used the categories: *very low*, *low*, *average*, *high*, and *very high*. The values which are very near to the mean will be in the *average* category, those not so near will be in the *low* or *high* categories if they are lower or higher than the mean, respectively, and so on. We have used the mean, maximum, and minimum value of the variable to define the categories. Each interval $[\text{minimum}, \text{mean}]$ and $[\text{mean}, \text{maximum}]$ has been divided into five intervals of width, w_1 for the first and w_2 for the other. The categories have then been defined as shown in Fig. 15. The structures of the three dimensions are, therefore, very similar.

- Return on asset $=$ $(\{\text{Values, Range, All}\}, \leq_{\text{RoA}}, \text{Values,All})$, where $\leq_{\text{RoA}}$ defines the hierarchical relation between the levels.
- Indebtness cost $=$ $(\{\text{Values, Range, All}\}, \leq_{\text{IC}}, \text{Values,All})$.
- Working capital $=$ $(\{\text{Values, Range, All}\}, \leq_{\text{WC}}, \text{Values,All})$.

Measures: We have used the return on asset and working capital. Both variables are considered as measures and dimensions because we want to analyze the relation between both (e.g., return on asset according to the working capital or vice versa). All the data is obtained from a reliable source so we assign a value 1 of α to all the facts.

DataCube: Finally, the structure of the DataCube is

$$C_{\text{Financial}} = \Big(\{\text{Time, Failure, Company, Return on asset, Working capital}\}, \{\text{return on asset, Working capital}\} \bigcup \emptyset, \Omega, A\Big)$$

where A is the relation that associates each fact with the corresponding values of the base level of the dimensions.

3) C_{Census}: This schema has been defined over 34 198 facts with one variable (*amount*) and nine dimensions using the data from adults in the Census database from the University of California.[1]

TABLE VII
RELATIONSHIP BETWEEN *EDUCATION* AND *LEVEL* IN C_{Census}

Level	Education
Basics	1/Preschool, 1/1st-4th, 1/5th-6th, 1/7th-8th, 0.8/9th
Medium	0.2/9th, 1/10th, 1/11th, 1/12th, 1/HS-grad, 0.2/Assoc-voc, 0.2/Some-college
High	0.8/Some-college, 0.8/Assoc-voc, 1/Bachelors, 1/Assoc-acdm, 0.2/Prof-school
Very high	0.8/Prof-school, 1/Doctorate, 1/Master

TABLE VIII
KINSHIP RELATIONSHIP BETWEEN RACES AND *YES* VALUE IN *MINORITY*

Race	$\mu_{\textbf{Yes,Race}}$
Amer-Indian-Eskimo	1
Asian-Pac-Islander	0.7
Black	0.5
Other	1
White	0

Dimensions:

- *Marital Status*: This dimension stores information about the individual's marital status. We consider different aspects about this topic to build the hierarchy as shown. The structure of the dimension is as follows:

$$\text{Marital status} = (\{\text{Marital status, Married, Married in the past, Married at any time, All}\}, \leq_{\text{MT}}, \text{Marital status,All}).$$

- *Education*: We also consider the level of education. We have grouped the values according to four categories: *basic*, *medium*, *high*, and *very high*. The relationships are defined imprecisely because we usually manage these concepts with imprecise borders between them. Table VII collects the values for the relationships.

  The level *Grouped level* groups these four categories into *normal* (values *basic* and *medium*) and *high* (*high* and *very high*). The dimension has the following structure:

$$\text{Education} = (\{\text{Education, Level, Grouped level, All}\}, \leq_{\text{Education}}, \text{Education,All}).$$

- *Person*: We consider the combination of the individual's age, sex, and race as the base level. Over this level, we group the values according to these three variables. The ages are grouped in the same way as in the medical DataCube. The races have been categorized depending on whether they can be considered as minorities. The *black* race represents 10% of the population whereas others such as *amer-indian-eskimo* only 3%, and each of these may be considered a minority at different degrees. Table VIII shows the kinship relationship.
- *Working Hours*: We also consider the working hours. The values have been grouped according to how the number is considered as *very low*, *low*, *normal*, *high*, or *very high*. In order to build this classification, we have used fuzzy

[1][Online]. Available: http://kdd.ics.uci.edu.

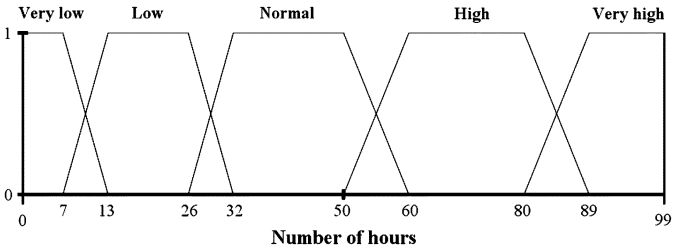


Fig. 16. Definition of ranges over *Hours* for C_{Census}.

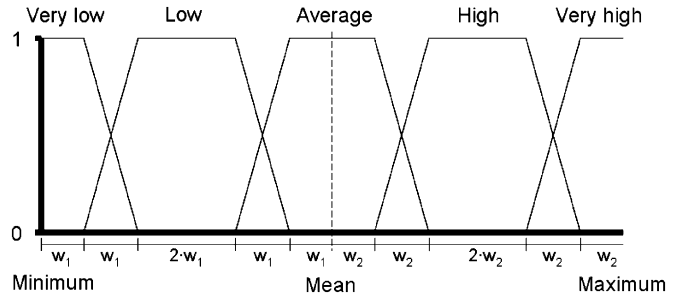


Fig. 17. Definition of ranges over *Loss capital* and *Gain capital* for C_{Census}.

intervals because the borders between them are not clear. Fig. 16 shows the structures of the intervals.

- *Loss Capital* and *Gain Capital*: These two dimensions represent the loss capital and the gain capital. The values have been categorized following a similar approach as for the economic-financial variables in the $C_{\text{Financial}}$ DataCube but changing the middle value (see Fig. 17). The structure is, therefore, similar to those proposed for these dimensions in $C_{\text{Financial}}$.
- *Relationship*: Over the values of the base level (*husband*, *wife*, *own-child*, *other-relative*, and *not-in-family*) we have defined a level to classify the values according to the degree of relationship. We have considered the values as direct or not but this classification is not always clear. Table IX shows the defined kinship relationship.
- *Country*: Another variable to classify the measures is the individual's country. Countries are classified by continent. The dimension has the following structure:

$$\text{Country} = (\{\text{Country, Continent, All}\}, \leq_{\text{Country}}, \text{Country,All}).$$

- *Job Type*: In this dimension, we consider the job type. The values are grouped according to whether the job types are paid (level *Paid*) and if they correspond to the civil service (level *Public*).

Measures: We only consider the number of transactions with the same values for all the considered dimensions (*amount*) as measures.

