Summarizing Multidimensional Databases Using Fuzzy Rules

Yeow Wei Choong, Anne Laurent, Dominique Laurent, Pierre Maussion

Summarizing Multidimensional Databases Using Fuzzy Rules

Yeow Wei Choong, Pierre Maussion
HELP Institute
Kuala Lumpur - MALAYSIA
choongyw@help.edu.my
maussp@help.edu.my

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

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

Abstract

In the context of multidimensional data, OLAP tools are appropriate for the navigation in the data, aiming at discovering pertinent and abstract knowledge. However, due to the size of the dataset, a systematic and exhaustive exploration is not feasible. Therefore, the problem is to design automatic tools to ease the navigation in the data and their visualization. In this paper, we present a novel approach allowing to build automatically blocks of similar values in a given data cube and to associate these blocks with rules. Our method is based on a levelwise algorithm (*a la* Apriori) and on the fuzzy set theory. The latter is considered here due to the fact that some of the blocks computed by our algorithm can overlap.

Keywords: Multidimensional Databases, Levelwise Algorithms, Fuzzy Rules, Data Visualization.

1 Introduction

Multidimensional databases have been studied and used for about 10 years [5]. Relational databases have been intially proposed for On-Line Transactional Processing (OLTP) in order to facilitate and manage transaction-oriented applications. However, relational databases are not suitable for the analysis of huge volumes of data, referred to as On-Line Analytical Processing (OLAP) in the literature. Multidimensional databases have been proposed to cope with this problem.

In this context, data are stored in multidimensional tables, the so-called *data cubes*, defined over several dimensions. The data within the cube is called *measure*. For instance, Figure 1 displays sale results in a data cube defined over two dimensions.

OLAP tools provide users with operations that ease manipulation of data in order to dicover relevant information.

However, data are often voluminous, and thus, an exhaustive exploration is impossible. Moreover, users of such systems are non-computer scientists but rather decision makers or experts of the data. Although many reporting tools exist in commercial softwares, it is still very difficult to visualize multidimensional databases so that relevant information is automatically displayed. Cube dimensionality is normally greater than 4, which makes visualization difficult.

In this paper, an automatic processing is proposed to facilitate multidimensional data visualization. Our method identifies automatically sub-cubes of similar data. In the previous example, the sub-cube defined by products $P1$, $P2$ and city $C1$ is a block corresponding to the measure value 6. Since a block is a sub-cube, it can be associated with a rule. For instance, the rule associated with the block mentioned above is: **If** PRODUCT = $P1$ **or** $P2$ **and** CITY = $C1$ **Then** sales = 6.

In this case, all cells of the block contain the same measure value. However, it is not always the case. For instance, the block corresponding to products $P1$, $P2$, $P3$ and cities $C3$ and $C4$ is mainly composed by the measure value 5. Therefore, one may consider it as being associated with 5. Similarly, a block associated with the measure value 2 can be considered for products $P3$, $P4$ and cities $C4$, $C5$, $C6$. These two blocks overlap since they share a cell for product $P3$ and city $C4$. As illustrated by this example, blocks may overlap. Fuzzy logic is used in order to convey these overlappings, making it possible to represent information such as *for product* $P2$ *and* in a lower manner *for product* $P3$.

The goal of this work is to identify blocks and their overlappings (as shown in Figure 1) as quickly as possible and then to build the corresponing rules (fuzzy or not).

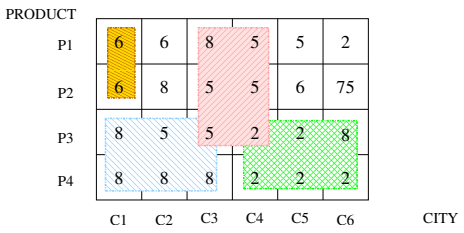

Figure 1: Data cube and associated blocks

Building such blocks facilitates visualization. The more the relevance of the blocks, the more the representation quality. In [4], the authors study different manners to represent data cubes. In particular, the authors highlight the fact that some representations are more relevant than others since they allow to display the cube according to user-specified criteria. In this work, we consider as relevant the configurations where same measure values are grouped. Another organization is possible, as described in [4], where the data are organized so that the measure is ordered in an increasing manner over all the dimensions. However, it is an NP-hard problem to automatically build such a representation of a cube. In this paper, the organization of data is not altered before the block identification, although it could be interesting.

