# Mining Multiple-Level Fuzzy Blocks from Multidimensional Data

Yeow Wei Choong, Anne Laurent, Dominique Laurent

# Mining Multiple-Level Fuzzy Blocks from Multidimensional Data ⋆

Yeow Wei Choong [a] Anne Laurent [b] Dominique Laurent [c]

[a] *HELP University College - Kuala Lumpur - Malaysia*
*choongyw@help.edu.my*

[b] *LIRMM - CNRS UMR 5506 - Université Montpellier 2 - France*
*laurent@lirmm.fr*

[c] *ETIS - CNRS UMR 8051 - Université Cergy-Pontoise - France*
*dominique.laurent@u-cergy.fr*

**Abstract**

Multidimensional databases are now recognized as being the standard way to store aggregated and historized data. Multidimensional databases are designed to store information on measures (also known as indicators) regarding a set of dimensions.

One important issue in this framework is the identification of homogeneous areas in data cubes, which allows users to summarize and visualize the data through the main trends they contain. In our previous work, we have proposed a levelwise approach to mine homogeneous areas of the data, called *blocks* that can be interpreted, for instance, as *If product is Chocolate and month is between January and March and city is London or Paris, then the number of sales is 5.*

However, in this work, the information provided by the hierarchies defined over the dimensions is not taken into account. In this paper, we consider the case where measure values are discretized using a fuzzy partition, and we extend our method so as to mine *multiple-level fuzzy* blocks, that is, blocks that are defined using hierarchies and that characterize fuzzy measure values. Moreover, in order to avoid redundancies in the output set of blocks, only the most specific ones (according to hierarchies) are computed. We show that our algorithms are linear in the size of the cube, thus providing an efficient method for summarizing data cubes.

*Key words:* Multidimensional Databases, Hierarchies, Levelwise Algorithms, Fuzzy Data Mining.

⋆ This paper is an extended version of the work presented in the proceedings of the eleventh international conference IPMU 2006, under the title "Building Fuzzy Blocks from Data Cubes" ([1]).

## 1 Introduction

Multidimensional databases have become very popular for decision making frameworks. They are designed to store information about particular so-called measures (*e.g.,* number of sales) organized by means of dimensions (*e.g.,* product, city, type of customer and month). For this reason, such databases are called *hypercubes*, or simply *cubes*. Considering a position on every dimension, the corresponding information is the measure value for this combination of values, and is called a *cell*.

These databases are aggregated views of raw data stored in operational databases, which are far too voluminous to be analyzed efficiently. For instance, it is not possible for shopping malls to record and analyze all the detail sales (cashier ticket by cashier ticket) as: (*i*) the number of transactions is huge, and (*ii*) decision makers do not care about the details, but rather prefer being aware of some trends.

In this context, the huge amounts of data stored in multidimensional databases are used by decision makers who:

- either try and navigate through this data using On-Line Analytical Processing tools, usually referred to as OLAP tools
- or use reporting tools,
- or use data mining techniques to automatically get relevant information.

Roughly speaking, a multidimensional database allows to define cubes, that are used to analyse the data according to specified *dimensions* and to one or several *measures*. Moreover, hierarchies can be defined over dimensions so as to analyze the data at different levels of granularity. For instance, sale quantities can be analyzed according to dimensions *Location* and *Product*, over which hierarchies can be used to aggregate sale quantities at different levels (*e.g.,* country instead of city for the dimension *Locations*, or type of product instead of individual product for the dimension *Product*).

OLAP tools provide users with operations over cubes (usually coming with graphical tools using commercial software) to navigate through the data. Even if there is no consensus on the operations defined for multidimensional databases, it is usually agreed on defining operations for:

- visualizing data from different points of view ;
- visualizing data at different levels of granularity using the roll-up and drill-down operations (note that these operations require the use of an aggregation operator so as to merge different values into a sigle one) ;
- selecting data, either by means of criteria on dimensions, or by means of criteria on measure values.

Although asking for reports and navigating through the data are key issues when anylizing multidimensional data, they require that the user has some *a priori* background knowledge about the data, which might not be the case for a casual user.

It is thus very important and challenging to design new tools to automatically extract relevant knowledge from multidimensional databases, based only on the stored data. In this framework, many issues are still open, such as the automatic identification of relevant parts in the data. These parts can for instance be related to homogeneous parts. The identification of such areas would then provide users with methods to summarize and visualize their data by using the main trends that they contain.

In our previous work ([2]), we have defined methods to mine homogeneous areas of the data, called *blocks*. Roughly speaking, a block can be seen as a sub-cube. These blocks are built up according two quality measures, called *support* and *confidence*. Given a measure value $m$ and a block $b$, the support of $b$ for $m$ is defined as being the ratio of the number of cells of $b$ that contain $m$ over the total number of cells in the data set. Similarly, the confidence of $b$ for $m$ is defined as being the ratio of the number of cells of $b$ that contain $m$ over the total number of cells in $b$.

Given a support threshold $\sigma$ and a confidence threshold $\gamma$, the blocks whose support and confidence are respectively above $\sigma$ and $\gamma$ are mined using a levelwise method that has been shown to be scalable when facing large amounts of data. In our method, the powerset of the set of dimensions is iteratively scanned level by level, so as to combine intervals of member values defined on dimensions. These intervals are computed at the first iteration, considering successively each individual dimension.

In [3], this approach has been extended to the case of *fuzzy blocks,* whereby, instead of considering individual measure values as such, measure values are partitioned in a fuzzy way. This allows for aggragating values closed to each other, and thus, provides a more relevant way of analyzing the data. For instance, a block identifying an area interpreted as

> *When product is Chocolate and month is between January and March and city is London or Paris then the number of sales is* almost *5*

involves sale values such as 4.8 or 5.7 with a membership degree defined by the considered fuzzy partition.

Note that these areas are described over the dimensions defining the dataset, which are usually more numerous than 2. Thus, when it comes to visualize these areas, they have to be displayed in a 2D manner. An approach to this issue was proposed in [4].

However, our previous work does not take into account the information provided by the hierarchies defined over the dimensions, and this prevents our approach from discovering summaries defined over different levels in these hierarchies.

For instance, when considering the sales almost equal to 8, even if no individual product of type food can be associated to a block, it may be the case that, when considering the data at a higher level, for instance at the level of types of products, the number of sales almost equal to 8 is sufficient to appear in a block, as shown by Figure 1.

In order to take this remark into account, in this paper, we assume that measure values are partitioned according a fuzzy partition and we consider *multiple-level fuzzy blocks, i.e.,* fuzzy blocks characterizing a fuzzy interval in the partition of measure values, and that are defined by intervals of values at different levels of the hierarchies defined on the dimensions. It is important to note that, in doing so, we do *not* aggregate measure values as would be done by applying a roll-up operation on the cube. Instead, each cell is considered at the lowest level and counted as such in the supports, but the blocks are defined according to a given level of the hierarchy on each dimension.

In our example, at the level of *category of products* and *geographical areas*, when considering the block defined by the category *Food* and the *South* area, we still consider individual cells for counting the support and the confidence. In this case the support and the confidence are equal to 5/24 and 5/6, respectively. As these values are respectively greater than the corresponding thresholds, namely 3/24 and 80%, the block is output.

On the other hand, it should be noticed that counting supports and confidences at the levels of individual products and cites produces no block. This is so because, according to our method, the cube contains two slices with at least 3 cells whose measure value is almost 8, namely the slices defined by *City*1 and *bread*, respectively. But

- when combining these slices, the produced block contains only one cell and thus, cannot have a support greater than or equal to 3/24,
- when considering these slices as blocks, their confidences are 3/4 and 5/6, respectively, which is less than 80%.

We note in this respect that this example illustrates that our approach is *not* complete, in the sense that there might exist blocks that fullfil the threshold conditions but that are not output. This point is discussed in details in [3], in the case where no hierarchies over dimensions are considered.