Datacube: Finally, the structure of the DataCube is

$$C_{\text{Census}} = \Big(\{\text{Marital status, Person, Working hours, Education, Loss capital, Gain capital, Relationship, Country, Job type}\}, \{\text{amount}\} \bigcup \emptyset, \Omega, A\Big).$$

TABLE IX
KINSHIP RELATIONSHIP BETWEEN RELATIONSHIPS AND *YES* VALUE IN *DIRECT* RELATIONSHIP

Relationship	$\mu_{\text{Yes,Relationship}}$
Husband	1
Wife	1
Unmarried	0
Own-child	0.5
Other-relative	0.25
Not-in-family	0

B. Support Calculation

In all of the DataCubes presented, there is a fact that stores the number of elements in the original data sharing the coordinates (e.g., in the medical DataCube, the fact *amount* represents the number of patients of the same sex and age, with the same ZIP code, undergoing the same operation, lasting the same amount of time on the same date). When calculating the support, we must, therefore, consider the number of transactions that each fact represents (e.g., in the medical data if a fact *amount* has the value 5, this means that these coordinates represent five operations).

In crisp DataCubes, this only involves changing one aggregation operator: instead of the count operator, we will use the sum aggregation operator. In fuzzy ones, we also have the sum operator, but it returns fuzzy sets as the result (see [9] for further details). Concrete values are needed to apply the quality measures and the support. In this case, we will use the same approach proposed in [45]. The authors propose the use of quantified sentences of the type: "Q of F are G," where F and G are fuzzy sets and Q is a linguistic quantifier for calculating the support. In order to evaluate the sentence and obtain the support, the GD quantifier [48] is used

$$GD_Q\left(\frac{G}{F}\right) = \sum_{\alpha_i \in \Delta(G/F)} (\alpha_i - \alpha_{i+1}) Q\left(\frac{|(G \bigcap F)_{\alpha_i}|}{|F_{\alpha_i}|}\right) \tag{22}$$

where $\Delta(G/F) = \Lambda(G \bigcap F) \bigcup \Lambda(F)$, $\Lambda(F)$ is the level set of F, and $\Delta(G/F) = \{\alpha_1, \ldots, \alpha_p\}$ with $a_i > a_{i+1}$ for every $i \in \{1, \ldots, p\}$.

If we consider the quantifier $Q(x) = x$, it can be proven that it behaves coherently in the crisp case (see [45] for more details). There are the following two reasons for using this approach.

1) The quantifier GD can be adapted to work over the result of the aggregation operators for the fuzzy multidimensional model.
2) The support calculation is efficient.

We now have all the elements needed to run the experiments.

C. Complexity Reduction

In this section, we will study the reduction capabilities of the second stage of COGARE: the generalization process. We want to test whether this step is useful to reduce the complexity of the result obtained in the generation process. In the experiment, we look at the evolution of the quality, complexity, and number of rules of the result set during the method evolution.

1) Parameters: Let us discuss the following parameters used to run COGARE on each domain.

TABLE X
EXAMPLE OF ASSOCIATION RULES USING FUZZY CONCEPTS

$C_{Financial}$
if *[Size is Small]* **and** *[Working capital is Low]* **then** *[Not Fail]* **support:** *0.207* **CF:** *0.524*
if *[Indebtedness cost is Very low]* **then** *[Not Fail]* **support:** *0.11* **CF:** *0.387*
if *[Size is Small]* **and** *[Age is Mature]* **then** *[Not Fail]* **support:** *0.332* **CF:** *0.377*
$C_{Medical}$
if *[Cause is Respiratory disease]* **then** *[Duration is Normal]* **support:** *0.202* **CF:** *0.793*
if *[Material is No Blood]* **and** *[Patient is Adult]* **then** *[Duration is Normal]* **support:** *0.557* **CF:** *0.443*
C_{Census}
if *[Continent is Noth America]* **and** *[Loss capital is Very low]* **and** *[Relationship is Direct]* **then** *[Race is not Minority]* **support:** *0.437* **CF:** *0.835*
if *[Marital status is not Married any time]* **then** *[Gain capital is Very low]* **support:** *0.316* **CF:** *0.494*

- $\text{threshold}_{\text{Complexity}}$: We have set these parameters to 0 because we want to test the complexity reduction possibilities of COGARE.
- $\text{threshold}_{\text{SUP}}$: We have used the following different thresholds for each schema (obtained empirically):
  — C_{Medical}: 0.01;
  — $C_{\text{Financial}}$: 0.01;
  — C_{Census}: 0.15.
- $\text{threshold}_{\text{CF}}$: In all the cases, this value is set to 0.2. This value is low for the measure, but we want to obtain a high number of rules to test the reduction capability of the algorithm.
- β: We have set a value of 10 (a high penalty for values already used without success).
- γ: The user normally prefers to lose quality if the method obtains a good complexity reduction. We propose a value of $\gamma = 1.2$.
- δ: The global threshold for quality reduction is 0.6. The method will never accept a new rule set if the quality is less than 60% of the best quality obtained throughout the process.
- Quality Measure: In the rule generation process, we have mentioned that the quality measure used is the certainty factor. For the generalization process, we will allow different quality measures to be selected. We believe that once we have the rules, we can focus on certain aspects of the relations between items when applying generalization. Under this circumstance, we will test the algorithm in the experiments using all the quality measures presented in Section V-A (considering the accuracy used in IMAFO as another measure: Q_{AC}). We will then run the method using 14 different quality measures to test whether the complexity reduction is quality measure dependent.

We will, therefore, consider 84 results: 3 domains, defining two DataCubes (one crisp and another fuzzy) over each domain, and 14 different quality measures.

2) Results: Examples of association rules obtained using fuzzy concepts are shown in Table X.

An example evolution of the quality and complexity (together with the reduction of the number of rules) is shown in Fig. 18. Fig. 18 shows that the loss-less generalization process produces an improvement in the quality of the rule set, while complexity is also reduced. The following iterations in the generalization process also reduce both the quality and the complexity of the set. However, in this latter case, the quality of the result never reaches the established threshold (60% of the best rule set).

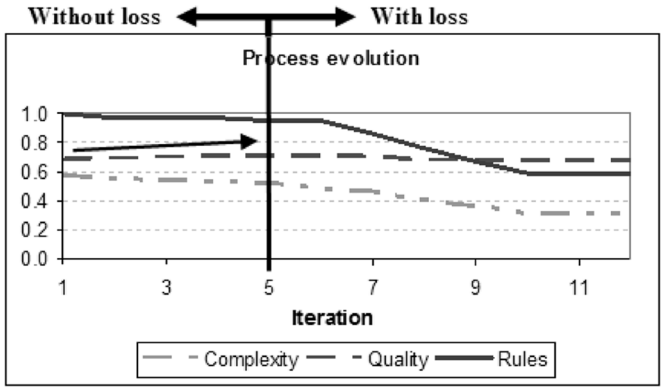


Fig. 18. Example of the process evolution, using *Cons* as the quality measure over the fuzzy DataCube C_{Census}. The *Rules* value represents the percentage over the initial number of rules generated.