It is also relevant to use the blocks computed by our approach in order to assess the quality of the representation of the cube. More precisely, this quality is related to the following:

- the proportion of cells included in the blocks (the higher the proportion, the lower amount of cells not covered by the rules),

- the number of built blocks (the more blocks there are, the more data are heterogeneous),

- the number of blocks in comparison with the number of measure values (if several blocks are built for the same measure value, then it is not stored in a contigous manner),

- the number of overlappings between blocks and their size (the higher the number of overlapping blocks, the more mixed are the data).

2 Multidimensional Databases

"*A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process*" [8]. Such systems, devoted to intensive decision-oriented querying, are different from classical relational database management systems which are not suitable in the OLAP framework. Multidimensional databases have thus been proposed in [5]. Data are represented as multidimensional tables. No consensual definition has emerged for now concerning data representation and manipulation.

Generally, a multidimensional database is a set of *hypercubes* (hereafter *cubes*). A cube can be seen as a set of cells and a cell represents the association of a *measure* with one member in each *dimension. Hierarchies* may be defined over dimensions, for instance to describe sales in function of states and not of cities.

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

- C is the name of the cube,
- $dom_1, \ldots, dom_k$ 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:* $dom_1 \times \ldots \times dom_k \rightarrow dom_m$.

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 a $m - cell$.

Operations are defined to manipulate data cubes: selection, projection, rotation, and switch. *Switch* is defined to modify cube representations without altering the data.

PRODUCT							
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	
	C3	C1	C2	C4	C5	C6	**CITY**

Figure 2: Inverting dimension values.

[4] details a study on cube representations. The authors propose an approach to restructure the dataset in order to obtain an appropriate representation.

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* $\{1, \ldots |dom_i|\}$.

Figures 1 and 2 display two different representations of the same cube.

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 *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 in the case where an interval is not specified on a dimension d_i, the interval is set to $\delta_i = ALL = dom_i$. Therefore, we shall consider a block as defined by k intervals.

Definition 4 - Block Overlapping. *Two blocks 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* iff for each dimension d_i, $\delta_i \cap \delta'_i \neq \emptyset$.

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

Definition 5 - 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 an hyperplan, 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 Figure 1, the slices $\mathcal{T}(P3)$ and $\mathcal{T}(P4)$ are contiguous since $P3$ and $P4$ are contiguous in the considered representation. Cells are considered as *neighbors* if they share a side or a corner in the representation. For instance, in Figure 1 the cells $\langle P2, C2, 8\rangle$ and $\langle P3, C1, 8\rangle$ are neighbors.

Definition 6 - Cell Neighborhood. *Two distinct cells* $c = \langle v_1, \ldots, v_k, m\rangle$ *and* $c' = \langle v'_1, \ldots, v'_k, m'\rangle$ *are* neighbors *if for every* $i = 1, \ldots, k$, $|rep_i(v_i) - rep_i(v'_i)| \leq 1$.

We note that in a k-dimensional cube, a cell has $3^k - 1$ neighbors.

Support and confidence are defined for blocks as follows.

Definition 7 - Support. *The support of a block* b *from* C *for a measure value* m *is defined as:* $supp(b, m) = \frac{\#\ cells\ containing\ m\ in\ b}{\#\ cells\ in\ C}$.

Considering a user-given minimum support threshold σ and a measure value m, a block b such that $supp(b, m) > \sigma$ is called σ-frequent for m. Note that the support is anti-monotonic, meaning that for all blocks b, b' and for each measure value m:

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

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

Definition 8 - Confidence. *The confidence of a block b for a measure value m is defined as:* $conf(b,m) = \frac{\#\ cells\ containing\ m\ in\ b}{\#\ cells\ in\ b}$.

