Building Fuzzy Blocks from Data Cubes

Yeow Wei Choong, Anne Laurent, Dominique Laurent

Building Fuzzy Blocks from Data Cubes

Yeow Wei Choong
HELP University College
Kuala Lumpur
MALAYSIA
choongyw@help.edu.my

Anne Laurent
LIRMM
Université Montpellier II
Montpellier - FRANCE
laurent@lirmm.fr

Dominique Laurent
ETIS
Université de Cergy-Pontoise
Cergy-Pontoise - FRANCE
dominique.laurent@dept-info.u-cergy.fr

Abstract

Multidimensional databases have become very popular for decision making frameworks. In this context, huge amounts of data are stored in data warehouses and decision makers try and navigate through this data using OLAP tools in order to visualize and analyze it. Although navigating through the data is one of the key issues, many issues are still open, and users are still not provided with intelligent tools for automatically identifying relevant parts from the data.

In this paper, we address this problem and we propose to mine homogeneous areas of the data, which we call *blocks*. In previous work, we have defined a level-wise method to automatically mine such blocks. However, these blocks are crisp in their definition, although they are described by fuzzy rules. We extend our previous work by proposing ways to mine fuzzy blocks, and we compare the three approaches, showing that fuzzifying blocks leads to more clearly defined areas from the data.

Keywords: Multidimensional Databases, Levelwise Algorithms, Fuzzy Data Mining.

1 Introduction

Exploring large volumes of data is known to be a tedious process, and therefore, it is often the case that users wish to have a rough idea of the content of their data in order to identify relevant areas. This problem is particulary crucial when considering large *data cubes* in the context of multidimensional databases.

We recall in this respect that multidimensional databases were introduced about 10 years ago in [7], for the analysis of huge volumes of data, referred to as On-Line Analytical Processing (OLAP) in the literature. In this context, data are presented in data cubes that are defined over several *dimensions*, and that contain information called the *measure*. For instance, Figure 1 displays sales results in a data cube defined over two dimensions.

The OLAP operators called *roll-up* and *drill-down* are commonly used for the purpose of exploring the content of a data cube. These operators allow to explore the data cube according to different levels of granularity defined on dimensions: while rolling up according to one or several dimensions displays the data at a lower level of details, drilling down has the reverse effect of displaying the data at a higher level of details. However, it should be noticed that these operators work based on predefined hierarchies on dimensions, and thus, do not allow to summarize a data cube based on the measure values.

In [4], we have presented a method in order to automatically identify blocks of homogeneous measure values in a data cube, as shown in

Figure 1: Data cube and associated blocks

Figure 2: Interval-based blocks

Figure 1. This figure shows for instance a block defined by products $P1$, $P2$ and city $C1$ containing the measure value 6.

One important feature of our approach is that the blocks are computed based on thresholds that, roughly speaking, guarantee a minimal size and a minimal homogeneity. Consequently, a block may contain several measure values, and not only one value. For instance, in Figure 1, one block contains mainly the measure value 2, but also the measure value 8. Thus it turns out that blocks may overlap.

On the other hand, in most cases, the measure values in a cube are numerical and not frequently duplicated, making it difficult to identify areas containing the same measure value. This is why we propose in this paper to build blocks that contain *almost* the same measure value, instead of *exactly the same* measure value. For this purpose, we define two novel methods: the first one mines blocks containing measure values belonging to an interval, while the second method considers fuzzy intervals. The three methods are compared through experiments.