TABLE XI
COMPARATIVE RESULTS (* RESULTS WITH AVERAGE STATISTICALLY DIFFERENT VALUES)

Measure	Crisp	Fuzzy	Significant
Lossless generalization			
Average complexity	0.2293152	0.2218327	0.1201
Average quality	0.6172338	0.6081381	0.2948
Lossy generalization			
Average complexity	0.1505985	0.1429530	0.0645
Average quality	0.5458529	0.5343061	0.096
Global improvement			
No. of rules reduction	37.67%	38.57%	0.5799
Average complexity reduction	39.86%	42.73%	0.0794
Average quality reduction	5.42%	6.59%	0.7213

Comparative results for $C_{Medical}$ crisp and fuzzy

Measure	Crisp	Fuzzy	Significant
Lossless generalization			
Average complexity	0.7363038	0.7000179	0.0053
Average quality	0.5129792	0.5137552	0.3564
Lossy generalization			
Average complexity	0.5948773	0.5610449	**0.00001***
Average quality	0.4561585	0.460702	0.3534
Global improvement			
No. of rules reduction	43.31%	49.74%	**0.00001***
Average complexity reduction	26.26%	30.45%	**0.00001***
Average quality reduction	−4.96%	−7.06%	0.2152

Comparative results for $C_{Financial}$ crisp and fuzzy

Measure	Crisp	Fuzzy	Significant
Lossless generalization			
Average complexity	0.5220438	0.5325539	**0.0018***
Average quality	0.7019994	0.7163766	**0.00001***
Lossy generalization			
Average complexity	0.3406067	0.3188788	**0.0149***
Average quality	0.6306086	0.6522445	**0.0417***
Global improvement			
No. of rules reduction	36.58%	41.21%	**0.011***
Average complexity reduction	39.83%	43.85%	**0.0116***
Average quality reduction	6.93%	7.12%	0.9329

Comparative results for C_{Census} crisp and fuzzy

Table XI contains a summary of the results for each considered domain (medical, financial, and census data) for both fuzzy and crisp approaches. The studied aspects are the quality, complexity, and number of rules in two stages of the algorithm: after the loss-less generalization and the final result (after the

lossy generalization), and also the global improvement. The last column (entitled *Significant*) represents the degree to which the average values can be considered equal using a Student average differences test (considering pair observations and two queues).

By way of summary of the results, we can see that the global average complexity reduction is 39.01% (with a cardinality reduction of 43.17%) and quality reduction that never exceeds 10% of the first rule set generated. In this last aspect, in the case of the $C_{\text{Financial}}$ DataCube, the generalization process has, on average, produced result sets with a greater quality than the initial one.

Comparing both approaches (fuzzy and crisp), the global improvement aspects which are statistically different (Student test with a value under 0.05) are *average complexity reduction* and *number of rules reduction* for $C_{\text{Financial}}$ and C_{Census}. In both cases, the fuzzy approaches have obtained better results. In the case of C_{Medical}, there are no significant differences, nor when talking about the quality of the rule sets. In short, we can conclude that when there are significant differences, the fuzzy approaches have obtained less complex results with a quality which is comparable to that of crisp DataCubes, bearing in mind that the fuzzy DataCubes use concepts which are easier for the user to understand.

D. Other Methods

In this section, COGARE is compared with other association rule extraction methods proposed in the literature. The methods proposed by Srikant and Agrawal [16], Zhu [25], and Kaya and Alhajj [7] are used in the experimentation. Only the last one is considered to work over fuzzy DataCubes, but the fuzzy model proposed is simple so we need to adapt all the methods to support the fuzzy multidimensional model presented. In all the cases, we used the same support calculation.

1) Parameters: Since different support thresholds are considered over each DataCube to compare the methods, we can compare the evolution of the results according to the considered support. For each DataCube, we have used the following values:

- C_{Medical}: 0.05, 0.10, 0.15, 0.20, and 0.25;
- $C_{\text{Financial}}$: 0.05, 0.10, 0.15, 0.20, and 0.25;
- C_{Census}: 0.20, 0.25, 0.30, 0.35, and 0.40.

The following selected values for the other parameters are needed by each method.

Srikant and Agrawal: To accept a generated rule, the method considers a *Confidence* threshold and for this purpose we consider the value 0.5. As we have mentioned, the method uses an interesting measure to reduce redundant rules. In this process, the method needs a parameter (R) to define the *R-interesting* rules. In [16], the authors proposed the value 1.1 for this parameter and we make the same selection.

Zhu: This method is defined to work with an abstraction level on each dimension. As mentioned in [25], in order to consider different levels, we must execute the methods with all the level combinations. Only the inter-dimensional associations are considered.

The method uses the *Confidence* of the rules to accept them. We have used a value 0.5 as the threshold for this quality measure.

Kaya and Alhajj: While this method considers intra-dimensional and inter-dimensional associations, only the second type are considered in our experimentation. The method proposes the use of different support thresholds for each level. If we use a support threshold for each level, we need to define between 28 (for C_{Medical} and $C_{\text{Financial}}$) and 35 values (in the case of C_{Census}), and this may be difficult for the user. If we define a threshold depending on the depth of the level, we need four or five different values (for $C_{\text{Financial}}$ and C_{Census} in the first case, and C_{Medical} in the second). In this latter case, the problem is that it may be complex for the user to define the values bearing in mind that levels at different dimensions with the same depth may represent elements defined at different levels of granularity. In [7], the authors choose the values without explaining the method used to select them and so we prefer to use the same abstraction-based methodology presented in COGARE and the previously mentioned values for the *base levels*. For each item, we consider the abstraction to define the support threshold and as the authors propose, the minimum of these values as the threshold for an item set. As in the two previous methods, we consider a value 0.5 for the *Confidence* threshold.

Cogare: For our proposal, we consider the same parameters as those considered for the previous experiment.

2) Results: In this section, we will present the results of the experiment. The number of frequent item sets, the number of rules obtained and the time taken are the three analyzed variables. Let us discuss the results.[2] As we have mentioned before, most of the methods have two main stages: 1) the calculation of frequent item sets and 2) the generation of the rules using the calculated item sets. We compare both steps, beginning with the number of frequent item sets.