Thus compared to our previous work ([2,3]), the main contribution of this paper is to propose an extended method in order to mine multiple-level fuzzy

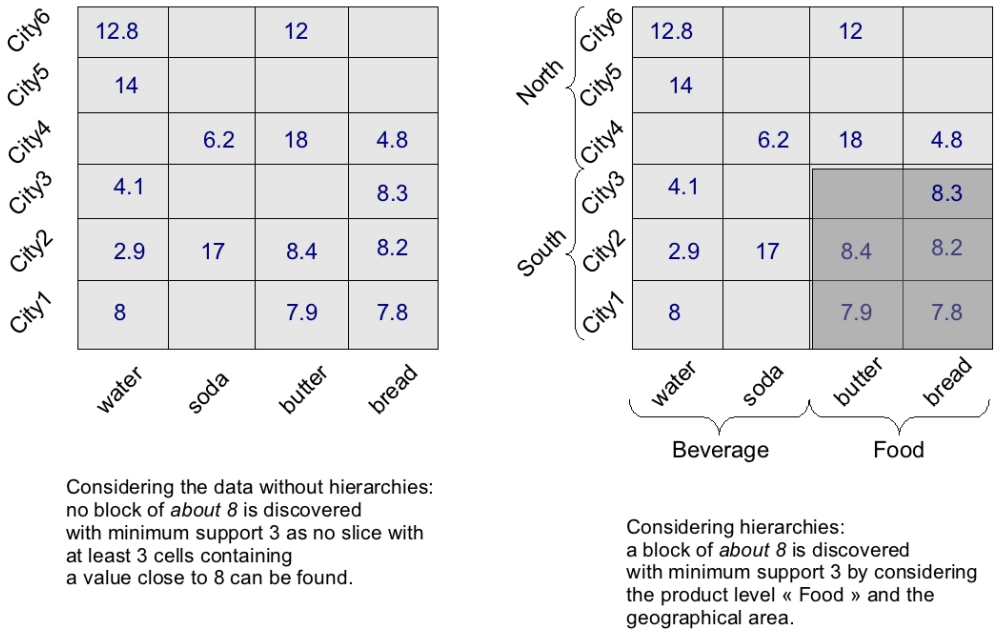


Fig. 1. Rolling-Up for Discovering Blocks ($\sigma = 3/24$ and $\gamma = 80\%$)

blocks from multidimensional databases. These blocks are defined so as:

- Measure values are considered according to a fuzzy partition over their domain, thus complying with the fact that two measure values closed to each other can be considered as *almost* equal.
- For each dimension, the most specific level in the associated hierarchy is considered, in order to avoid redundancies among output blocks.

To do so, we assume that we are given a support threshold $\sigma$ and a confidence threshold $\gamma$, and we use a levelwise approach that roughly works according to the following two steps, for each fuzzy interval $\varphi$ defined over measure values:

(1) The first step consists in computing for each dimension and at all levels of the corresponding hierarchy all maximal intervals of member values that define a slice whose support is greater than or equal to $\sigma$.
(2) In the second step, the intervals are combined to form blocks defined over all dimensions, among which only the most specific ones whose support and confidence are respectively greater than or equal to $\sigma$ and $\gamma$ are kept.

It is important to note that considering hierarchies over dimensions can still be processed efficiently, since it is shown in this paper that the time complexity of the corresponding algorithms is linear in the size of the cube. Moreover, it is also shown that our approach can be extended, while keeping linear algorithms in the size of the cube, to the cases where $(i)$ a fuzzy hierarchy is defined over

measure values, and (*ii*) hierarchies over dimensions are fuzzy.

The paper is organized as follows: In Section 2, we introduce the basic notions regarding the multidimensional data model (cubes and their associated hierarchies, and representations) and our approach (blocks and slices). In Section 3, we define the notions of support and confidence of a block, along with a specificity relation between blocks, with respect to which the support measure is shown to be anti-monotonic. Section 4 is devoted to the implementation issues of our approach (namely the corresponding algorithms and their time complexity) and to the extensions mentioned above. In Section 5, we report on preliminary experiments that show the effectiveness of our approach, and in Section 6, we briefly survey related work. We conclude the paper in Section 7.

## 2 Background

### *2.1 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. Formally, a cube is defined as follows.

**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*. A hierarchy over a dimension of a cube $C$ is defined as follows.

**Definition 2 - Hierarchy.** *Let $C$ be a $k$-dimensional cube and $d_i$ a dimension of $C$. A* $h_i$-level hierarchy *$H_i$ over $d_i$ is a $h_i$-tuple of pairs $(dom_i^j, h_i^j)$ where for every $j = 1, \ldots, h_i$, $dom_i^j$ and $h_i^j$ are respectively a set of values and a mapping defined as follows: Using the convention that $dom_i$ can also be denoted by $dom_i^0$, $H_i$ is such that*

- *For all distinct* $j$ *and* $j'$ *in* $\{0, \ldots, h_i\}$, $dom_i^j \cap dom_i^{j'} = \emptyset$.
- *The set* $dom_i^{h_i}$ *is a singleton whose element is denoted by* $\top_i$.
- *For every* $j = 1, \ldots, h_i$, $h_i^j$ *is a mapping from* $dom_i^{j-1}$ *to* $dom_i^j$.

*For every* $j = 1, \ldots, h_i$, *and every element* $v$ *in* $dom_i^{j-1}$, $h_i^j(v)$ *is called the* successor *of* $v$ *according to* $H_i$, *and* $v$ *is called a* predecessor *of* $h_i^j(v)$ *according to* $H_i$. *For every* $j' = j, \ldots, h_i$, $h_i^{j'}(\ldots(h_i^j(v))\ldots)$ *is called an* ancestor *of* $v$ *according to* $H_i$, *and* $v$ *is called a* descendent *of* $h_i^{j'}(\ldots(h_i^j(v))\ldots)$ *according to* $H_i$.

For example, referring back to Figure 1, we consider a 2-dimensional cube $C$. If the dimension represented on the horizontal axis is $d_1$, then a 2-level hierarchy $H_1$ is defined on $d_1$ by:

- $dom_1 = dom_1^0 = \{water, soda, butter, bread\}$, $dom_1^1 = \{Beverage, Food\}$ and $dom_1^2 = \{\top_1\}$,
- $h_1^1(water) = h_1^1(soda) = Beverage$ and $h_1^1(butter) = h_1^1(bread) = Food$,
- $h_1^2(Beverage) = h_1^2(Food) = \top_1$.

Thus, *Beverage* is the successor and so, one ancestor of *water* and *soda*. Similarly, $\top_1$ is an ancestor of all elements in $dom_1$.

### *2.2 Representations*

Operations such as selection, projection, rotation or switch have been defined in the literature to manipulate data cubes. In [5], 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 [5] below.

**Definition 3 - 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, and it should be clear that, although displaying the same cube, these two representations do not yield the same blocks.

Let $C$ be a $k$-dimensional cube and $R = \{rep_1, \ldots, rep_k\}$ a representation of $C$. 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 $|rep_i(v_1) - rep_i(v_2)| = 1$. Then, the *interval* $[v_1, v_2]$ is the set of all contiguous values between $v_1$ and $v_2$.

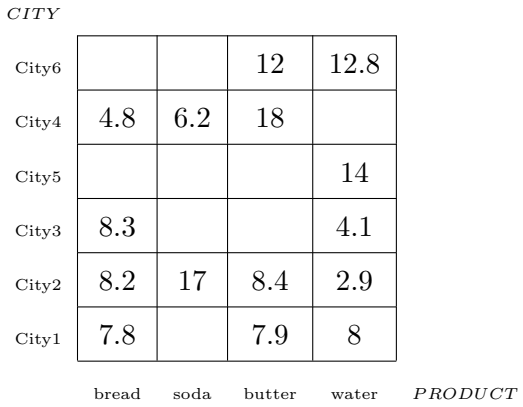


Fig. 2. A non Hierarchy-Consistent Representation of the Cube of Figure 1

Since hierarchies are taken into account in this work, we generalize the notion of representation as follows. Let $C$ be a $k$-dimensional cube with hierarchies $H_1, \ldots, H_k$. Then, a representation $R$ of $C$ is said to be *hierarchy-consistent* if the following holds:

> For every $i = 1, \ldots, k$ and every $j = 1, \ldots, h_i$, the set of all member values in $dom_i^0$ having the same ancestor in $dom_i^j$ is an interval.

For example, the representation of the cube in Figure 1 is hierarchy-consistent, whereas the representation of the same cube shown in Figure 2 is not. Indeed, the representation in Figure 2 is obtained from that of Figure 1 by switching $City4$ and $City5$ on dimension $CITY$, and *water* and *bead* on dimension $PRODUCT$. Although switching $City4$ and $City5$ preserves consistency for the hierarchy over this dimension, when switching *water* and *bread*, the set of all members having $Food$ as ancestor, namely $\{butter, bread\}$, is *not* an interval in this representation.

Considering a hierarchy-consistent representation, for every $i = 1, \ldots, k$ and every $j = 1, \ldots, h_i$, it is then possible to define one-to-one mappings $rep_i^j$ from $dom_i^j$ to $\{1, \ldots, |dom_i^j|\}$. Consequently, the notions of contiguous values and of intervals in $dom_i^j$ are defined as above at any level of any hierarchy $H_1, \ldots, H_k$. Moreover, in order to simplify notation, an interval containing only one value, *i.e.,* an interval of the form $[v, v]$, is simply denoted by $[v]$.

Based on this notation, when considering a hierarchy-consistent representation, it is easy to see that for every $i \in \{1, \ldots, k\}$, if for $j \in \{0, \ldots, h_i\}$, $\delta_i^j$ is an interval in $dom_i^j$ then, for every $j' > j$, the set

$$\{v' \in dom_i^{j'} \mid (\exists v \in \delta_i)(v' = h_i^{j'}(\ldots(h_i^j(v))\ldots)\}$$

is an interval in $dom_i^{j'}$. This interval is denoted by $h_i^{j'}(\delta_i)$.