Referring to related work, in [2, 9, 10] the authors propose segmentation methods for data cubes. However, these approaches do not consider the measure value as the central criterion, as we do in our approach. Compression methods have been proposed in [12], but in this work, cube representations and homogeneous blocks generation are not considered. The work in [3] and [13] aims at dividing cubes into regions, which are meant to represent the whole cube, without loss of information. On the contrary, our goal is to summarize the data cube, so as to point out relevant areas, even if the whole cube is not represented.
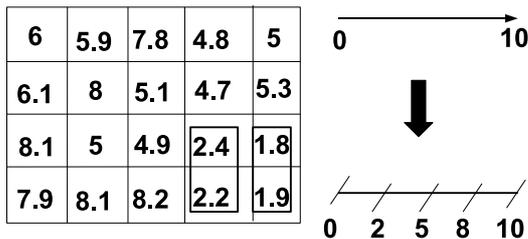
The paper is organized as follows: Section 2 motivates our work by an example. Section 3 introduces the multidimensional database framework considered in our approach. Section 4 recaps the way we build blocks from multidimensional databases. Section 5 introduces the contribution of this paper by presenting the way fuzzy blocks are defined. Section 6 presents the experiments we have performed, and Section 7 concludes the paper.

2 Motivating Example

Since in most cases a cube contains many measure values, it is neither efficient nor meaningful to consider each of them as a candidate block value, as done in [4]. For instance, in the cube of Figure 2, no relevant block can be found. For this reason, we propose to consider intervals instead of single measure values, so that two values in an interval can be considered as equal. For instance, in the cube of Figure 2, if we consider the intervals $[0, 2]$, $]2, 5]$, $]5, 8]$ and $]8, 10]$, then two blocks can be mined when considering the bottom-right part of the cube, since in this case 2.4 and 2.2 belong to $]2, 5]$ and 1.8 and 1.9 belong to $[0, 2]$.

Although considering intervals is more relevant to convey the main trends from the data, this method has some drawbacks due to the crisp cuts between two consecutive intervals. For instance, the two blocks displayed in Figure 2 could be merged as shown in Figure 3, since the associated values are close to each other. To this end, we consider fuzzy inter-
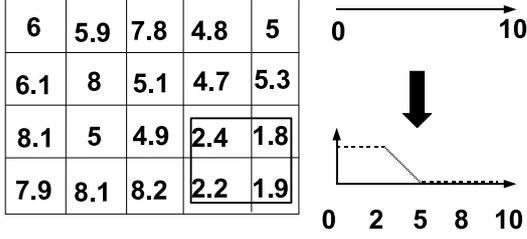
Figure 3: Fuzzy-interval-based blocks

vals in order to soften the membership of a measure value to an interval.

3 Multidimensional Databases

A cube can be seen as a set of cells. A cell represents the association of a *measure* value with one *member* value in each *dimension.* Moreover, *hierarchies* can be defined over dimensions so as to aggregate the data. For instance, the sales can be displayed in function of states instead of cities. In this paper, we do not consider hierarchies, and thus, this notion is not present in our definition of a cube:

Definition 1 - Cube. *A k-dimensional cube C, or simply a cube, is a tuple $\langle dom_1, \ldots, dom_k, dom_m, m_C \rangle$ where*

- $dom_1, \ldots, dom_k$ are k finite sets of symbols for the members associated with dimensions $1, \ldots, k$ respectively,

- let dom_{mes} be a finite totally ordered set of measures. Let $\perp \notin dom_{mes}$ be a constant (to represent null values). Then $dom_m = dom_{mes} \cup \{\perp\}$,

- m_C is a mapping from $dom_1 \times \ldots \times dom_k$ to dom_m assigning a measure value (possibly null) to each k-tuple of member values.

A *cell* c of a *k*-dimensional cube C is a $(k + 1)$-tuple $\langle v_1, \ldots, v_k, m \rangle$ such that for every $i = 1, \ldots, k$, v_i is in dom_i and where $m = m_C(v_1, \ldots, v_k)$. Moreover, m is called the *content* of c and c is called an *m-cell*.

Operations such as selection, projection, rotation or switch have been defined in the literature to manipulate data cubes. In [6], the switch operation is used to modify the representation of a cube without altering the data, while presenting the cube in an "ordered" way, so as to ease data exploration. We recall the main definition of [6] below.

Definition 2 - Representation. *A representation of a cube C is a set $R = \{rep_1, \ldots, rep_k\}$ where for every $i = 1, \ldots, k$, rep_i is a one-to-one mapping from dom_i to $\{1, \ldots, |dom_i|\}$.*

PRODUCT

	C3	C1	C2	C4	C5	C6
P4	8	8	8	2	2	2
P2	5	6	8	5	6	75
P1	8	6	6	5	5	2
P3	5	8	5	2	2	8

CITY

Figure 4: Inverting dimension values.