Number of Frequent Item Sets: We consider the total number of frequent item sets obtained that are used later for rule generation and so we do not consider the one-item sets. Although COGARE has been tested using different quality functions, this parameter does not affect the frequent item set calculation and therefore all the executions for the same support thresholds obtain the same number of item sets and are, therefore, represented by the same graph. In the case of Zhu's and Agrawal and Srikant's proposal, both use the same process to obtain the frequent item sets and so the results are represented together in the charts.

Table XII and Figs. 19–21 present the frequent item sets for $C_{\text{Financial}}$, C_{Medical}, and C_{Census}, respectively.

The methods proposed by Zhu and Agrawal and Srikant apply a direct adaptation of the frequent item set calculation of *a priori* so they obtain a very high number of these in the three DataCubes. The reason is simple and was presented in the introduction of this paper: once the methods obtain a frequent item set, all the ones that are a generalization (i.e., they include items at a higher level in the hierarchy which include these) are also frequent. Since these new frequent item sets will represent the same information at a different abstraction level, the set of frequent item sets will present a high redundancy.

In addition, the elements belonging to levels that are near the top of the hierarchy will group a high number of elements at the

[2]All the methods have been implemented using Java language and executed using Sun Java V.M. 1.5 in a bi-processor AMD Opteron 64 bits computer with 4 GB of RAM running under Linux O.S. version 2.6.11.

TABLE XII
NUMBER OF FREQUENT ITEM SETS

Method	Support				
	0.05	0.10	0.15	0.20	0.25
Zhu	7233	2283	1027	523	333
Srikant & Agrawal	7233	2283	1027	523	333
Kaya & Alhajj	505	266	153	79	49
COGARE	87	61	48	33	22

$C_{Medical}$

Method	Support				
	0.05	0.10	0.15	0.20	0.25
Zhu	4606	1251	484	234	135
Srikant & Agrawal	4606	1251	484	234	135
Kaya & Alhajj	395	122	56	26	20
COGARE	175	81	45	26	20

$C_{Financial}$

Method	Support				
	0.20	0.25	0.30	0.35	0.40
Zhu	12007	7122	3972	2240	1241
Srikant & Agrawal	12007	7122	3972	2240	1241
Kaya & Alhajj	3392	2229	1350	812	485
COGARE	938	654	456	303	232

C_{Census}

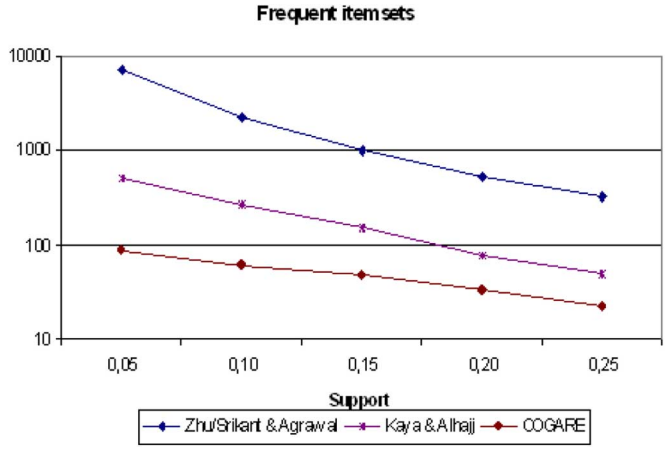


Fig. 19. Number of total item sets obtained by the methods for C_{Medical}.

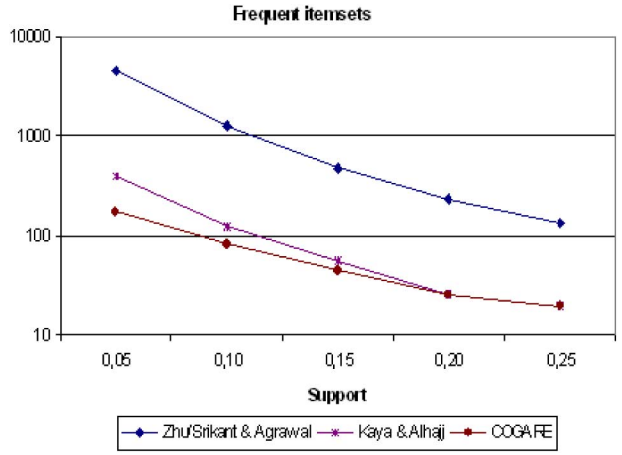


Fig. 20. Number of total item sets obtained by the methods for $C_{\mathrm{Financial}}$.

base levels, so their support will be high. Consequently, most of the items at these levels will be considered frequent.

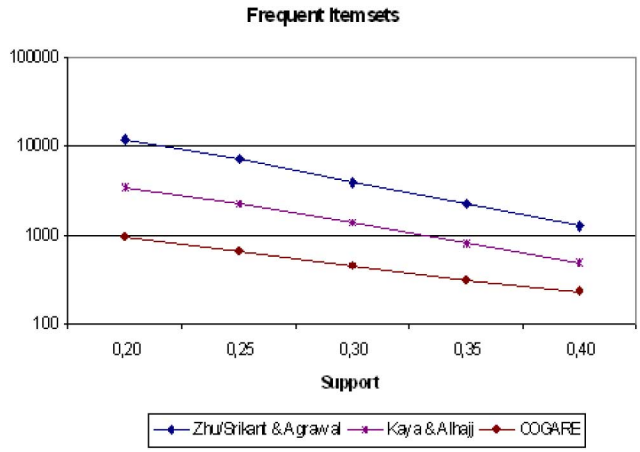


Fig. 21. Number of total item sets obtained by the methods for C_{Census}.

Kaya and Alhajj's method tries to solve these problems by considering different support thresholds for each level in the DataCubes. The result is a significant reduction of frequent item sets: 89.75% in $C_{\mathrm{Financial}}$; 87.84% in C_{Medical}; and 67.55% in C_{Census} on average. However, the problem of redundancy can appear. If an item at a higher level has a support over the threshold, it will be considered frequent regardless of whether a more specific item grouped by this was frequent or not. The process applied by COGARE manages this kind of imprecision and, therefore, obtains the smallest number of frequent item sets in all the experiments. Compared with Zhu's and Srikant and Agrawal's methods, the reduction is 92.33% in $C_{\mathrm{Financial}}$, 96.29% in C_{Medical}, and 89.5% in C_{Census} on average, and in the case of Kaya and Alhajj's method 27.24%, 71.56%, and 67.98%, respectively. If we bear in mind that the result of this step will be used for calculating rules, COGARE reduces the complexity of the final rule set from this first stage.