In the remainder of this paper, we consider a fixed $k$-dimensional cube $C$ and a fixed hierarchy-consistent representation of $C$, $R = \{rep_1, \ldots, rep_k\}$.

### *2.3 Blocks*

In our approach, a block of $C$ is a sub-cube of $C$.

**Definition 4 - Block.** *Given a $k$-dimensional cube $C$ with hierarchies $H_1, \ldots, H_k$, a* block *$b$ is a set of cells of $C$ defined by $b = \delta_1 \times \ldots \times \delta_k$ where for $i = 1, \ldots, k$, $\delta_i$ is an interval of contiguous values in $dom_i^j$, for some $j$ in $\{0, \ldots, h_i\}$.*

Note that we consider a block $b = \delta_1 \times \ldots \times \delta_k$ 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 $ALL_i^j$, where $j$ is the level of the hierarhy $H_i$ according to which $\delta_i$ is defined, *i.e.,* the integer in $\{0, \ldots, h_i\}$ such that $\delta_i \subseteq dom_i^j$. It should be noticed that a block $b$ defined by $b = \delta_1 \times \ldots \times \delta_{i-1} \times ALL_i^j \times \delta_{i+1} \times \ldots \times \delta_k$ contains the same cells as the block $b'$ defined by $b' = \delta_1 \times \ldots \times \delta_{i-1} \times [\top_i] \times \delta_{i+1} \times \ldots \times \delta_k$.

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 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 $j \neq i$, $\delta_j = ALL_j^0$.*

Two slices defined on the same dimension $d_i$ are said to be *contiguous* if they are associated with two contiguous values from $dom_i$.

For instance, in Figure 1, the slices $\mathcal{T}(City3)$ and $\mathcal{T}(City4)$ are contiguous since $City3$ and $City4$ are contiguous in the considered representation. On the other hand, $\mathcal{T}(soda)$ and $\mathcal{T}(bread)$ are not contiguous because *soda* and *bread* are not contiguous.

## 3 Multiple-Level Fuzzy Blocks

As mentioned in the introductory section, in order to obtain relevant summaries, we do not consider measure values as such. Instead, we assume that

a fuzzy partition is defined over the set $dom_{mes}$. In this case, counting the support and the confidence of a given block $b$ is achieved accordingly.

### *3.1 Definitions*

When considering fuzzy intervals, counting cells in a block $b$ with respect to a fuzzy interval $\varphi$ can be computed according to the following methods ([6,7]):

(1) The $\Sigma$-*count* sums up the membership degrees of all cells of $b$.
(2) The *threshold-count* counts those cells of $b$ whose membership degree is greater than a user-defined threshold.
(3) The *threshold*-$\Sigma$-*count* sums up those cell membership degrees that are greater than a user-defined threshold.

In what follows, given a fuzzy interval $\varphi$ and a cell $c$ with content $m$, we denote by $\mu(c, \varphi)$ the membership value of $m$ in $\varphi$. Moreover, given a block $b$, $\Sigma^f_{c \in b} \mu(c, \varphi)$ denotes the count of cells in $b$ whose content is in $\varphi$, according to one of the three counting methods mentioned above. In this case, the support and confidence of a block $b$ are defined as follows.

**Definition 6 - Fuzzy Support and Confidence**. *The* support *of a block $b$ in $C$ for a fuzzy interval $\varphi$ is defined as:*

$$supp(b, \varphi) = \frac{Count(b, \varphi)}{|C|}$$

*where $Count(b, \varphi) = \Sigma^f_{c \in b} \mu(c, \varphi)$. Given a support threshold $\sigma$, $b$ is said to be* $\sigma$-frequent for $\varphi$ *if $supp(b, \varphi) \geq \sigma$.*

*Similarly, the* confidence *of $b$ for $\varphi$ is defined as:*

$$conf(b, \varphi) = \frac{Count(b, \varphi)}{|b|}.$$

In order to illustrate the definition above, let us consider the cube shown in Figure 1. Let us assume that $\varphi$ is a fuzzy interval whose kernel is $[7, 8]$ and whose support is $[6, 9]$ and that counting is achieved using the $\Sigma$-count function. For the block $b = [soda, bread] \times [South]$, we have:

(1) If we consider the cells of $b$ in a left-right and bottom-up way, we have: $supp(b, \varphi) = \frac{0+0+0+0.9+0.6+0+0.8+0.8+0.7}{24} = \frac{3.8}{24}$.
(2) Thus, $conf(b, \varphi) = \frac{3.8}{9}$.

Consequently, for a support threshold $\sigma = 3/24$, $b$ is $\sigma$-frequent for $\varphi$, but, if we consider a confidence threshold $\gamma = 80\%$, we have $conf(b, \varphi) < \gamma$.

We now define the following *specificity relation* between blocks of a given $k$-dimensional cube $C$ and its associated hierarchies $H_1, \ldots, H_k$.

**Definition 7 - Specificity Relation**. *Let $b = \delta_1 \times \ldots \times \delta_k$ and $b' = \delta'_1 \times \ldots \times \delta'_k$ be two blocks. $b'$ is said to be* more specific *than $b$, denoted by $b \sqsubseteq b'$, if for every $i = 1, \ldots, k$,*

$$\delta_i \neq \delta'_i \Rightarrow (j > j') \wedge (h_i^j(\delta'_i) \subseteq \delta_i)$$

*where $j$ and $j'$ are the levels in $H_i$ over which $\delta_i$ and $\delta'_i$ are defined, that is, $j$ and $j'$ are integers in $\{0, \ldots, h_i\}$ such that $\delta_i \subseteq dom_i^j$ and $\delta'_i \subseteq dom_i^{j'}$.*

For instance, in the cube of Figure 1, for $b = [\top_{PRODUCT}] \times [City1, City3]$ and $b' = [Food] \times [City1, City3]$, we have $b \sqsubseteq b'$ since the intervals defining $b$ and $b'$ satisfy the above definition.
On the other hand, $b$ and $b'' = [water, butter] \times [North]$ are not comparable according to $\sqsubseteq$, because $[water, butter]$ is defined at a lower level than $[\top_{PRODUCT}]$, whereas $[North]$ is defined at a higher level than $[City1, City3]$.

### *3.2 Properties*

The following lemma states that the relation $\sqsubseteq$ as defined above is a partial ordering over the set of all blocks of the cube $C$ that refines inclusion of blocks.

**Lemma 1** *Given a $k$-dimensional cube $C$ with hierarchies $H_1, \ldots, H_k$, the relation $\sqsubseteq$ is a partial ordering over the set of all blocks. Moreover, for all blocks $b$ and $b'$, if $b \sqsubseteq b'$ then $b' \subseteq b$.*

Proof. We first note that $\sqsubseteq$ is trivially reflexive. Regarding anti-symmetry, let us consider $b = \delta_1 \times \ldots \times \delta_k$ and $b' = \delta'_1 \times \ldots \times \delta'_k$ such that $b \sqsubseteq b'$ and $b' \sqsubseteq b$. If $b \neq b'$ then there exists $i \in \{1, \ldots, k\}$ such that $\delta_i \neq \delta'_i$. In this case, according to Definition 7, these intervals are defined at levels $j$ and $j'$ of $H_i$ such that $j < j'$ and $j' < j$, which is a contradiction. Therefore, $b = b'$, which shows that $\sqsubseteq$ is anti-symmetric.
Let us now consider three blocs $b = \delta_1 \times \ldots \times \delta_k$, $b' = \delta'_1 \times \ldots \times \delta'_k$ and $b'' = \delta''_1 \times \ldots \times \delta''_k$ such that $b \sqsubseteq b' \sqsubseteq b''$, and let us show that $b \sqsubseteq b''$. If $b = b''$ then the result follows trivially. Otherwise, let $i$ be such that $\delta_i \neq \delta''_i$. In this case, we have $\delta_i \neq \delta'_i$ or $\delta'_i \neq \delta''_i$. Thus, assuming that $\delta_i$, $\delta'_i$ and $\delta''_i$ are respectively defined at levels $j$, $j'$ and $j''$, by Definition 7, we have $j > j''$ and $h_i^j(\delta''_i) \subseteq \delta_i$. Therefore, $\sqsubseteq$ is transitive.
Let $b$ and $b'$ such that $b \sqsubseteq b'$, and let $c = \langle v_1, \ldots, v_k, m \rangle$ be a cell in $b'$. Then, for every $i = 1, \ldots, k$, $v_i$ has an ancestor in $\delta'_i$. By Definition 7, for every $i = 1, \ldots, k$, we have either $\delta'_i = \delta_i$ or $h_i^j(\delta'_i) \subseteq \delta_i$. Thus, in both cases, $v_i$ has an ancestor in $\delta_i$, which shows that $c$ belongs to $b$. Thus we have $b' \subseteq b$. □

Given a set of blocks $B$, the maximal (respectively minimal) elements of $B$ are said to be *most specific* (respectively *most general*) in $B$. Most specific blocks and most general blocks are called *MS-blocks* and *MG-blocks*, respectively.