Given a measure value m, a support threshold σ, a confidence threshold γ, *Most-General* and *Most-Specific* blocks with respect to m, σ and γ are defined as follows.

Definition 9 - MG-Block. *Given σ a support threshold, m a measure value, and b a σ-frequent block for m, b is said to be* most-general (MG) for m and the confidence threshold γ *if*

- $conf(b,m) > \gamma$
- *there does not exist a block b' such that:*
  - *b' is σ-frequent for m*
  - *$\exists j \in [1,k]$ such that $\delta'_j = ALL$ and $\delta_j \neq ALL$*
  - *$\forall j' \in [1,k]$, $j' \neq j \Rightarrow \delta'_j = \delta_j$*
  - *$conf(b',m) > \gamma$.*

Definition 10 - MS-Block. *Given σ a support threshold, m a measure value, and b a σ-frequent block for m, b is said to be* most-specific (MS) for m and the confidence threshold γ *if*

- $conf(b,m) > \gamma$
- *there does not exist a block b' such that:*
  - *b' is σ-frequent for m*
  - *$\exists j \in [1,k]$ such that $\delta'_j \neq ALL$ and $\delta_j = ALL$*
  - *$\forall j' \in [1,k]$, $j' \neq j \Rightarrow \delta'_j = \delta_j$*
  - *$conf(b',m) > \gamma$.*

3 Block Generation

In this paper, our goal is to discover blocks b such that $supp(b,m) > \sigma$ and $conf(b,m) > \gamma$, where σ and γ are respectively support and confidence thresholds specified by the user, and where m is a measure value appearing in the cube. Our method is based on a levelwise Apriori-like algorithm [1], for scalability reasons. The proposed algorithm builds MS-blocks, which correspond to rules in which the measure value is determined by the greatest number of dimensions. However, this algorithm is also suitable for the mining of MG-blocks, which correspond to the less specific rules. The user can decide whether he (she) wants MS- or MG-blocks. The proposed method is outlined in algorithms 1 and 2. Algorithm 2 computes the blocks, provided the intervals are computed.

PRODUCT

P1	5	3	7	8	8	10	
P2	4	5	1	8	8	8	
P3	8	8	8	5	9	12	
P4	8	8	8	6	3	1	
	C1	C2	C3	C4	C5	C6	**CITY**

Figure 3: The Cube for Example 1

Regarding the computation of the intervals, we introduce the following notation: Given a measure value m and a slice $\mathcal{T}(v)$ associated to value v in dom_i, let v' be the value in dom_i such that $rep_i^{-1}(v') = rep_i^{-1}(v) - 1$. If c is a m-cell in $\mathcal{T}(v)$, then we denote by $\mu(c)$ the number of m-cells in $\mathcal{T}(v')$ that are neighbors of c.

Then, the computation of the intervals for dimension d_i is based on two additional thresholds denoted by μ and ν. μ is a minimum number of m-cells in $\mathcal{T}(v')$ that are neighbors of a given m-cell c in $\mathcal{T}(v)$. ν is the minimum ratio between the number of m-cells in $\mathcal{T}(v)$ having at least μ m-cells neighbors in $\mathcal{T}(v')$ and the total number of m-cells in $\mathcal{T}(v)$. The following example illustrates how μ and ν are used in our approach.