The final aspect which needs to be studied is the evolution of the number of frequent item sets as the support threshold changes. This will give us some idea of how sensitive the methods are to this parameter. In this way, a steep curve means that small variations in this value would result in a significant increase in frequent sets, and the more gradual the curve, the less sensitive it is to this value and the less complicated it is to fix the threshold for the user. Bearing in mind that the graph scale is logarithmic, the methods most sensitive to this value are those proposed by Zhu and Srikant and Agrawal, in such a way that adjusting the threshold value will be very important for controlling the number of frequent sets. Kaya and Alhajj's method presents less of a curve so it is less sensitive. In the $C_{\mathrm{Financial}}$ case, this method behaves in a similar way to COGARE for high values. For all other cases, COGARE presents a more gradual slope, i.e., less sensitivity to this parameter.

If we observe the domains, we can see that the most complex one (i.e., the one which has obtained the greatest number of frequent sets) in all cases was C_{Census}, followed by C_{Medical}, and then $C_{\mathrm{Financial}}$.

Number of Rules: Fig. 22 shows the number of rules obtained using COGARE with the different support thresholds for $C_{\mathrm{Financial}}$. In all the cases, the number of rules is very similar regardless of the quality function. In the case of the time taken

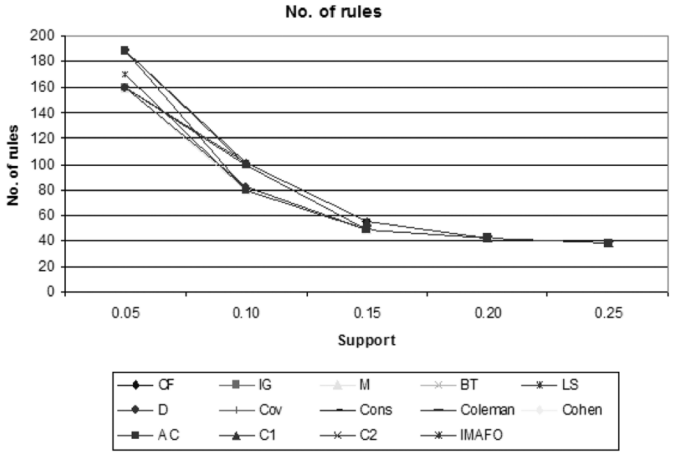


Fig. 22. Number of rules obtained by COGARE using different support thresholds and quality functions for $C_{\text{Financial}}$.

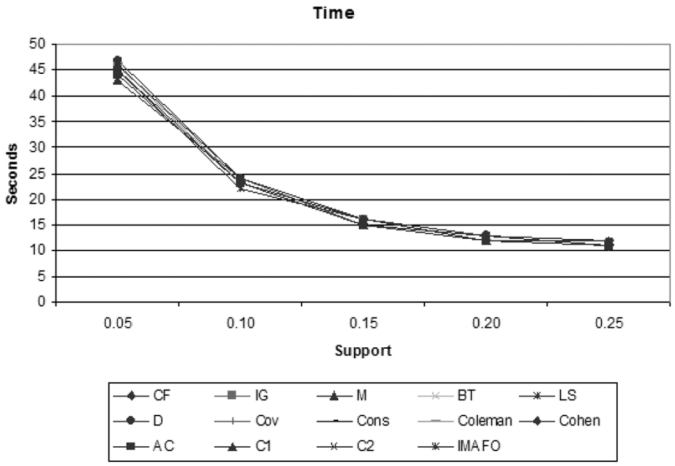


Fig. 23. Time spent by COGARE using different support thresholds and quality functions for $C_{\text{Financial}}$.

TABLE XIII
NUMBER OF RULES FOR THE THREE DOMAINS

Method	Support 0.05	0.10	0.15	0.20	0.25
Zhu	236199	9685	4444	2370	1398
Srikant & Agrawal	12071	4949	2586	1335	848
Kaya & Alhajj	1223	689	318	162	92
COGARE (avg.)	548.1	31.1	24.6	14.3	11.1

$C_{Medical}$

Method	Support 0.05	0.10	0.15	0.20	0.25
Zhu	17156	4660	1649	872	570
Srikant & Agrawal	8925	2673	1062	586	351
Kaya & Alhajj	854	264	102	58	54
COGARE (avg.)	174.6	87.4	50.3	42.0	38.0

$C_{Financial}$

Method	Support 0.20	0.25	0.30	0.35	0.40
Zhu	236199	124363	58856	30522	16141
Srikant & Agrawal	26506	15935	9206	5335	3184
Kaya & Alhajj	37613	21385	10367	6061	3038
COGARE (avg.)	548.4	216.0	89.1	31.0	19.6

C_{Census}

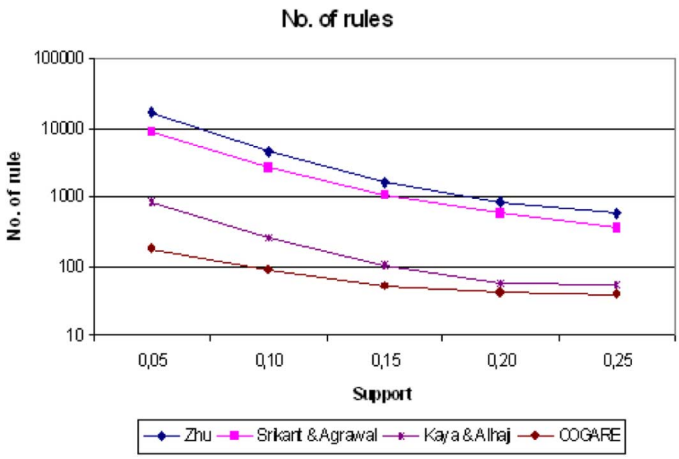


Fig. 24. Number of rules obtained by the methods using different support thresholds for $C_{\text{Financial}}$.

during the process (see Fig. 23) the values are also similar. In the other two domains, the behavior is similar. In order to reduce the complexity of the graphs, we, therefore, represent the average value obtained by the different executions of COGARE to compare with the remaining methods.

Table XIII and Figs. 24–26 show the number of rules obtained for $C_{\text{Financial}}$, C_{Medical}, and C_{Census} datacubes, respectively. The charts use logarithmic axes for the number of rules due to the large differences between the methods.

In all the cases, Zhu's method has obtained the highest number of rules. In Srikant and Agrawal's proposal (although obtaining the same number of frequent item sets), the pruning based on the interest of the rules reduces the final result considerably (depending on the domain the method reduces the number of rules to 10% of Zhu's result). However, the number of rules is too high to be easy enough for the user to understand.

Kaya and Alhajj's proposal uses a different interesting measure to prune the rule set. The use of this measure and the fact that the method obtains less frequent item sets explain the reduction of the number of rules compared with the previous methods. The results over C_{Census} are an exception. In this case, this method, although it works with less frequent item sets, obtains more rules than Srikant and Agrawal's proposal. Both interesting measures, therefore, behave in different ways depending on the domain.

Our proposal, COGARE, obtained the smallest number of frequent item sets, and in Section VII-C, we looked at the complexity reduction capacities, obtaining an average reduction of 43.17% of the rule set. It, therefore, obtains rule sets with the lowest cardinality in all the cases.