Figures 1 and 4 display two different representations of the same cube, and it should be clear that these two representations do not yield the same blocks.

In this paper, we consider a *fixed k-dimensional cube C* and a fixed representation of C, $R = \{rep_1, \ldots, rep_k\}$. Given a dimension d_i in C, and v_1 and v_2 in dom_i, v_1 and v_2 are said to be *contiguous* if $rep_i(v_1)$ and $rep_i(v_2)$ are consecutive integers, *i.e.,* if $|rep_i(v_1) - rep_i(v_2)| = 1$. Moreover, the *interval* $[v_1, v_2]$ is the set of all contiguous values between v_1 and v_2. In our approach, a block of C is a sub-cube of C.

Definition 3 - Block. *A block b is a set of cells defined over a cube C by $b = \delta_1 \times \ldots \times \delta_k$ where δ_i are intervals of contiguous values from dom_i, for $i = 1, \ldots, k$.*

Note that we consider a block as defined by k intervals, meaning that it can happen that intervals can contain the whole set of member values for a dimension. Such an interval is denoted by *ALL*.

Moreover, two blocks are said to *overlap* if they share at least one cell. It is easy to see

that two blocks $b = \delta_1 \times \ldots \times \delta_k$ and $b' = \delta'_1 \times \ldots \times \delta'_k$ *overlap* if and only if for each dimension d_i, $\delta_i \cap \delta'_i \neq \emptyset$.

In our formalism, a slice is defined as a particular block.

Definition 4 - Slice. *Let v_i be a value from dom_i. A* slice *of C associated with v_i, denoted $\mathcal{T}(v_i)$, is the block $\delta_1 \times \ldots \times \delta_k$ such that $\delta_i = \{v_i\}$, and for all $j \neq i$, $\delta_j = ALL$.*

A slice is a hyperplane, reduced to a single row in the particular case of a two-dimensional cube. Two slices defined on the same dimension d_i are said to be *contiguous* if they are associated with two contiguous values from d_i. For instance, in Fig. 1, the slices $\mathcal{T}(P3)$ and $\mathcal{T}(P4)$ are contiguous since $P3$ and $P4$ are contiguous in the considered representation.

In the following definitions of support and confidence for blocks, as well as in the rest of the paper, we use the following notation. Given a cube C, a block b and a measure value m, $|C|$ and $|b|$ denote the number of cells in C and in b, respectively, and $count(b, m)$ denotes the number of m-cells in b.

Definition 5 - Support. *The support of a block b from C for a measure value m is defined as: $sup(b, m) = \frac{count(b,m)}{|C|}$.*

Considering a user-given minimum support threshold σ and a measure value m, a block b such that $sup(b, m) \geq \sigma$ is called σ-frequent for m. Note that the support is monotonic with respect tro set inclusion, meaning that for all blocks b, b' and for each measure value m, we have:

$b \subseteq b' \Rightarrow sup(b, m) \leq sup(b', m)$.

This property is used in our Apriori-like algorithm, given in the forthcoming section.

Definition 6 - Confidence. *The confidence of a block b for a measure value m is defined as: $conf(b, m) = \frac{count(b,m)}{|b|}$.*

As mentioned previously, the support threshold determines the minimal size of blocks while the confidence threshold determines the homogeneity of the cell values within a block.

Indeed, for a given value of support σ, denoting by N the number of cells of the cube, a block can be frequent only if it contains at least $\sigma * N$ cells. Moreover, for a given value of confidence γ, a block of cardinality M is kept only if it contains at least $\gamma * M$ cells having the measure value m.