Example 1 *Consider the cube C shown in Figure 3 in which blocks are to be computed according the support and confidence thresholds equal to $1/12$ and 50% respectively. Let b be the block $[P1,P4]\times[C1,C5]$. Clearly, $supp(b,8) > 1/12$, but $conf(b,8) \leq 50\%$, showing that b cannot be output by our method. On the other hand, for the measure value 8, the blocks $b_1 = [P1,P2] \times [C4,C5]$ and $b_2 = [P3,P4] \times [C1,C3]$ have their supports and confidences greater than $1/12$ and than 50% respectively.*

In order to see how our method can actually output b_1 and b_2, let us consider $\mu = 2$ and $\nu = 60\%$, and let us detail the computation for the slice $\mathcal{T}(C3)$, assuming that $\mathcal{T}(C1)$ and $\mathcal{T}(C2)$ have already been considered. In this case, an interval of the form $[C1, NIL]$ is under construction (NIL stands for an unknown value). At this stage of the computation, we know that $C1$ and $C2$ belong to this interval, and we have to decide for $C3$. To

this end, we first compute supp($\mathcal{T}(C3), 8$) and, as the result is greater than 1/12, the process continues (note that, otherwise NIL would be replaced by $C2$). The two 8-cells in $\mathcal{T}(C3)$ to consider are $c = \langle P3, C3, 8\rangle$ and $c' = \langle P4, C3, 8\rangle$, and we have $\mu(c) = \mu(c') = 2$. Since $\mathcal{T}(C3)$ contains two 8-cells, then the ratio between the number of 8-cells in $\mathcal{T}(C3)$ having at least μ 8-cells neighbors in $\mathcal{T}(C2)$ and the number of 8-cells in $\mathcal{T}(C3)$ is 1, which is greater than 60%. In this case, we consider that $C3$ should belong to the interval under construction, and thus, its current value remains $[C1, NIL]$.

The same computation for the slice $\mathcal{T}(C4)$ gives the following:
- supp($\mathcal{T}(C4), 8$) $= 2$ and thus the computation can continue
- the 8-cells $c = \langle P1, C4, 8\rangle$ and $c' = \langle P2, C4, 8\rangle$ are such that $\mu(c) = 0$ and $\mu(c') = 1$.
Thus, the ratio between the number of 8-cells in $\mathcal{T}(C4)$ having at least μ 8-cells neighbors in $\mathcal{T}(C3)$ and the number of 8-cells in $\mathcal{T}(C4)$ is 0. In this case, we consider that $C4$ should not belong to the interval under construction. Thus, this interval is set to $[C1, C3]$ and the "new" interval $[C4, NIL]$ is considered. Continuing this process, it can be seen that $C5$ is incorporated in $[C4, NIL]$, and that $C6$ is not, since supp($\mathcal{T}(C6), 8$) $\leq 1/12$. Therefore, we obtain the interval $[C4, C5]$.

Similar considerations on the dimension PRODUCT show that the blocks b_1 and b_2 can be computed for $\mu = 2$ and $\nu = 60\%$.

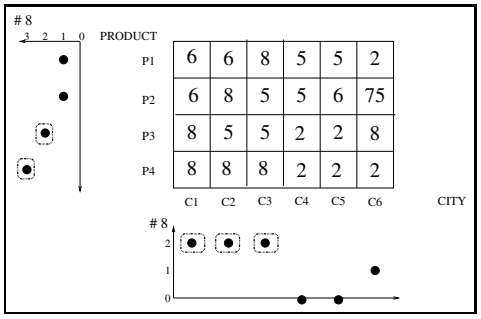

Figure 4: Occurences of measure value 8

Figure 4 illustrates the construction of the blocks for value 8. Figure 5 illustrates the result of this process for all measure values $(2, 5, 6, 8, 75)$. Four blocks are built:
$b_1 = [P1, P2] \times [C1, C1]$ for value 6
$b_2 = [P3, P4] \times [C1, C3]$ for value 8
$b_3 = [P1, P3] \times [C3, C4]$ for value 5
$b_4 = [P3, P4] \times [C4, C6]$ for value 2

Note that there are two overlappings (between b_2 and b_3, and between b_3 and b_4).

It is also worth mentioning that 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.

The algorithm is easily modified to build *less specific rules*. For this purpose, at each step the confidence of blocks is computed, and the scan of dimensions is stopped for each block having a sufficient confidence.