The following property states that the support measure is anti-monotonic with respect to the preordering $\sqsubseteq$.

**Proposition 1** *Let $C$ be a $k$-dimensional cube. For all blocks $b$ and $b'$ of $C$, if $b \sqsubseteq b'$, then for every fuzzy interval $\varphi$, $supp(b', \varphi) \leq supp(b, \varphi)$.*

Proof. Since we assume that $b \sqsubseteq b'$, by Lemma 1, we have that $b' \subseteq b$. Then, considering any of the three counting method mentioned above, we have $Count(b', \varphi) \leq Count(b, \varphi)$, which implies that $supp(b', \varphi) \leq supp(b, \varphi)$. □

Proposition 1 above is used in our algorithms in the same way as done in [8] for pruning the search space when computing frequent itemsets. More precisely, given a support threshold $\sigma$, assuming that $b$ is a block such that $supp(b, \varphi) < \sigma$ then, for every block $b'$ such that $b \sqsubseteq b'$, we know that $supp(b', \varphi) < \sigma$ without accessing the data.

The following proposition shows that when considering intervals over all dimensions of the cube, at all levels of the associated hierarchies, then the set of blocks obtained by combining these intervals is a lattice.

**Proposition 2** *Let $C$ be a $k$-dimensional cube with hierarchies $H_1, \ldots, H_k$. Let $\mathcal{I}$ be a set of intervals such that, for every $i = 1, \ldots, k$:*

*(1) $\mathcal{I}$ contains at least one interval $I$ such that $I \subseteq dom_i^j$, for some $j$ in $\{0, \ldots, h_i\}$.*
*(2) If $I$ and $I'$ are intervals in $\mathcal{I}$ such that $I \subseteq dom_i^j$ and $I' \subseteq dom_i^j$ for some $j$ in $\{0, \ldots, h_i\}$, then $I \cap I' = \emptyset$.*
*(3) For every $I$ in $\mathcal{I}$ such that $I \subseteq dom_i^j$, for some $j$ in $\{0, \ldots, h_{i-1}\}$, there exists $I'$ in $\mathcal{I}$ such that $I' \subseteq dom_i^{j+1}$, and $h_i^{j+1}(I) \subseteq I'$.*

*Let $\mathcal{B}(\mathcal{I})$ be the set containing the empty block along with all blocks of the form $I_1 \times \ldots \times I_k$ where, for every $i = 1, \ldots, k$, $I_i$ is in $\mathcal{I}$. Then $\langle \mathcal{B}(\mathcal{I}), \sqsubseteq \rangle$ is a lattice.*

Proof. Let $b = I_1 \times \ldots \times I_k$ and $b' = I'_1 \times \ldots \times I'_k$ be two blocks $\mathcal{B}(\mathcal{I})$. We first note that condition (1) in the proposition ensures that at least one such block exists. In order to show that $\langle \mathcal{B}(\mathcal{I}), \sqsubseteq \rangle$ is a lattice, we have to prove that $\mathcal{B}(\mathcal{I})$ contains a unique least upper bound and a unique greatest lower bound for $b$ and $b'$, according to $\sqsubseteq$. For every $i = 1, \ldots, k$, we have $I_i \subseteq dom_i^j$ and $I'_i \subseteq dom_i^{j'}$, where $j$ and $j'$ are in $\{0, \ldots, h_i\}$. By Definition 2, we have $dom_i^{h_i} = \{\top_i\}$, thus by condition (3) in the proposition, there exist a unique

least integer $j-$ in $\{0,\ldots,h_i\}$ and an interval $I_i^-$ in $\mathcal{I}$ such that:
$(i)$ $I_i^- \subseteq dom_i^{j-}$ and $j- \geq min(j,j')$,
$(ii)$ $h_i^{i-}(I_i) \subseteq I_i^-$ and $h_i^{j-}(I_i') \subseteq I_i^-$.
Moreover, the condition (2) in the proposition implies that this interval $I_i^-$ is unique. Thus, denoting by $b^-$ the block $I_1^- \times \ldots \times I_k^-$, we have that $b^- \in \mathcal{B}(\mathcal{I})$. Moreover, $b^- \sqsubseteq b$ and $b^- \sqsubseteq b'$, and it turns out that $b^-$ is the unique greatest lower bound of $b$ and $b'$.
A similar reasoning holds for the lowest upper bound of $b$ and $b'$, considering the unique greatest $j+$ in $\{0,\ldots,h_i\}$ and the interval $I_i^+$ in $\mathcal{I}$ such that:
$(i)$ $I_i^+ \subseteq dom_i^{j+}$ and $j+ \leq max(j,j')$,
$(ii)$ $h_i^{i+}(I_i) \subseteq I_i^+$ and $h_i^{j+}(I_i') \subseteq I_i^+$.
In this case, denoting by $b^+$ the block $I_1^+ \times \ldots \times I_k^+$, we have that $b^+ \in \mathcal{B}(\mathcal{I})$, $b \sqsubseteq b^+$ and $b' \sqsubseteq b^+$, and it turns out that $b^+$ is the unique lowest upper bound of $b$ and $b'$. We note however that $b^+$ may be empty when, for at least one $i$ in $\{1,\ldots,k\}$, $I_i^+ = \emptyset$. □

Proposition 2 above is used in the algorithms given below in order to show that, when computing $\sigma$-frequent blocks for a given fuzzy interval $\varphi$, the search space is a lattice.

## 4 Implementation Issues

In this section we first present the algorithms for mining multiple-level blocks, and we show that the time complexity of these algorithms is linear in the size of the cube. Then, we discuss two possible extensions of our approach.

### *4.1 Algorithms*

Given a $k$-dimensional cube $C$ with hierarchies $H_1,\ldots,H_k$, a support threshold $\sigma$ and a confidence threshold $\gamma$, our method works according to the following three steps, for every fuzzy interval $\varphi$:

(1) *Step 1.* Compute all intervals of values defining a $\sigma$-frequent slice for $\varphi$.
(2) *Step 2.* Compute all $\sigma$-frequent blocks for $\varphi$, based on the intervals obtained at Step 1.
(3) *Step 3.* Among all blocks obtained at Step 2, sort out all non MS-blocks.

We now describe the three steps above in more details.

*Step 1.* For every $i = 1,\ldots,k$, Algorithm 1 and Algorithm 2 compute all maximal intervals $I$ of values in $dom_i$ such that, for every $v$ in $I$, the slice $\mathcal{T}(v)$ is $\sigma$-frequent for $\varphi$. It should be noticed that, while doing so, all levels

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

**Data**: A $k$-dimensional data cube $C$, a fuzzy interval $\varphi$, a support threshold $\sigma$

**Result**: The set of intervals $\mathcal{L}_1(\varphi)$, the set of corresponding blocks $\mathcal{B}_1(\varphi)$

$\mathcal{L}_1(\varphi) \leftarrow \emptyset$

**foreach** *dimension* $d_i$, $i = 1, \ldots, k$ **do**

  **foreach** *level* $p$ *of* $H_i$, $p = 0, \ldots, h_i$ **do**

    $int(\varphi, i, p) \leftarrow \emptyset$

    $currentInterval_p \leftarrow [NIL]$

    $precMember_p \leftarrow NIL$

  **foreach** $j = 1, \ldots, |dom_i|$ **do**

    $currentMember_0 \leftarrow rep_i^{-1}(j)$

    $s_0 \leftarrow supp(\mathcal{T}(currentMember_0), \varphi)$

    $updateInterval(0)$

    $precMember_0 \leftarrow currentMember_0$

    **foreach** *level* $p$ *of* $H_i$ *such that* $p = 1, \ldots, h_i$ **do**

      $currentMember_p \leftarrow h_i^p(currentMember_{p-1})$

      **if** $currentMember_p = precMember_p$ **then**

        $s_p \leftarrow s_p + s_0$

      **else**

        $s_p \leftarrow s_0$

      $updateInterval(p)$

      $precMember_p \leftarrow currentMember_p$

  $\mathcal{L}_1(\varphi) \leftarrow \mathcal{L}_1(\varphi) \cup \left(\bigcup_{p=0}^{p=h_i} int(\varphi, i, p)\right)$

$\mathcal{B}_1(\varphi) \leftarrow \{b = \delta_1 \times \ldots \times \delta_k \mid (\exists i)(\exists p)(\delta_i \in int(\varphi, i, p)$ and $(\forall j \neq i)(\delta_j = ALL_j^0))$ and $conf(b, \varphi) \geq \gamma\}$

**Algorithm 2:** Function updateInterval

**Data**: A level $p$ in the hierarchy $H_i$ associated to dimension $i$

**Result**: The updated values of $currentInterval_p$ and $int(\varphi, i, p)$

**if** $s_p < \sigma$ **then**

  **if** $(currentInterval_p = [\alpha_p, NIL]$ *where* $\alpha_p \neq NIL)$ *and* $(precMember_p \neq currentMember_p)$ **then**

    /* close the current interval with $precMember_p$, and set the current interval to the empty interval */