4 Block Generation

In this section, we briefly recall the algorithm defined in [4] which aims at mining blocks from a given cube, based on user-defined support and confidence thresholds. Blocks are mined using a levelwise method for scalability reasons. Given a support threshold σ and confidence thereshold γ, the corresponding Apriori-like algorithm works as follows:

For each measure value m present in the cube, do the following:

- *Step 1.* For every dimension i, all contiguous slices $\mathcal{T}(v_i)$ such that $v_i \in dom_i$ and $sup(\mathcal{T}(v_i), m) \geq \sigma$ are computed. This set of contiguous frequent slices defines a set, denoted by $\mathcal{L}_1(m)$, of intervals of contiguous measure values. Moreover, the set of all corresponding blocks is denoted by $\mathcal{B}_1(m)$. We note that this step requires one pass over the whole cube for each dimension.

- *Step 2.* The intervals in $\mathcal{L}_1(m)$ are then combined in much the same way as itemsets are combined in Apriori ([1]) in order to generate block candidates defined by 2, 3, $\ldots$, k intervals over different dimensions, in a levelwise manner. Note that in this step a pruning phase is considered in order to rule out blocks whose support for m in known to be less than σ. The supports of the remaining block candidates are then computed and the set of all blocks having a support for m greater than σ is treated as the input for the next level. The iteration stops when no new frequent blocks are found, or when the k levels have been explored. We note that for a given level, this step requires one pass over the whole cube.

- *Step 3.* Among all frequent blocks computed at the previous step, only the minimal ones are kept, and their confidence for m is computed so as to select those whose confidence is greater than or equal to γ. We note that this step does not require to access the cube, since for a given block b, $sup(b, m)$ is known from the previous step and $|b|$ can be easily obtained from the size of each interval defining b.

Algorithm 1 presents the computations of the last two steps above and we refer to [4] for more details about Step 1, as well as about the way fuzzy rules are generated to characterize these blocks.

5 Fuzzy Blocks

As argued in Section 2, considering separately all measure values present in the cube can lead to very poor results, while requiring a lot of computations. For instance, in a cube containing billions of cells and where the measure values range from 1 to $1,000$ with very few repetitions, almost $1,000$ values have to be considered separately in Algorithm 1. On the other hand, in this case, the measure values 5 and 5.2 are likely to be considered as similar.

In order to take this important point into account, we propose two ways to build fuzzy blocks, which rely on the fact that measure values are numerical and are rarely duplicated in the cube. The two methods we propose are based on the one hand, on *intervals* of measure values, and on the other hand, on *fuzzy intervals*. It is important to note that these methods do not require to change Algorithm 1, but only the way the support and the confidence are computed.

Definition 7 - Interval Support and Confidence. *The interval support of a block b in C for a measure value interval $[m1, m2]$ is defined as:*

$i\_sup(b, [m1, m2]) = \frac{iCount(b,[m1,m2])}{|C|}$

where $iCount(b, [m1, m2])$ is the number of m-cells in b such that $m \in [m1, m2]$.

Similarly, the interval confidence of b for $[m1, m2]$ is defined as:

$i\_conf(b, [m1, m2]) = \frac{iCount(b,[m1,m2])}{|b|}$.

When computing the fuzzy support of a block b with respect to a fuzzy interval φ, several methods are possible ([8]): (i) The Σ-*count* sums up the membership degrees of all cells of b; (ii) the *threshold-count* counts those cells of b whose membership degree is greater than a user-defined threshold; (iii) the *threshold-Σ-count* sums up those cell membership degrees that are greater than a user-defined threshold.

Definition 8 - Fuzzy Support and Confidence. *The fuzzy support of a block b in C for a fuzzy interval φ is defined as:*

$f\_sup(b, \varphi) = \frac{fCount(b,\varphi)}{|C|}$

where $fCount(b, \varphi)$ is one of the three fuzzy counting methods mentioned above. Similarly, the fuzzy confidence of b for φ is defined as:

$f\_conf(b, \varphi) = \frac{fCount(b,\varphi)}{|b|}$.

When considering intervals and fuzzy intervals, a pre-processing must be applied on the data in order to discretize the measure values into (fuzzy) intervals. This discretization can be automatically performed, provided the user defines the number of intervals (s)he wants to consider.