- Zhu's Method: COGARE obtained rules sets with a cardinality varying between 0.16% and 4.1% of the ones produced by this method.
- Srikant and Agrawal's Method: Cardinality varies between 1.12% and 6.5%.
- Kaya and Alhajj's Method: In this case, COGARE obtained rule sets with a cardinality in the interval 15% and 59% compared to the ones produced by this method.

As in Section VI-C, we will study the evolution of the number of rules in relation to the value of the support threshold. In this case, it is possible to distinguish between Zhu's and Srikant and Agrawal's method. Therefore, the measure of interest for

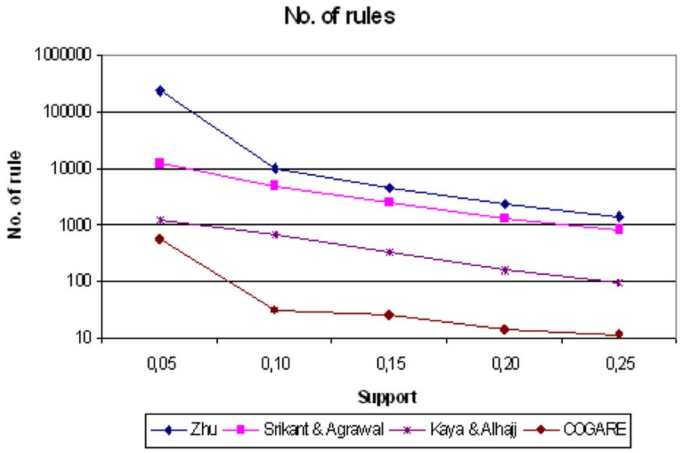


Fig. 25. Number of rules obtained by the methods using different support thresholds for C_{Medical}.

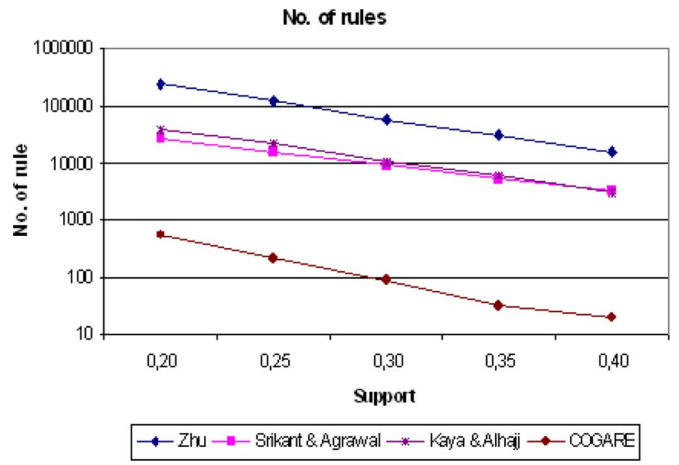


Fig. 26. Number of rules obtained by the methods using different support thresholds for C_{Census}.

the pruning slightly reduces the sensitivity of the method to the support value. In the case of frequent sets, Kaya and Alhajj's method has a smaller dependence than the previous ones. The difference between COGARE and other methods which was apparent in the frequent set calculation is further increased in the case of the rules, particularly in the case of C_{Census}. The generalization process which is applied, therefore, means that sensitivity is further reduced to the support threshold value.

The complexity of the domains which we previously observed remains the same.

Time: The last aspect analyzed is the time taken by each method to run over the DataCubes. Table XIV and Figs. 27–29 show the number of seconds taken for the executions. Zhu's method and Srikant and Agrawal's proposal are the most time-consuming processes, and the second method takes even longer since it requires the same amount of time as the first to generate the rules in addition to the time needed to prune the redundant rules. The fastest method is Kaya and Ahlajj's proposal, closely followed by COGARE.

For the evolution of the time in relation to the support it can be seen that the least sensitive method is the one proposed by Kaya and Alhajj. This is followed by our proposal, and then Zhu's and Srikant and Agrawal's proposals. The order of complexity of the domains remains the same.

TABLE XIV
TIME IN SECONDS FOR THE THREE DOMAINS

Method	Support				
	0.05	0.10	0.15	0.20	0.25
Zhu	22858	3597	1941	962	708
Srikant & Agrawal	12434	3792	1962	967	688
Kaya & Alhajj	1265	620	366	295	213
COGARE (avg.)	9598.3	637.4	512.0	414.3	311.1

$C_{Medical}$

Method	Support				
	0.05	0.10	0.15	0.20	0.25
Zhu	106	46	27	16	12
Srikant & Agrawal	1297	142	38	19	14
Kaya & Alhajj	14	8	5	4	3
COGARE (avg.)	45.4	23	15.6	12.3	11.7

$C_{Financial}$

Method	Support				
	0.20	0.25	0.30	0.35	0.40
Zhu	22858	12737	6801	2678	2183
Srikant & Agrawal	58063	21951	9569	4410	2468
Kaya & Alhajj	4814	3069	1915	1211	819
COGARE (avg.)	9590.4	3944.7	2055.8	1024.6	837.1

C_{Census}

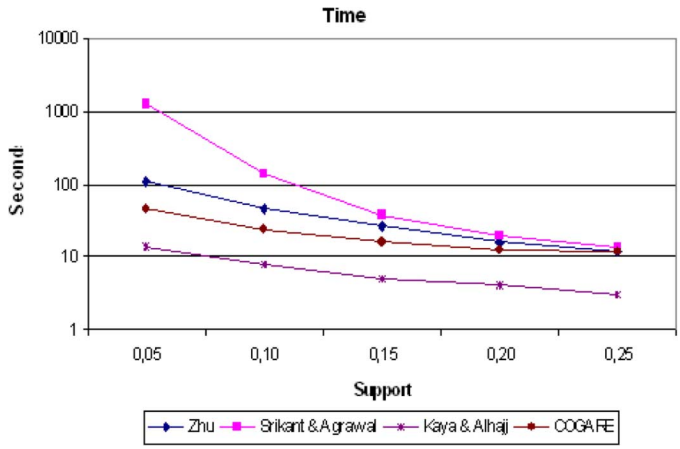


Fig. 27. Time spent by the methods using different support thresholds for $C_{\mathrm{Financial}}$.