    $int(\varphi, i, p) \leftarrow int(\varphi, i, p) \cup \{[\alpha_p, precMember_p]\}$

    $currentInterval_p \leftarrow [NIL]$

**else**

  **if** $currentInterval_p = [NIL]$ **then**

    /* start a new current interval with $currentMember_p$ */

    $currentInterval_p \leftarrow [currentMember_p, NIL]$

  **if** $j = |dom_i|$ *and* $currentInterval_p = [\alpha_p, NIL]$ *where* $\alpha_p \neq NIL$ **then**

    $int(\varphi, i, p) \leftarrow int(\varphi, i, p) \cup \{[\alpha_p, currentMember_p]\}$

**Algorithm 3:** Discovery of MS-blocks

**Data**: A $k$-dimensional data cube $C$, a support threshold $\sigma$, and a confidence threshold $\gamma$.

**Result**: The set of $MS$-blocks $\mathcal{B}$ associated with $C$

**foreach** *fuzzy interval* $\varphi$ **do**
- Compute $\mathcal{L}_1(\varphi)$ and $\mathcal{B}_1(\varphi)$
- Let $b_\top = [\top_1] \times \ldots \times [\top_k]$
- /* $b_\top$ can be written as $ALL_1^0 \times \ldots \times ALL_{i-1}^0 \times [\top_i] \times ALL_{i+1}^0 \times \ldots \times ALL_k^0$ for any $i = 1, \ldots, k$ */
- **if** $supp(b_\top, \varphi) < \sigma$ **then**
  - /* This is tested using $\mathcal{L}_1(\varphi)$ */
  - $\mathcal{B}_0(\varphi) \leftarrow \emptyset$
- **else**
  - **if** $conf(b_\top, \varphi) < \gamma$ **then**
    - /* This is tested using $\mathcal{B}_1(\varphi)$ */
    - $\mathcal{B}_0(\varphi) \leftarrow \{b_\top\}$
  - $l \leftarrow 1$
  - **while** $\mathcal{L}_l(\varphi) \neq \emptyset$ **do**
    - $l \leftarrow l + 1$
    - $\mathcal{B}_l(\varphi) \leftarrow \emptyset$
    - /* Generate and prune candidates at level $l$ */
    - $C_l \leftarrow Generate(\mathcal{L}_{l-1}(\varphi))$
    - $\mathcal{L}_l(\varphi) \leftarrow \emptyset$
    - **foreach** $b = \delta_1 \times \ldots \times \delta_k$ *in* $C_l$ **do**
      - **if** $supp(b, \varphi) \geq \sigma$ **then**
        - $\mathcal{L}_l(\varphi) \leftarrow \mathcal{L}_l(\varphi) \cup \{b\}$
        - **if** $conf(b, \varphi) \geq \gamma$ **then** $\mathcal{B}_l(\varphi) \leftarrow \mathcal{B}_l(\varphi) \cup \{b\}$
  - $\mathcal{B}(\varphi) \leftarrow \{b \in \bigcup_{i=0}^{i=l} \mathcal{B}_i(\varphi) \mid b \text{ is an MS-block}\}$

$\mathcal{B} \leftarrow \bigcup_{\varphi} \mathcal{B}(\varphi)$

of the corresponding hierarchy $H_i$ are considered and all maximal intervals of values in $dom_i^j$ are computed for every $j = 0, \ldots, h_i$. In Algorithm 1, the values in $dom_i$ are scanned, and the support of the corresponding slice is computed. For each such value, the support of the slices corresponding to higher levels in $H_i$ are computed accordingly, without any further access to the cube. In Algorithm 2, the intervals are explicitly constructed at all levels of the hierarchy $H_i$, in a bottom-up manner. Moreover, the confidence of the slices are also computed in Algorithm 1, so as to obtain all slices seen as blocks that are $\sigma$-frequent for $\varphi$ and whose confidence is greater than or equal to $\gamma$. We note that computing the confidence of a block knowing its support does not require to access the cube.

It is also important to notice that, if at least one interval is computed for each dimension, then in this case, the set of intervals obtained at this step satisfies

**Algorithm 4:** Function Generate

**Data**: The set $\mathcal{L}_{l-1}(\varphi)$ of blocks $b$ of level $l-1$ such that $supp(b, \varphi) \geq \sigma$

**Result**: The set $C_l$ of block candidates at level $l$

$C_l \leftarrow \emptyset$

**for** *all* $b = \delta_1 \times \ldots \times \delta_k$ *and* $b' = \delta'_1 \times \ldots \times \delta'_k$ *in* $\mathcal{L}_{l-1}(\varphi)$ **do**

  **if** *there exist two distinct integers* $i_1$ *and* $i_2$ *such that for every* $i \neq i_1$ *and* $i \neq i_2$, $\delta_i = \delta'_i$ **then**

    **if** *there exist* $p$ *and* $q$ *such that* $h^p_{i_1}(\delta_{i_1}) \subseteq \delta'_{i_1}$ *and* $h^q_{i_2}(\delta'_{i_2}) \subseteq \delta_{i_2}$ **then**

      Let $b_{12} = \mu_1 \times \ldots \times \mu_k$ be the block such that
      $-$ $\mu_i = \delta_i$, for every $i$ such that $i \neq i_1$ and $i \neq i_2$
      $-$ $\mu_{i_1} = \delta_{i_1}$ and $\mu_{i_2} = \delta'_{i_2}$
      /* Pruning */
      **if** *for every* $i = 1, \ldots, k$, $\mathcal{L}_{l-1}(\varphi)$ *contains a block* $\mu_1 \times \ldots \times \mu_{i-1} \times \nu_i \times \mu_{i+1} \times \ldots \times \mu_k$ *such that, either* $\nu_i = \top_i$, *or for some* $p \in \{1, \ldots, h_i\}$, $h^p_i(\mu_i) \subseteq \nu_i$ **then**
        $C_l \leftarrow C_l \cup \{b_{12}\}$

the three hypotheses of Proposition 2. Indeed:

(1) The first hypothesis is trivially satisfied.
(2) The second hypothesis stating that two distinct intervals at the same level of the hierarchy $H_i$ are disjoint is satisfied thanks to Algorithm 2. Indeed, the domain $dom_i$ is scanned in a linear manner without backtracking and at any level in $H_i$, a new interval is created with a member value occurring after the member value closing the previous interval.
(3) Regarding the third hypothesis, if an interval is obtained at a certain level, then by considering the higher levels of $H_i$ in Algorithm 1, an interval at each of these levels is obtained. This is so because it is easy to see that if, for $v$ in $dom^j_i$ $(0 \leq j \leq h_i - 1)$, $\mathcal{T}(v)$ is $\sigma$-frequent for $\varphi$, then $\mathcal{T}(h^j_i(v))$ is also $\sigma$-frequent for $\varphi$.

Thus, by Proposition 2, the set of all blocks obtained by combining the intervals computed at this step is a lattice.

*Step 2.* The intervals obtained by the previous step are combined in a levelwise manner as shown in Algorithm 3. The first level consists in checking wether the whole cube is $\sigma$-frequent for $\varphi$. We note that this test does not require to scan the cube. Indeed, the cube can be seen as any slice of the form $ALL^0_1 \times \ldots \times ALL^0_{i-1} \times [\top_i] \times ALL^0_{i+1} \times \ldots ALL^0_k$, for any $i = 1, \ldots, k$, and thus, these blocks are computed during the previous step, if they are $\sigma$-frequent for $\varphi$. We also note that, in this case, the confidence of the block is equal to its support, and thus, again, no scan is necessary at this stage. Moreover, if the whole cube is not $\sigma$-frequent for $\varphi$, then the computation stops, due to the anti-monotonicity property of the support (see Proposition 1).
Otherwise, the intervals computed at the previous step are used to form in a

levelwise manner all possible candidates whose support have to be counted. In this computation, the hierarchies over dimensions are considered in a top-down manner. Moreover, when generating the candidates at level $l$, we consider the $\sigma$-frequent blocks for $\varphi$ at level $l-1$ in a similar was as done in [8]. This step is achieved in Algorithm 4 in which a pruning phase is included to sort out all candidates for which one immediate precedessor according to the partial ordering $\sqsubseteq$ is not $\sigma$-frequent for $\varphi$. As in [8], this pruning step is a consequence of the anti-monotonicity of the support (see Proposition 1).
Once all $\sigma$-frequent blocks for $\varphi$ are computed, which requires one scan of the cube for the current level, the confidence of these blocks is checked against the threshold $\gamma$, and this does not require to access the cube.

*Step 3.* Based on the set of all blocks computed in the previous step, only those blocks that are MS-blocks are kept in the output set. To do so, each block is compared to the others with respect to the ordering $\sqsubseteq$, which does not require to access the cube.

### *4.2 Complexity Issues*

In this section, we show that our method for computing blocks is linear in time with respect to the size of the cube. To see this, given a $k$-dimensional cube $C$ with hierarchies $H_1, \ldots, H_k$, we denote by:

- $|C|$ the size of $C$, *i.e.,* the number of cells contained in $C$,
- $h$ the maximal height of the hierarchies $H_1, \ldots, H_k$,
- $N$ the number of fuzzy intervals that are defined over the set $dom_{mes}$ of measure values.