Denoting by n this number, and assuming that m_b (respectively m_t) is the bottom value (respectively the top value) of the measure values, $[m_b, m_t]$ can be divided into n intervals either in an equi-width manner (*i.e.,* the widths of all intervals are equal), or in an equi-depth manner (*i.e.,* all intervals cover the same number of cells). These intervals are denoted by $[bot_i, top_i]$ for $i = 1, \ldots, n$, and we note that for every $i = 2, \ldots, n$, $bot_i = top_{i-1}$.

Then, if fuzzy intervals are considered, a fuzzification is performed as illustrated in Figure 5. In this case, we consider n trapezoidal membership functions $\mu_1, \ldots, \mu_n$ such that:

- $[bot_1, top_1]$ and $[bot_1, top_2]$ are respectively the kernel and the support of μ_1,

- $[bot_n, top_n]$ and $[bot_{n-1}, top_n])$ are respec-

Algorithm 1: Computation of blocks

Data : data cube C, σ: support threshold, γ: confidence threshold

Result : set of blocks $\mathcal{B}$ associated with C

foreach *measure value m from C* **do**

> Compute $\mathcal{L}_1(m)$ and $\mathcal{B}_1(m)$;
>
> **for** $l = 2$ *to* k **do**
>
> > $\mathcal{B}_l(m) \leftarrow \emptyset$;
> >
> > Generate from $\mathcal{L}^i_{l-1}$ candidates $\delta_{i_1} \times \ldots \times \delta_{i_l}$ such that $\forall p, p' \in [1, l]$, $i_p \neq i_{p'}$;
> >
> > Let $\mathcal{L}_l(m)$ be this set ;
> >
> > Pruning: Delete from $\mathcal{L}_l(m)$ all candidates $\delta_{i_1} \times \ldots \times \delta_{i_l}$ such that there exists $p \in \{1, \ldots, l\}$ such that $\delta_{i_1} \times \ldots \times \delta_{i_{p-1}} \times \delta_{i_{p+1}} \times \ldots \times \delta_{i_l}$ is not frequent ;
> >
> > **foreach** *remaining candidate* $\delta_{i_1} \times \ldots \times \delta_{i_l}$ **do**
> >
> > > Let b be the block $\delta_1 \times \ldots \times \delta_k$ where $\delta_p = \delta_{p_j}$ if dimension d_p has been treated and $\delta_p = ALL$ otherwise.
> > >
> > > **if** $sup(b, m) < \sigma$ **then** remove $\delta_{i_1} \times \ldots \times \delta_{i_l}$ from $\mathcal{L}_l(m)$ **else** $\mathcal{B}_l(m) \leftarrow \mathcal{B}_l(m) \cup \{b\}$
>
> Let $\mathcal{B}(m)$ be the set of all minimal blocks b in $\bigcup_{l=1}^{l=k} \mathcal{B}_l(m)$ such that $conf(b, m) \geq \gamma$

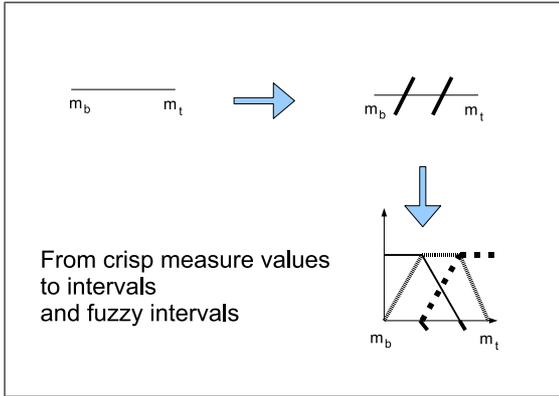$\mathcal{B} \leftarrow \bigcup_m \mathcal{B}(m)$

Figure 5: Measure value management

tively the kernel and the support of μ_n,

- $[bot_{i-1}, top_{i+1}]$ and $[bot_i, top_i]$ are respectively the support and kernel of μ_i, for $i = 2, \ldots, n - 1$.

Note that the support (respectively confidence) of blocks based on intervals and fuzzy intervals are greater than the support (respectively confidence) blocks based on crisp values, as stated in the following proposition whose easy proof is omitted.