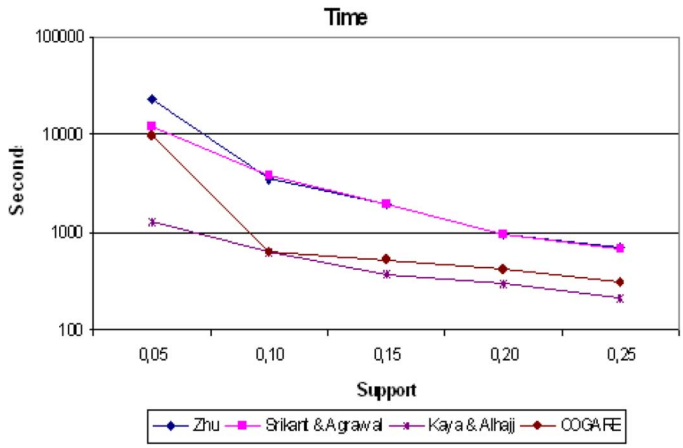


Fig. 28. Time spent by the methods using different support thresholds for C_{Medical}.

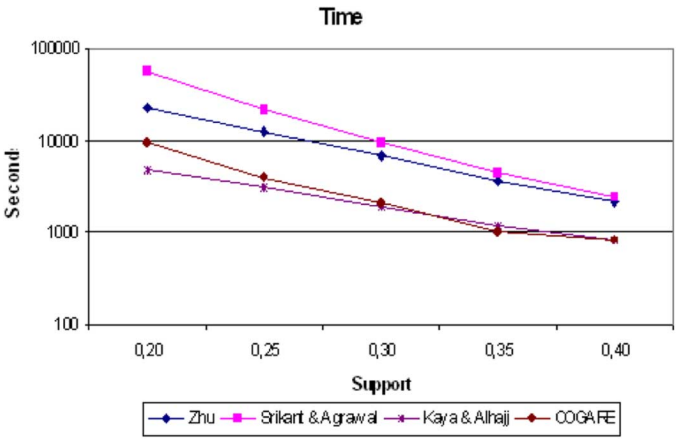


Fig. 29. Time spent by the methods using different support thresholds for C_{Census}.

VIII. Conclusion and Future Work

In this paper, we have presented COGARE: an algorithm to extract association rules over fuzzy DataCubes that can represent imprecise concepts and the ill-defined relations between them. It attempts to reduce the complexity of the results by using fuzzy concepts and the hierarchical relation in the DataCube. As far as the authors know, this is a novel approach in OLAM.

The proposed method uses a *bottom-up* approach which attempts to obtain inter-dimension relations at the most detailed level possible. Having identified two factors for the complexity (abstraction and cardinality), it tries to obtain a result which is below a complexity threshold established by the user. In order to achieve this, the algorithm uses the hierarchical relations between the elements at the dimensions.

COGARE has been tested over DataCubes which are defined using real data from three different domains (medical, financial, and census data), obtaining an average complexity reduction of 39%, with a quality loss of less than 10%. The method has also been compared with other association rule extraction methods and COGARE has always obtained less complex results. The number of rules has been significantly reduced, varying from 0.16% to 4.1% in the case of Zhu's method (which has always obtained the largest rule set), and between 15% and 59% in the case of Kaya and Alhajj's method (the smallest rule sets after COGARE). In addition, our proposal presents less sensitivity in the number of rules from the support threshold.

An interesting aspect to study is the set of heuristics for the generalization process and the proposal of alternative generalization approaches. The proposed process has the following two aspects which need to be improved.

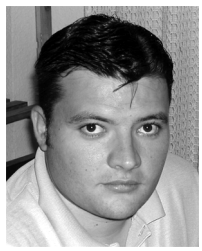
- In certain situations, the proposed heuristic applies the generalization over the rule set using the same items and fails to accept the new set, even if we consider a high penalization for these items. It, therefore, spends time on generalizations that involve a high number of rules without accepting the results.
- The process only considers a generalized rule set at each step; it can evolve to a local minimum. Another process that compares more than one set at each step or develops different generalizations at the same time would be interesting in this regard.

The next step is to develop other OLAM methods that work over the multidimensional model. The user may sometimes need not only to analyze the information in the DataCube but also to predict the behavior in the future. Therefore, classification techniques (e.g., classification trees and classifications rules) that manage imprecision would be an interesting topic to study.

References

[1] "OLAP Council," 2007 [Online]. Available: http://www.olap-council.org
[2] J. Han, "OLAP mining: Integration of OLAP with data mining," in *Proc. IFIP Conf. Data Semantics*, 1997, pp. 1–11.
[3] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*. Cambridge, MA: AAAI/MIT Press, 1996.
[4] G. Gorry and M. S. Morton, "A framework for management information systems," *Sloan Management Rev.*, vol. 13, pp. 50–70, 1971.
[5] D. H. Lee and M. H. Kim, "Database summarization using fuzzy ISA hierarchies," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 27, no. 1, pp. 68–78, Jan. 1997.
[6] A. Laurent, "Extraction de Connaissances Pertinentesà Partir de Baes de Données Multidimensionnelles," Ph.D. dissertation, Lab. Inf. Paris 6, Paris, France, 2002.
[7] R. Alhajj and M. Kaya, "Integrating fuzziness into OLAP for multidimensional fuzzy association rules mining," in *Proc. 3rd IEEE Int. Conf. Data Mining (ICDM)*, Washington, DC, 2003, pp. 469–469.
[8] C. Jensen, A. Kligys, T. Pedersen, and I. Timko, "Multimendional data modeling for location-based services," *VLDB J.*, vol. 13, pp. 1–21, 2004.
[9] C. Molina, D. Sánchez, M. A. Vila, and L. Rodríguez-Ariza, "A new fuzzy multidimensional model," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 6, pp. 897–912, Dec. 2006.
[10] R. Agrawal, T. Imielinksi, and A. Swami, "Mining association rule between sets of items in large databases," in *Proc. ACM SIGMOD*, 1993, pp. 207–216.
[11] R. Agrawal and R. Sritkant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th Int. Conf. Very Large Data Bases*, 1994, pp. 478–499.
[12] J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash based algoritm for mining association rules," in *Proc. ACM SIGMOD*, Mayo, 1995, pp. 175–186.
[13] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1997, pp. 255–264.
[14] A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases," in *Proc. 21st Conf. Very Large Databases*, 1995.
[15] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2000.
[16] R. Srikant and R. Agrawal, "Mining generalized association rules," in *Proc. 21st Int. Conf. Very Large Data Bases (VLDB)*, 1995, pp. 407–419.
[17] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 1995, pp. 420–431.
[18] L. Shen and H. Shen, "Mining flexible multiple-level association rules in all concept hierarchies (extended abstract)," in *Proc. Database Expert Syst. Appl.: 9th Int. Conf. (DEXA)*, 1998, pp. 786–796.
[19] S.-J. Yen, "Mining generalized multiple-level association rules," in *Proc. Principles Data Mining Knowledge Discovery: 4th Euro. Conf. (PKDD)*, 2000, pp. 679–684.
[20] C.-L. Lui and F.-L. Chung, "Discovery of generalized association rules with multiple minimum supports," in *Proc. 4th Euro. Conf. Principles Data Mining Knowledge Discovery (PKDD)*, 2000, pp. 510–515.
[21] J. Han, Y. Cai, and N. Cercone, "Data-driven discovery of quantitative rules in relational databases," *IEEE Trans. Knowledge Data Eng.*, vol. 5, no. 1, pp. 29–40, Feb. 1993.
[22] M. K. Muyeba and J. A. Keane, "Interestingness in attribute-oriented induction (aoi): multiple-level rule generation," in *Proc. 4th Euro. Conf. Principles Data Mining Knowledge Discovery (PKDD)*, 2000, pp. 542–549.
[23] R. A. Angryk and F. E. Petry, "Mining multi-level associations with fuzzy hierarchies," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2005, pp. 785–790.