Let $\varphi$ be a fuzzy interval. In Algorithm 1, the cube is scanned once for each dimension $d_i$. Thus, this step requires $k$ scans of the cube $C$. On the other hand, regarding the complexity of Algorithm 3, at each level, the whole cube is scanned at most once in order to compute the support of all candidate blocks. Therefore, this step requires at most a number of scans equal to the maximal height of the considered lattice. As shown in Algorithm 4, the candidates generated at level $l$ are obtained from the $\sigma$-frequent blocks for $\varphi$ in $\mathcal{L}_{l-1}(\varphi)$ by "going down" one level in the hierarchy associated to one of the $k$ dimensions.

Thus, the maximal height of the lattice is $h \,.\, k$, and so, the computation of all $\sigma$-frequent blocks for $\varphi$ requires a number of scans of $C$ in $O(h \,.\, k)$. As computing the confidence of a block knowing its support does not require the scanning of the cube (because the size of a block is the product of the sizes of the intervals defining this block), the time complexity of Algorithm 3 is in $O(h \,.\, k \,.\, |C|)$.

Therefore, considering all fuzzy intervals, it turns out that the computation of the blocks is in $O(N \,.\, h \,.\, k \,.\, |C|)$, *i.e.,* linear with respect to the size $|C|$ of $C$.

We note that the complexity result above does not take into account the fact that the hierarchies $H_1, \ldots, H_k$ have to be scanned during the processing. This is so because we assume these hierarchies to be stored in main memory, which is justified in practice by the fact that the cardinalities of the domains $dom_i$ $(i = 1, \ldots, k)$ is negligeable compared to that of the cube.

### *4.3 Extensions of the Approach*

In this section, we show that the following extensions of our approach can be implemented, based on the algorithms given previously, without increasing their time complexity in the number of scans of $C$.

(1) Instead of considering a fuzzy partition over measure values, assume that a *fuzzy hierarchy* is defined over measure values.
(2) Consider *fuzzy* hierarchies over dimensions, instead of crisp ones.

(1) When considering a fuzzy hierarchy over measure values, which we denote by $H_m$, a naive way to deal with this case could be to apply the algorithms given above at each level of $H_m$. Of course, such an implementation would increase the complexity of the approach, simply because the number of fuzzy intervals over measure values becomes larger in this case.

However, when considering a fuzzy interval $\varphi$ at the lowest level of $H_m$, it is also possible to compute the supports and the confidences of blocks with respect to all fuzzy intervals in $H_m$ that generalize $\varphi$. To this end, given a fuzzy interval $\varphi$ at the lowest level of $H_m$, Algorithm 1 and Algorithm 3 have to be modified so as to compute respectively the intervals and then the $\sigma$-frequent blocks associated to all intervals generalizing $\varphi$. It is important to note that, considering these modifications, the complexity in the scans of the cube $C$ of the algorithms is not changed.

(2) Let us now consider the extension of our approach to the consideration of *fuzzy* hierarchies over dimensions. In this case, given a block $b = \delta_1 \times \ldots \times \delta_k$, the intervals $\delta_1, \ldots, \delta_k$ are *fuzzy* intervals, and thus, the definitions of support and confidence must be modified accordingly.

To this end, we consider a fixed way of fuzzy counting, denoted by $\Sigma^f$ (see Section 3), and we denote membership of a member value $v_i$ to a fuzzy interval $\delta_i$ by $\mu_i(v_i, \delta_i)$. Then, given a cell $c = \langle v_1, \ldots, v_k, m \rangle$ in $C$, membership of $c$ to $b$, denoted by $\mu(c, b)$, is defined by $\mu(c, b) = \bigotimes_{i=1}^{i=k}(\mu_i(v_i, \delta_i))$, where $\otimes$ is a t-norm (usually the minimum or the product, as mentioned in [7]). Using this

notation:

- The *fuzzy* cardinality of $b$, denoted by $|b|^f$, is defined by: $|b|^f = \Sigma^f_{c \in C}(\mu(c, b))$.
- If $\varphi$ is a fuzzy interval of measure values, the count of cells in $b$ whose content is in $\varphi$, denoted by $f\text{-}Count(b, \varphi)$, is defined by: $f\text{-}Count(b, \varphi) = \Sigma^f_{c \in C}(\mu(c, b) \otimes \mu(c, \varphi))$.

In this setting, the *fuzzy-support* and the *fuzzy-confidence* of a block $b$ for $\varphi$, respectively denoted by $f\text{-}supp(b, \varphi)$ and $f\text{-}conf(b, \varphi)$, are then defined as follows:

$$f\text{-}supp(b, \varphi) = \frac{f\text{-}Count(b, \varphi)}{|C|} \qquad \text{and} \qquad f\text{-}conf(b, \varphi) = \frac{f\text{-}Count(b, \varphi)}{|b|^f}.$$

As a consequence, based on the algorithms given above, the extension to fuzzy partitions on the dimensions can easily be implemented by simply changing the ways the supports and confidences are computed. Moreover, it is important to note that, in this case, the obtained algorithms are still linear in the size of $C$. This is so, because knowing the value of $f\text{-}supp(b, \varphi)$, $f\text{-}conf(b, \varphi)$ can be obtained without scanning the cube, since computing $|b|^f$ requires only to scan dimensions.

## 5 Experiments

In this section, we report on experiments in terms of runtime, number of blocks, and rate of overlapping blocks, with and without taking into account hierarchies over dimensions. These experiments are run on synthetical datasets and on a real data set.

### *5.1 Synthetic Data*

For experiments performed on synthetic multidimensional, data have been randomly generated.

Depending on the experiments, the cubes contain up to $10^7$ cells, the number of dimensions ranges from 2 to 9, the number of members per dimension ranges from 2 to 10, and the number of cell values ranges from 5 to $1,000$.

The first experiments report on the impact of taking into account single values, crisp intervals, or fuzzy intervals, when dealing with measure values. It should be noticed in this respect that the case of fuzzy intervals considered in the present paper is the most general one, compared to considering single measure

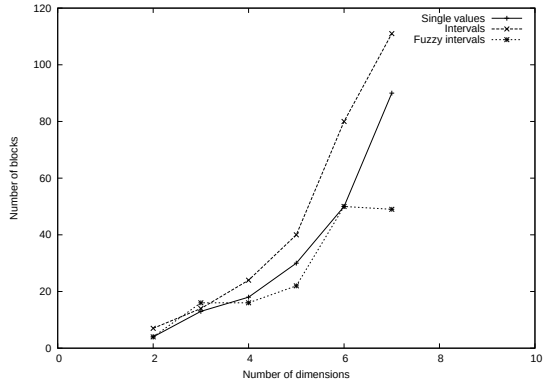


Fig. 3. Number of discovered blocks w.r.t. the number of dimensions

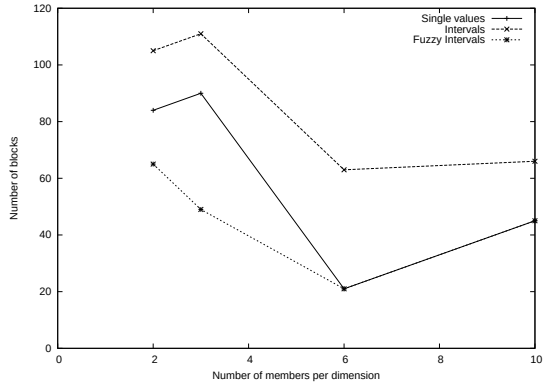


Fig. 4. Number of discovered blocks w.r.t. the number of members

values or crisp intervals of measure values. As shown below, fuzzy intervals lead to more valuable results than single measure values or crisp intervals.

Figure 3 shows the number of blocks output by the three methods (single values, crisp intervals and fuzzy intervals) according to the number of dimensions, and Figure 4 shows the number of blocks output by the three methods (single values, crisp intervals and fuzzy intervals) according to the number of members per dimension.

It should be noted that we obtain more blocks based on intervals than blocks based on single values. This is due to the fact that taking intervals into account increases the chance for a value to match a block value. However, the number of blocks based on fuzzy intervals is lower than the number of blocks based on the other two methods. This is due to the fact that fuzzy blocks can merge several blocks (which would have overlapped, as shown below).