Proposition. *For every block b and every measure value m, let m_1 and m_2 be measure values such that $m \in [m_1, m_2]$, and let φ be a fuzzy interval such that $kernel(\varphi) = [m_1, m_2]$. Then, for any of the three fuzzy counting*

methods *fCount, we have:* $count(b, m) \leq iCount(b, [m_1, m_2]) \leq fCount(b, \varphi)$.

As a consequence:
$sup(b, m) \leq i\_sup(b, [m_1, m_2]) \leq f\_sup(b, \varphi)$ *and* $conf(b, m) \leq i\_conf(b, [m_1, m_2]) \leq f\_conf(b, \varphi)$

It turns out that blocks based on intervals and fuzzy intervals are larger than blocks based on crisp values, as shown in Figures 2 and 3.

6 Experiments

In this section, we report on experiments in terms of (i) runtime, (ii) number of blocks and (iii) rate of overlapping blocks.

Experiments have been performed on synthetic data randomly generated. When considering intervals and fuzzy intervals, we chose to have 5 values defined in a equi-width manner and to use the Σ-count counting method.

Figure 6 shows how the runtime behaves vs. the number of cells in the cube, highlithing that fuzziness does not affect scalability.

Figure 7 shows the behavior of the ratio of blocks overlapping another one over the total number of blocks. It can be seen that building fuzzy blocks leads to less overlappings. Note that the less the rate of overlapping blocks, the better the quality of blocks.
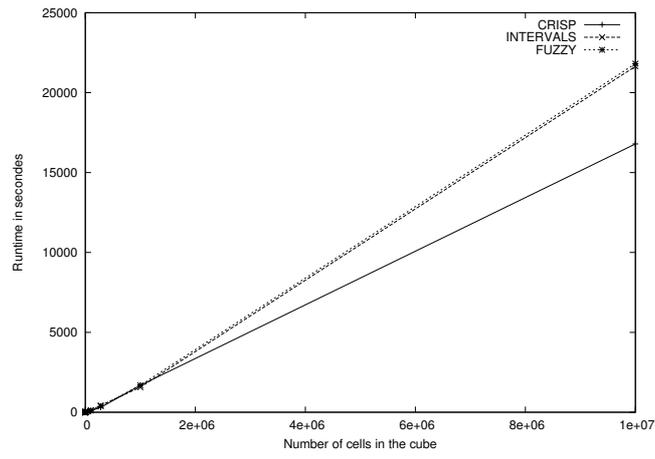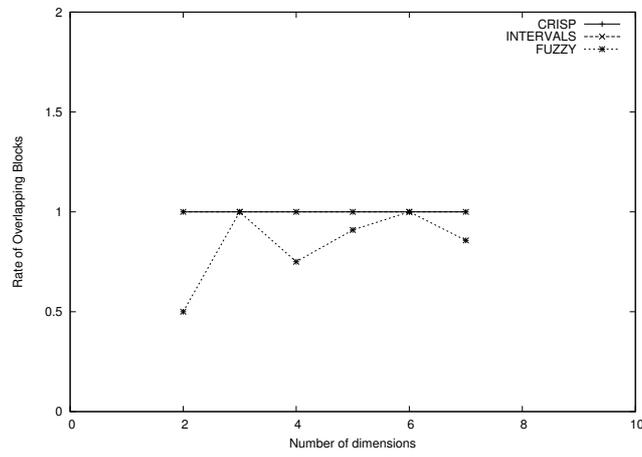
Figure 6: Runtime vs. the size of the cube

Figure 7: Rate of overlapping blocks vs. the number of dimensions
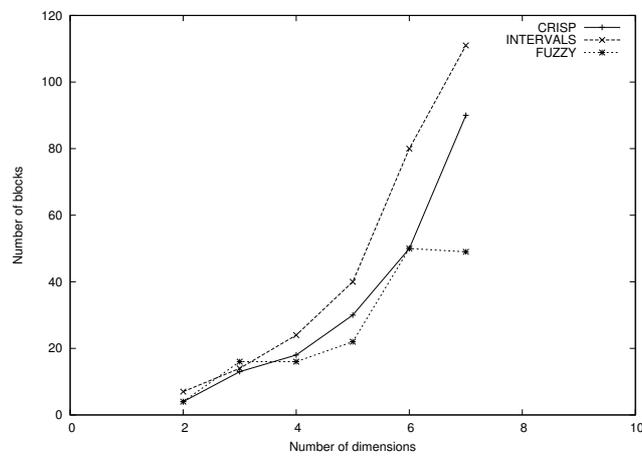