[24] M. Kamber, J. Han, and J. Chiang, "Metarule-guided mining of multi-dimensional association rules using data cubes," in *Proc. Knowledge Discovery Data Mining*, 1997, pp. 207–210.
[25] H. Zhu, "On-line analytical mining of association rules" Ph.D. dissertation, Simon Fraser University, Burnaby, BC, Canada, 1998.
[26] M. Kaya and R. Alhajj, "Fuzzy OLAP association rules mining-based modular reinforcement learning approach for multiagent systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 35, no. 2, pp. 326–338, Apr. 2005.
[27] R. Agrawal, A. Gupta, and S. Sarawagi, "Modeling multidimensional databases," IBM, IBM Almaden Research Center, San Jose, CA, 1995.
[28] R. Kimball, *The Data Warehouse Toolkit*. New York: Wiley, 1996.
[29] L. Cabibbo and R. Torlone, "Querying multidimensional databases," in *Proc. 6th Int. Workshop Databases Program. Lang. (DBPL6)*, 1997, pp. 319–335.
[30] L. Cabibbo and R. Torlone, "A logical approach to multidimensional databases," in *Proc. 6th Int. Conf. Extending Database Technol. (EDBT)*, 1998, pp. 183–197.
[31] C. Li and X. Wang, "A data model for supporting on-line analytical processing," in *Proc. 5th Int. Conf. Inf. Knowledge Manag. (CIKM)*, 1996, pp. 81–88.
[32] A. Datta and H. Thomas, "The cube data model: A conceptual model and algebra for on-line analytical processing in data warehouses," *Dec. Support Syst.*, vol. 27, pp. 289–301, 1999.
[33] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, and M. Venkatrao, "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," *Data Mining Knowl. Discovery*, vol. 1, pp. 29–53, 1997.
[34] R. R. Yager, "Aggregation operators and fuzzy systems modeling," *Fuzzy Sets Syst.*, vol. 67, pp. 129–145, 1994.
[35] R. R. Yager, "On the theory of bags," *Int. J. General Syst.*, vol. 13, pp. 23–37, 1986.
[36] M. Delgado, M. J. Martín-Bautista, D. Sánchez, and M. A. Vila, "On a characterization of fuzzy bags," *Lecture Notes Comput. Sci.*, vol. 2715, pp. 119–126, Jan. 2003.
[37] M. Atzmueller, J. Baumeister, and F. Puppe, "Quality measures for semi-automatic learning of simple diagnostic rule bases," in *Proc. 15th Int. Conf. Appl. Declarative Program. Knowledge Manag.*, 2004, pp. 203–213.
[38] A. An and N. Cercone, "An empirical study on rule quality measures," *Lecture Notes Comput. Sci.*, vol. 1711, pp. 482–491, Jun. 2004.
[39] P. Dean and A. Famili, "Comparative performance of rule quality measures in an induction system," *Appl. Intell.*, vol. 7, no. 2, pp. 113–124, 1997.
[40] W. Pedrycz, "Associations and rules in data mining: a link analysis," *Int. J. Intell. Syst.*, vol. 19, pp. 653–670, 2004.
[41] P.-N. Tan and V. Kumar, "Interestingness measures for association patterns: a perspective," presented at the Workshop Post-Process. Machine Learn. Data Min. (KDD), Boston, MA, 2000.
[42] R. Michalski, "Pattern recognition as rule-guided inductive inference," *IEEE Trans. Pattern Anal. Machine Learn.*, vol. PAMI-2, no. 4, pp. 349–361, Jul. 1980.
[43] P. Brazdil and L. Torgo, "Knowledge acquisition via knowledge integration," in *Proc. Current Trends Knowledge Acquisition*, 1990, pp. 90–104.
[44] E. Shortliffe and B. Buchanan, "A model of inexact reasoning in medicine," *Math. Biosci.*, vol. 23, pp. 351–379, 1975.
[45] M. Delgado, N. Marin, D. Sanchez, and M. Vila, "Fuzzy association rules: General model and applications," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 2, pp. 214–225, Apr. 2003.
[46] I. Bruha, *Machine Learning and Statistics*. New York: Wiley, 1996, ch. Quality of decision rules: Definitions and classification schemes for multiple rules.
[47] A. Famili, "Integrating learning and decision-making in intelligent manufacturing systems," *J. Intell. Robot. Syst.*, vol. 3, no. 2, pp. 117–130, 1990.
[48] M. Delgado, D. Sánchez, and M. Vila, "Fuzzy cardinality based evaluation of quantified sentences," *Int. J. Aprox. Reason.*, vol. 23, no. 1, pp. 23–66, Jan. 2000.



Nicolas Marin received the Ph.D. degree in computer science from the University of Granada, Granada, Spain.

He is an Associate Professor with the Department of Computer Science and Artificial Intelligence, University of Granada and a Researcher with the Intelligent Databases and Information Systems Research Group. His research interests include database design, data mining, software modelling, and mathematical theory.

Prof. Marín is a member of the IEEE Computer Society.

Carlos Molina was born in Granada, Spain, on October 15, 1979. He received the M.S. and Ph.D. degrees in computer science from the University of Granada, Granada, Spain, in 2002 and 2005, respectively.

Since 2004, he has been an Assistant Professor with the Department of Computer Science, University of Jaén, Jaén, Spain. His current research interests include the fields of Multidimensional Model, Data Mining, and Soft Computing.

José M. Serrano was born in Baeza, Jaén, Spain, in 1975. He received the Ph.D. degree in computer science from the University of Granada, Granada, Spain, in 2003,

Currently, he is teaching with the Computer Science Department, University of Jaén. He was a researching fellowship holder for five years with the University of Granada. His main research interests include knowledge discovery and data mining in fuzzy relational databases.

Amparo Vila received the Ph.D. degree in mathematics from the University of Granada, Granada, Spain.

She is a Full Professor with the Department of Computer Science and Artificial Intelligence, University of Granada, where she leads the Intelligent Databases and Information Systems Research Group. Her main research interests include the fields of database design, data mining, and mathematical theory.