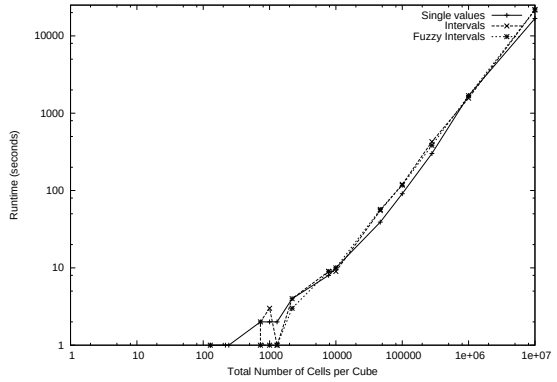


Fig. 5. Runtime w.r.t. the size of the cube

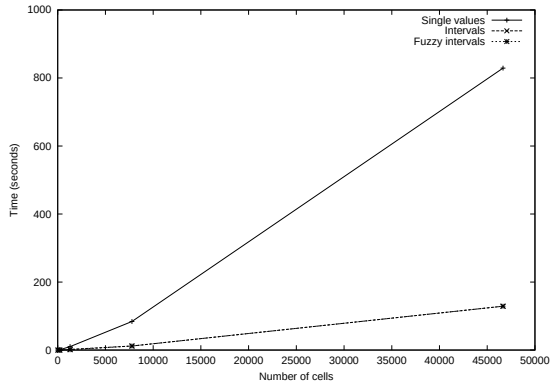


Fig. 6. Runtime w.r.t. the size of the cube

Figures 5 and 6 show the runtime of the three cases (single values, crisp intervals, fuzzy intervals) according to the size of the cube (number of cells). It can be seen that taking crisp and fuzzy intervals into account leads to slightly higher runtimes, if compared with the case of single measure values. However, all runtimes are still comparable and behave the same way.

Figure 7 shows the rate of overlapping blocks depending on the method (single values, crisp intervals of fuzzy intervals) according to the number of dimensions of the cube. This figure suggests that, in the case of this dataset, using crisp methods leads to the fact that many blocks overlap (100% in the case of this experiment), while taking fuzziness into account reduces the rate of overlapping. This fact should be put in relation with the imprecision/uncertainty trade off, *i.e.,* the more certain, the less precise and conversely.

In order to assess the impact of hierarchies, we have performed tests against data sets involving hierarchies having 1 level (*i.e.,* no hierarchy), and then 2

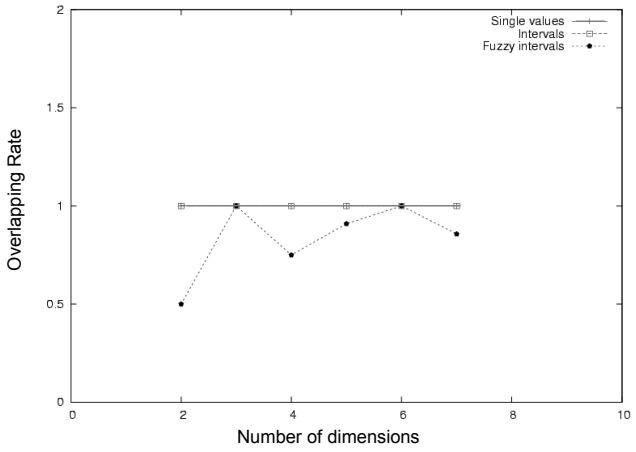


Fig. 7. Rate of Overlapping Blocks w.r.t. the Number of Dimensions

and 3 levels. These levels are successively generated by randomly partitioning the domains of the dimensions into fuzzy intervals. Moreover, these databases contain 5 up to 1,000 distinct measure values, which are treated as fuzzy intervals.

Figure 8 reports on the runtime depending on the size of the cube, while Figure 9 reports on the number of discovered blocks using fuzzy intervals over the measure, with respect to the size of the cube.

It should be noted that we obtain more blocks when considering hierarchies over dimensions. We also note that the number of blocks is even increased when hierarchies are fuzzy. This is due to the fact that, in this case, more cells contribute in building blocks. Of course, this increase in the number of blocks implies an increase in the runtime, as more support computations have to be performed, which is the most time-consuming task in levelwise approaches. We also note that the results shown in Figure 8 and Figure 9 have been obtained after several runs, from which the average have been computed.

### *5.2 Real Data*

The real database we have used is a Trademark Database, storing all registered industrial title deeds. This trademark database stores about 2 million trademarks that have been registered between 1961 and 2004.

This database has been studied for mining sequential patterns in [9]. In this work, the aim was to study the sequences of characters within the names. For

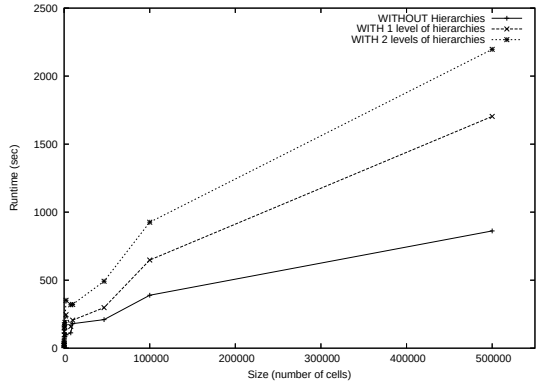


Fig. 8. Runtime w.r.t. the number of levels of hierarchies

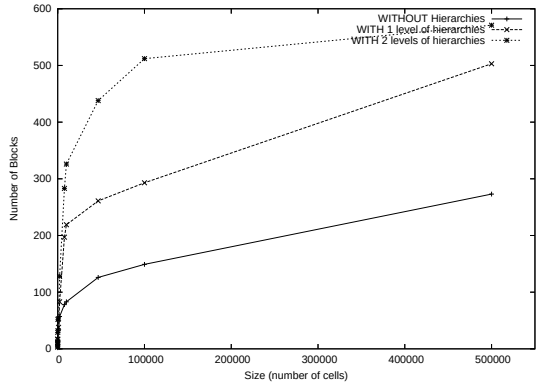


Fig. 9. Number of Blocks w.r.t. the number of levels of hierarchies

instance rules as

$$\langle([\#character, lot])([\#character, lot][\#letter, lot])\rangle \ (33\%)$$

were discovered regarding telecomunication trademarks. Such a rule means that "One third of trademarks consisting of two words contain one word with a *lot* of characters, then one word with a *lot* of characters *among which* a *lot* of letters, given that letters are distinguished from numbers, special characters, punctuations, etc.

In the context of the present paper, we aim at discovering trends in the relationships between lengths, deposit years and categories of trademarks, regarding the number of registered trademarks. To this end, we have built a 3-dimensional cube in which the measure value is the number of title deeds that have been registered. The dimensions of the cube are denoted by: *year*, *length* and *category*. The domain of the dimension *category* is set according to the so called *classification de Nice* that specifies the field of the trademark,

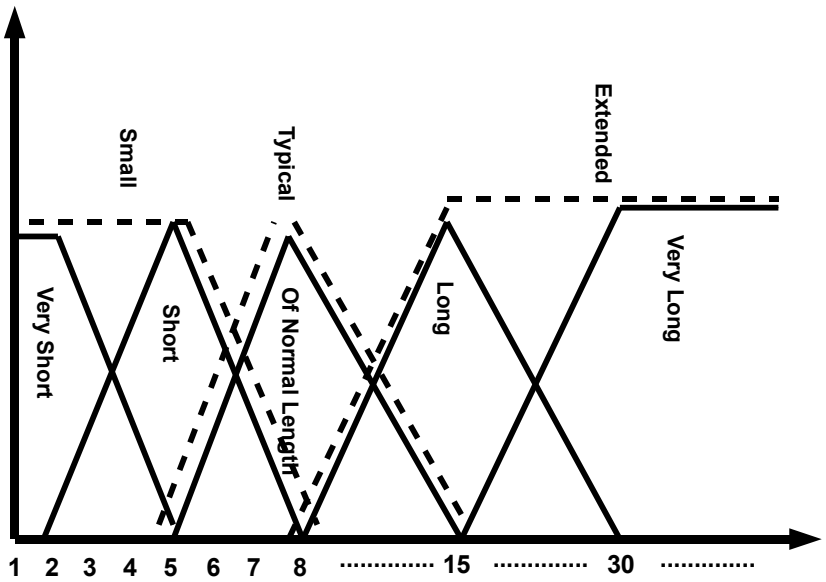


Fig. 10. Fuzzy Hierarchy defined on the Length of a Registered Name

*e.g.,* toys, clothing, shoes, food...

The domains of dimensions *year*, *length* and *category* contain 42, 279 and 45 distinct values, respectively. This results in 527,310 cells, among which 90,113 are not equal to 0 (showing that real multidimensional data can be sparse).

The goal of analyzing such a database is, for the linguistic expert, on the one hand to understand how trademarks impact customers depending on the way they are built, and on the other hand, the evolution of this impact over the years and over the categories of product. The hierarchies over the dimensions *length* and *category* have been defined with 3 levels by the linguistic expert: Over the dimension *length* is defined a *fuzzy* hierarchy that characterizes how a name can be said to be *very short*, *short*, *of normal length*, *long* or *very long* (see Figure 10). On the other hand, the hierarchy defined over dimension *category* contains four values in its intermediate level, namely *Food*, *Sciences and Technology*, *Transportation* and *miscelaneous*.