Figure 8: Number of blocks vs. the number of dimensions

Figure 8 shows how the number of blocks behaves vs. the number of dimensions in the cube. It can be seen that the increase in the number of blocks highly depends on the way measure values are partitioned: considering individual measure values or crisp intervals leads to similar exponential growths, whereas, in the case of fuzzy partitions, the increase is less fast.

7 Conclusion

Analyzing multidimensional databases is a challenging topic, as data warehouses are becoming larger in volume. Although decision making requires to navigate through the data, users are still not provided with automatic tools to identify relevant parts of these data. In our work, we propose to assist users to navigate through the data by automatically building blocks of homogeneous values.

In this paper, we have recalled the levelwise method first presented in [4], which aims at building blocks containing the same measure value and at describing these blocks using fuzzy rules. Then, we have extented this work by proposing to build blocks containing *almost* the same value. Two novel methods have been proposed: an interval-based method and a fuzzy-interval-based method. The three methods have been compared through experiments that show that considering fuzzy methods is relevant, since the blocks being discovered are more accurate.

As a future work, we aim at assessing our methods on real data and at comparing the results with different fuzzy partition and fuzzy counting methods. Moreover, we intend to study the impact of organizing the cubes based on the quality of the blocks. Finally, we plan to address the visualization issue based on our proposition, first proposed in [5].

Acknowledgements

References

[1] R. Agrawal and T. Imielinski and A. Swami (1993). Mining Association Rules Between Sets of Items in Large Databases. In *ACM SIGMOD.*

[2] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *ACM SIGMOD.*

[3] D. Barbara and M. Sullivan (1997). Quasi-cubes: Exploiting Approximation in Multidimensional Databases. In *SIGMOD Rec.,* vol. 26.

[4] Y.W. Choong, D. Laurent and A. Laurent (2004). Summarizing Multidimensional Databases Using Fuzzy Rules. In *Int. Conf. IPMU'04.*

[5] Y.W. Choong, D. Laurent and A. Laurent (2006). Pixelizing Data Cubes: a Block-Based Approach. In *Visual Information Expert Workshop (VIEW).*

[6] Y.W. Choong, D. Laurent and P. Marcel (2003). Computing Appropriate Representation for Multidimensional Data. In *Data and Knowledge Engineering Int. Journal,* vol. 45.

[7] E.F. Codd, S.B. Codd and C.T. Salley (1993). Providing OLAP to User-Analysts: An IT Mandate. *Tech. Rep..* Arbor Software Corp.

[8] D. Dubois, E. Hülermeier and H. Prade (2003). A note on quality measures for fuzzy association rules. *in Int. Fuzzy Systems Association World Congress on Fuzzy Sets and Systems.*

[9] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer and X. Xu (1998). Incremental Clustering for Mining in a Data Warehousing Environment. In *Int. Conf. on Very Large DataBases (VLDB).*

[10] A. Gyenesei and J. Teuholsa (2003). Multidimensional Partitioning of Attribute Ranges for Mining Frequent Fuzzy Patterns. In *FIP.*

[11] B. Inmon (1992). *Building the Data Warehouse.* John Wiley & Sons.

[12] L. Lakshmanan, J. Pei and J. Han (2002). Quotient Cube: How to Summarize the Semantics of a Data Cube. In *Int. Conf. on Very Large DataBases (VLDB).*

[13] W. Wang, H. Lu, J. Feng and J. Xu Yu (2002). Condensed Cube: An Effective Approach to Reducing Data Cube Size. In *Int. Conf. on Data Engineering (ICDE).*