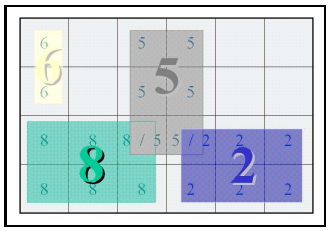

Figure 5: Output of Algorithm 1

As seen previously, blocks output by the above algorithm may overlap. In our approach, overlappings are computed based on the fact that two blocks overlap *if and only if all* intervals defining these blocks have a non-empty intersection. The proposed algorithm, which is omitted here, associates each block b_j in $\mathcal{B}$ with the set B_j of all blocks b in $\mathcal{B}$ such that $b \neq b_j$ and b overlaps b_j.

In our example (Figure 5), the following sets of blocks are obtained: $B_1 = \emptyset$, $B_2 = \{b_3\}$, $B_3 = \{b_2, b_4\}$, $B_4 = \{b_3\}$.

4 Rule Generation

In our model, one rule is associated with each block. When a block $b_j = \delta_{1,j} \times \ldots \times \delta_{k,j}$ does not overlap any other block ($B_j = \emptyset$), the generated rule for b_j is as follows:

If $d_1 \in \delta_{1,j}$ and ... and $d_k \in \delta_{k,j}$ Then m_j

where m_j is the associated measure value. For

Algorithm 1: Computation of $\mathcal{L}_1(m)$

$\mathcal{L}_1(m) \leftarrow \emptyset$;
foreach *dimension* d_i *i=1,...,k* **do**
  $intervals(m,i) \leftarrow \emptyset$; $currentInterval \leftarrow [NIL, NIL]$;
  foreach $j = 1, \ldots, |dom_i|$ **do**
    compute $supp(\mathcal{T}(rep_i^{-1}(j)), m)$;
    if $supp(\mathcal{T}(rep_i^{-1}(j)), m) \leq \sigma$ **then**
      if $currentInterval = [\alpha, NIL]$ **then**
        /*close the current interval at position $j-1$ and set the current interval to the empty interval */ $intervals(m,i) \leftarrow intervals(m,i) \cup \{[rep_i^{-1}(\alpha), rep_i^{-1}(j-1)]\}$;
        $currentInterval \leftarrow [NIL, NIL]$;
    else
      if $currentInterval = [NIL, NIL]$ **then**
        /*start a new current interval at position j */ $currentInterval \leftarrow [j, NIL]$;
      else
        /* $currentInterval = [\alpha, NIL]$ */ ;
        Let $neighbors(j,m)$ be the number of cells in $\mathcal{T}(rep_i^{-1}(j))$ having at least μ neighbors in $\mathcal{T}(rep_i^{-1}(j-1))$ containing m ;
        Let M be the number of cells in $\mathcal{T}(rep_i^{-1}(j))$ containing m ;
        if $(neighbors(j,m)/M) \leq \nu$ **then**
          /*close the current interval at position $j-1$ and start a new current interval at position j */ $intervals(m,i) \leftarrow intervals(m,i) \cup \{[rep_i^{-1}(\alpha), rep_i^{-1}(j-1)]\}$;
          $currentInterval \leftarrow [j, NIL]$;
$\mathcal{L}_1(m) \leftarrow \mathcal{L}_1(m) \cup intervals(m,i)$;
$\mathcal{B}_1(m) \leftarrow \{\delta_1 \times \ldots \times \delta_k \mid (\exists i, \delta_i \in intervals(i,m)) \text{ and } (\forall j \neq i, \delta_j = ALL)\}$

Algorithm 2: Discovery of MS-blocks

Data : data cube C, σ: support threshold, γ: confidence threshold, ν: homogeneity threshold.
Result : set of blocks $\mathcal{B}$ associated with C
foreach *measure value m from C* **do**
  Compute $\mathcal{L}_1(m)$;
  for $l = 2$ *to* k **do**
    $\mathcal{B}_l(m) \leftarrow \emptyset$;
    Generate from $\mathcal{L}_{l-1}^i$ 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 $supp(b,m) \leq \sigma$ **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 MS blocks b in $\bigcup_{l=1}^{l=k} \mathcal{B}_l(m)$ such that $conf(b,m) > \gamma$
$\mathcal{B} \leftarrow \bigcup_m \mathcal{B}(m)$
if *the data cube must be visualized* **then**
  foreach $b \in \mathcal{B}$ **do** replace the values of all cells of b by the measure value associated to b.
  /*Cells not belonging to a block are set to null. (Note that a cell may contain several values.)*/

instance, the rule built for the block b_1 (Figure 1) is:

If $CITY \in [C1, C1]$ *and* $PRODUCT \in [P1, P2]$ *Then* 6

If an overlapping is detected, then a fuzzy rule is built. Membership functions can be defined in different manners. In our approach, we define the membership function based on the number of overlapping blocks: for dimension values where there is no overlapping, the degree is 1, and if there are p overlapping blocks, then the degree is $\frac{1}{p}$.

Algorithm 3 describes this computation.

For instance, consider block $b_3 = [P1, P3] \times [C3, C4]$ with $B_3 = \{b_2, b_4\}$. For dimension $PRODUCT$, the interval $\delta_{PRODUCT,3} = [P1, P3]$ is divided into two sub-intervals $[P1, P2]$ and $[P3, P3]$. Applying the same method on dimension $CITY$, we obtain:

$$\Gamma_{PRODUCT,3}(P) = \begin{cases} 1 & \text{if } P \in [P1, P2] \\ \frac{1}{3} & \text{if } P \in [P3, P3] \\ 0 & \text{otherwise} \end{cases}$$

$$\Gamma_{CITY,3}(C) = \begin{cases} \frac{1}{2} & \text{if } C \in [C3, C3] \\ \frac{1}{2} & \text{if } C \in [C4, C4] \\ 0 & \text{otherwise} \end{cases}$$

Note that the intervals $[C3, C3]$ and $[C4, C4]$ can be merged into the single one $[C3, C4]$. Figure 6 illustrates the membership functions built for each block (or rule).

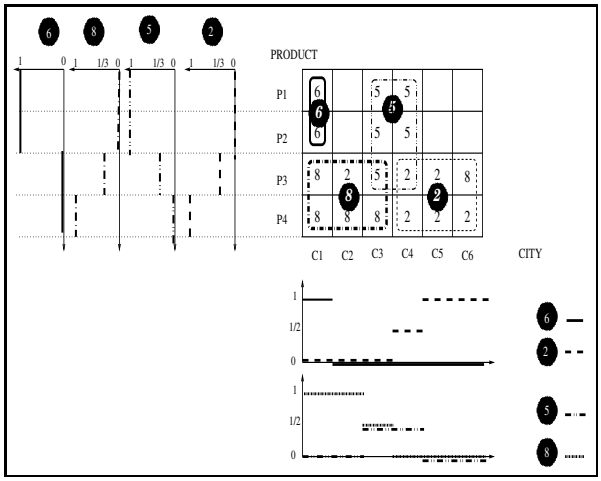

Figure 6: Fuzzy Set Construction

Algorithm 3: Computation of Fuzzy Sets

Data : n sets of blocks B_j

Result : Fuzzy Sets

foreach *set of blocks* $B_j = \{b_{j_1}, \ldots, b_{j_l}\}$ *with* $b_{j_p} = \delta_{1,j_p} \times \ldots \times \delta_{k,j_p}$ *and* $\delta_{r,j_p} = [\alpha_{r,j_p}, \beta_{r,j_p}]$ $(r = 1, \ldots, k)$ **do**

  foreach *dimension* d_i **do**

    foreach $q = 1, \ldots, l$ **do**

      $\alpha_{i,j_q} \leftarrow max(\alpha_{i,j}, \alpha_{i,j_q})$;

      $\beta_{i,j_q} \leftarrow min(\beta_{i,j}, \beta_{i,j_q})$;