Analyzing this dataset without considering hierarchies over dimensions produces lot of large blocks, but only associated to measure value 1 or *almost* 1, meaning that no trend could be extracted as only one trademark is registered in these cases. However, taking into account the hierarchies described above, allowed to discover blocks associated to measure values other than 1, which has provided the linguistic expert with relevant knowledge in the data analysis.

It is also important to note that the blocks have been built up based on fuzzy intervals over the measure value, because no block, apart from those associated with measure value 1, could be discovered using individual measure values.

As an example of discovered block, we mention the following:

*Around year* 2000*, for all categories, the number of trademarks registered with a long name is almost 300.*

We emphasize in this respect that no block was discovered for measure value *almost* 300 when no hierarchies were taken into account, thus preventing the linguistic expert from discovering such trends.

## 6 Related Work

The work on building blocks of similar values in a given data cube as presented in this paper can be related to the work on data clustering of high dimensional data, which is an important area in data mining. However, it is important to note that, to the best of our knowledge, no other work considers hierarchies over dimensions and fuzzy intervals over measure values, as we do in this paper.

For a large multidimensional database where the data space is usually not uniformly occupied, data clustering identifies the sparse and dense areas and thus discovers the overall distribution patterns (or summary) of the dataset. Some examples of work on clustering of high dimensional data include CLARANS [10], BIRCH [11], CLIQUE [12] and CURE [13].

Several subspace clustering methods are introduced to detect clusters residing in different subspaces (*i.e.,* subsets of the original dimensions). In this case, no new dimension is generated. Each resultant cluster is associated with a specific subspace. Some examples of these methods are CLIQUE [12] and ORCLUS [14].

CLIQUE (CLustering In QUEst) adopts a density-based approach to clustering in which a cluster is defined as a region that has higher density of points than its surrounding area. To approximate the density of data points, the data space is partitioned into a finite set of cells. Note that a block in our work is almost similar to the concept *unit* in [12] which is obtained by partitioning every dimension into intervals of equal length. Thus a unit in the subspace is the intersection of an interval from each of the $k$ dimensions of a $k$-dimensional cube. However, contrary to our approach, the construction of the blocks in [12] is not determined by the same measure value, but rather by arbitrary chosen partitions of the member values. We notice again that in [12], hierarchies over dimensions are not considered.

Research work on (fuzzy) image segmentation may appear as related works

[15]. Although the goals are the same, it is not possible to apply such methods in the case of multidimensional databases, due to problems of scalability and because also of the multidimensional nature of data. For example, clustering-based color image segmentation [16] is normally limited to a 2-dimensional environment with the possibility of an extension to 3 dimensions. Again, in this case, hierarchies over dimensions have not been considered.

Segmentation methods (*e.g.,* clustering) have been proposed in the multidimensional context [12], [17]. In [18], the authors study the generation of fuzzy partitions over numerical dimensions. However, these propositions are not related to our measure-based approach, and thus these propositions are different from our work where the measure value is the central criterion.

On the other hand, the feature selection methods are used to select a subset of dimensions for supervised classification problem [19]. The idea is to produce an *optimal* pool of *good* dimension subsets for searching clusters. Therefore, in this approach, clusters are built up according to criteria related to dimensions whereas in our approach, blocks are built up according to similarrity criteria on the measure values.

In [20] the authors aim at compressing data cubes. However there is no consideration on cube representations and homogeneous blocks generation.

The work presented in [21] 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 areas, which may not cover the whole cube.

In [22], the authors propose the concept of *condensed* data cube. However, the authors aim at considering the cube without loss of information, while we aim at displaying relevant information to the user, which may be a partial representation of data.

## 7 Conclusion

In this paper, we have considered an approach to summarize a data cube by means of multiple-level fuzzy blocks. Fuzziness comes from the fact that we assume that measure values in the cube are partitioned according to a fuzzy partition. On the other hand, blocks are defined at different levels of granularity, according to predefined hierarchies over the dimensions. These blocks are computed using a levelwise algorithm similar to Apriori ([8]), and the preliminary tests reported in the paper show the effectiveness of our approach.

We are currently considering further tests involving hierarchies so as to better measure the impact on performance and on the overall quality of the output blocks. Moreover, we are implementing the two extensions mentioned in the paper; while doing so, we are investigating possible optimizations of our algorithms.

We also intend to investigate the following research directions. First, the quality of the blocks output by our method will be assessed according to measure qualities other than support and confidence, based on [6,7]. Second, as we compute most specific blocks, we think that it would be relevant to consider algorithms such as MaxMiner ([23]), or ZigZag ([24]) that have been designed to this end. However, a deeper study of the structure of the search space in our approach is necessary in order to adapt efficiently these algorithms to our case.

## Acknowledgements

This work was partially supported by the French Ministry of Foreign Affairs under the STIC-Asia EXPEDO project. Moreover, the authors wish to thank the French Embassy in Malaysia for its valuable support.

## References

[1] Y. Choong, D. Laurent, A. Laurent, Building fuzzy blocks from data cubes, in: Proc. of Int. Conf. IPMU'06, 2006.

[2] Y. Choong, D. Laurent, A. Laurent, Summarizing multidimensional databases using fuzzy rules, in: Proc. of Int. Conf. IPMU'04, 2004, pp. 99–106.

[3] Y. Choong, D. Laurent, A. Laurent, Summarizing data cubes using blocks, in: Data Mining Patterns: New Trends and Applications, IDEA Group Inc., 2007.

[4] Y. Choong, D. Laurent, A. Laurent, Pixelizing data cubes: a block-based approach, in: Proc. of Visual Information Expert Workshop (VIEW), Vol. 4370 of LNCS, Springer-Verlag, 2007, pp. 63–76.

[5] Y. Choong, D. Laurent, P. Marcel, Computing appropriate representation for multidimensional data, Data and Knowledge Engineering Int. Journal 45 (2003) 181–203.

[6] D. Dubois, E. Hüllermeier, H. Prade, A note on quality measures for fuzzy association rules, in: Proc. of Int. Fuzzy Systems Association World Congress on Fuzzy Sets and Systems, LNAI 2715, 2003, pp. 346–353.

[7] D. Dubois, E. Hüllermeier, H. Prade, A sytematic approach to the assessment of fuzzy association rules, Data Mining and Knowledge Discovery 13 (2006) 167–192.

[8] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: Proc. of ACM SIGMOD, 1993, pp. 207–216.

[9] C. Fiot, A. Laurent, M. Teisseire, B. Laurent, Why Fuzzy Sequential Patterns can Help Data Summarization: an Application to the INPI Trademark Database, in: Proc. of the 15th IEEE International Conference on Fuzzy Systems, 2006, pp. 699–706.

[10] R. T. Ng, J. Han, Clarans: A method for clustering objects for spatial data mining, IEEE Transactions on Knowledge and Data Engineering 14 (5) (2002) 1003–1016.

[11] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in: Proc. of ACM SIGMOD, ACM Press, 1996, pp. 103–114.

[12] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proc. of ACM SIGMOD, 1998, pp. 94–105.

[13] S. Guha, R. Rastogi, K. Shim, Cure: An efficient clustering algorithm for large databases., in: Proc. of ACM SIGMOD, 1998, pp. 73–84.

[14] C. Aggarwal, P. Yu, Finding generalized projected clusters in high dimensional space, in: Proc. of ACM SIGMOD, 2000, pp. 70–81.

[15] S. Philipp-Foliguet, M. B. Vieira, M. Sanfourche, Fuzzy segmentation of color images and indexing of fuzzy regions, in: Proc. of CGIV, 2002, pp. 507–512.

[16] R. Turi, Clustering-based colour image segmentation, Ph.D. thesis, Monash University (Australia) (2001).

[17] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu, Incremental clustering for mining in a data warehousing environment, in: Proc. of Int. Conf. on Very Large Data Bases (VLDB), 1998, pp. 323–333.

[18] A. Gyenesei, A fuzzy approach for mining quantitative association rules, Tech. Rep. 336, Turku Center for Computer Science (TUCS) (2000).

[19] H. Motoda, L. H. Liu, Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, 1998.

[20] L. Lakshmanan, J. Pei, J. Han, Quotient cube: How to summarize the semantics of a data cube, in: Proc. of Int. Conf. on Very Large DataBases (VLDB), 2002, pp. 778–789.

[21] D. Barbara, M. Sullivan, Quasi-cubes: Exploiting approximation in multidimensional databases, SIGMOD Record 26 (1997) 12–17.

[22] W. Wang, H. Lu, J. Feng, J. X. Yu, Condensed cube: An effective approach to reducing data cube size, in: Proc. of Int. Conf. on Data Engeneering (ICDE), 2002, pp. 155–165.

[23] R. Bayardo, Efficiently mining long patterns from databases, in: Proc. of ACM SGMOD, 1998, pp. 85–93.

[24] F. D. Marchi, F. Flouvat, J. Petit, Adaptive strategies for mining the positive border of interesting patterns: Application to inclusion dependencies in databases, in: Constraint-Based Mining and Inductive Databases, LNCS 3848, Springer-Verlag, 2004, pp. 81–101.