      Sort the set $\{\alpha_{i,j_q} \mid q = 1, \ldots, l\} \cup \{\beta_{i,j_q} \mid q = 1, \ldots, l\}$;

      let $Int_{i,j} = \{[\alpha^1_{i,j}, \beta^1_{i,j}], \ldots, [\alpha^r_{i,j}, \beta^r_{i,j}]\}$ be the set of all sub-intervals of $[\alpha_{i,j}, \beta_{i,j}]$;

      Associate membership function $\Gamma_{i,j}$ to $[\alpha_{i,j}, \beta_{i,j}]$ as defined by:

      foreach *sub-interval* $[\alpha^s_{i,j}, \beta^s_{i,j}] \in Int_{i,j}$ **do**

        foreach *value* v *from* dom_i **do**

          $\Gamma_{i,j}(v) =$

          $\begin{cases} 0 & \text{if } v \notin [\alpha^s_{i,j}, \beta^s_{i,j}] \\ \frac{1}{(1+n_s)} & \text{otherwise} \end{cases}$

          n_s : # blocks $b_{j_p} \in B_j$ s.t. $[\alpha^s_{i,j}, \beta^s_{i,j}] \subseteq [\alpha_{i,j_p}, \beta_{i,j_p}]$

5 Related Work

Research work on (fuzzy) image segmentation may appear as related works [10]. Although

the goals are the same, it is not possible to apply such methods due to problems of scalability and because also of the multidimensional nature of data.

Segmentation methods (*e.g.* clustering) have been proposed in the multidimensional context [2, 6]. In [7], the authors study the generation of fuzzy partitions over numerical dimensions. However, these propositions are not related to the measure value and thus are different from our work where the measure value is the central criterion.

[9] aims at compressing data cubes. However there is no consideration on cube representations and homogeneous blocks generation.

[3] proposes a method to divide cubes into regions and to represent those regions. However, the authors aim at representing the whole cube. They use statistical methods to construct an approximation of the cube, while we aim at discovering relevant rules as fast as possible which may not cover the whole cube.

In [11], the authors propose the concept of *condensed* data cube. However, the authors aim at condensing the cube without loss of information while we aim at displaying relevant information to the user, which may be a partial representation of data.

6 Conclusion

In this paper, an efficient method for summarizing and visualizing multidimensional data is proposed. In our approach, blocks of homogeneous data are built and the block overlapping is managed using fuzzy sets. Each block is associated with one rule (fuzzy or not). The set of these rules is meant to summarize the cube. Further work include the implementation of our method, the introduction of fuzziness within blocks in order to discover blocks of *almost* the same value instead of homogeneous blocks, and the study of the pre-treatments in order to organize cubes in a relevant manner before mining blocks.

Acknowledgements

The authors acknowledge the French Embassy in Malaysia for its support.

References

[1] R. Agrawal and T. Imielinski and A. Swami (1993). Mining Association Rules Between Sets of Items in Large Databases. In *Proc. of ACM SIGMOD* pp 207-216.

[2] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proc. ACM SIGMOD.* pp 94-105.

[3] D. Barbara and M. Sullivan (1997). Quasicubes: Exploiting approximation in multidimensional database. In *SIGMOD Rec.*, 26:12-17.

[4] Y.W. Choong, D. Laurent and P. Marcel (2003). Computing Appropriate Representation for Multidimensional Data. In *DKE Int. Journal.* 45:181-203.

[5] E.F. Codd, S.B. Codd and C.T. Salley (1993). Providing OLAP to user-analysts: An IT mandate. In *Technical Report.* Arbor Software Corporation.

[6] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer and X. Xu (1998). Incremental Clustering for Mining in a Data Warehousing Environment. In *Proc. VLDB.* pp. 323-333.

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

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

[9] L. Lakshmanan, J. Pei and J. Han (2002). Quotient Cube: How to Summarize the Semantics of a Data Cube. In *Proc. of VLDB*, pages 778-789.

[10] S. Philipp-Foliguet (2000). Segmentation d'images en régions floues. In *Proc. LFA.* pp. 189-196 (french).

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