# From Crispness to Fuzziness: Three Algorithms for Soft Sequential Pattern Mining

Céline Fiot, Anne Laurent, Maguelonne Teisseire

# From Crispness to Fuzziness: Three Algorithms for Soft Sequential Pattern Mining

Céline Fiot, *Student Member, IEEE,* Anne Laurent, Maguelonne Teisseire

***Abstract*— Most real world databases consist of historical and numerical data such as sensor, scientific or even demographic data. In this context, classical algorithms extracting sequential patterns, which are well adapted to the temporal aspect of data, do not allow numerical information processing. Therefore the data are pre-processed to be transformed into a binary representation, which leads to a loss of information. Fuzzy algorithms have been proposed to process numerical data using intervals, particularly fuzzy intervals, but none of these methods is satisfactory. Therefore this paper completely defines the concepts linked to fuzzy sequential pattern mining. Using different fuzzification levels, we propose three methods to mine fuzzy sequential patterns and detail the resulting algorithms (SpeedyFuzzy, MiniFuzzy and TotallyFuzzy). Finally, we assess them through different experiments, thus revealing the robustness and the relevancy of this work.**

***Index Terms*— Sequential Patterns, Numerical Data, Fuzzy Intervals.**

## I. Introduction

MOST real world databases consist of historically-stamped and numerical data (sensor, scientific, demographic, monitoring, evolution phenomena data, etc.). Within the context of large-scale database mining, few studies have been carried out to process this kind of data and most work is restricted to association rules [1]–[7]. A first proposal [1] processes quantitative data for association rule mining using the attribute discretization within crisp intervals. However, some frequent associations could be lost because of too restrictive bounds. Recently, the use of fuzzy set theory has reduced stark cuts between intervals, thus leading to more relevant rules. [2]–[4] presented a new definition of support and confidence based on fuzzy set theory, in the context of association rule extraction from quantitative attributes. Rules will go from "75% of people who buy butter also purchase bread" to the new type "60% of people who eat *a lot of* candies purchase *few* potato chips".

These association rule based approaches do not allow easy and complete processing of temporal data. Their extension to sequential pattern mining methods takes the temporal or ordered aspect of data into account, so it is well adapted to historically-stamped data. However, the first propositions [8], [9] do not allow processing of numerical data and require pre-processing of these data into a binary representation, which necessarily leads to a loss of information.

More recently, [10] and [11]–[14] presented two methods to extract fuzzy sequential patterns to mine sequences within historically-stamped quantitative data. These works were the very first proposals concerning this original mining method. However, these approaches can be generalized and improved. These proposals have indeed introduced several frameworks with drawbacks, the first one being too restrictive, the second one rather ambiguous.

We can indeed extract different kinds of knowledge from quantitative attributes. Sequential patterns are characterized by their support or frequency, which indicates the proportion of sequences in a database that contains this pattern. In the case of quantitative sequences, one can mine for different information: mining for the presence and frequency of certain quantitative attributes or sequences, mining for the frequency of these sequences with minimum relevancy, or mining for a weighted frequency according to the relevancy of the quantitative sequence.
In this paper, we present a framework to take these different kinds of knowledge into account through the extraction of fuzzy sequential patterns. This framework extends and generalizes previous proposals concerning fuzzy sequential pattern mining approaches. We detail a complete and efficient fuzzy approach for sequential pattern mining which enables processing of numerical data, with different levels of consideration of numerical information.
Our approach is based on the definition of *fuzzy* intervals. Obtained patterns are of the type "60% of people purchasing *a lot of* candies and *few* video games buy later *a lot of* toothpaste". These patterns are characterized by their frequency (or support), which is by definition the proportion of customers who have bought these products. We define three approaches SpeedyFuzzy, MiniFuzzy and TotallyFuzzy that differ from each other by their evaluation of the frequency and thus extract different levels of information. The end-user is allowed to choose between the speed of result extraction and the accuracy of the obtained frequent patterns. Implementation of the different solutions is based on an efficient and original data scanning method, which extends the PSP algorithm proposed by F. Masseglia et al. [15]. Experiments were carried out on synthetic datasets. They highlight the feasibility of a fuzzy approach and its robustness. Furthermore our algorithms were assessed on different real-world datasets.

The authors are with the Laboratory of Computer Science, Robotics, and Microelectronics, University of Montpellier II - CNRS, France. Email: {fiot, laurent, teisseire}@lirmm.fr

This paper is structured as follows: Section II presents an introduction to sequential patterns and describes the shortcomings in former proposals of fuzzy sequential patterns. Section III presents a different way to assess the sequence frequency. Then functions calculating the frequency for each method are detailed in Section IV with their involvement in database scanning. TOTALLYFUZZY, the most complex algorithm is detailed along with its computational complexity. Then Section V describes the overall algorithm to mine fuzzy sequential patterns upon which our three algorithms SPEEDYFUZZY, MINIFUZZY and TOTALLYFUZZY are based, while providing a short example to describe how it works. Section VI presents the implementation of the algorithms and the experiments carried out on synthetic datasets and also on real-world data. Finally, in Section VII we conclude on the advantages of the different information levels and present different prospects generated by this work.

## II. FROM SEQUENTIAL PATTERNS TO FUZZY SEQUENTIAL PATTERNS

In this section, we briefly describe the basic concepts of sequential patterns and then take a look at earlier proposals on fuzzy sequential patterns, highlighting their weaknesses.

### A. Sequential Patterns

Let $\mathcal{R}$ be a set of objects records where each record R consists of three information elements: an object-id, a record timestamp and a set of attributes/items in the record.
Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items or attributes. An *itemset* is a non-empty set of attributes $i_k$, denoted by $(i_1 i_2 \ldots i_k)$. It is a non-ordered representation. A *sequence* $s$ is a non-empty ordered list of itemsets $s_p$, denoted by $< s_1 s_2 ... s_p >$. A *n-sequence* is a sequence of $n$ items (or of size $n$).

*Example 1:* Let us consider an example of market basket analysis. The object is a customer, and records are the transactions made by this customer. Timestamps are the date of transactions. If the customer Smith purchases products $1, 2, 3, 4,$ and $5$ made by the customer Smith according to the sequence $s =<$ (1) (2 3) (4) (5) $>$, then all items of the sequence were bought separately, except products 2 and 3 which were purchased at the same time. In this example, $s$ is a 5-sequence.

One sequence $< s_1\ s_2 ... s_p >$ is a *subsequence* of another one $< \ s'_1\ s'_2\ ... s'_m >$ if there are integers $l_1 < l_2 < ... < l_p$ such that $s_1 \subseteq s'_{l_1},\ s_2 \subseteq s'_{l_2}, ...,\ s_p \subseteq s'_{l_p}$.

*Example 2:* The sequence $s'$ = <(2) (5)> is a subsequence of $s$ because (2) $\subseteq$ (2 3) and (5) $\subseteq$ (5). However, <(2) (3)> is not a subsequence of $s$.

All records from the same object are grouped together and sorted in increasing order of their timestamp. They are called a data sequence. An object *supports* a sequence $s$ if it is included within the data sequence of this object ($s$ is a subsequence of the data sequence). The *frequency* of a sequence ($freq(s)$) is defined as the percentage of objects supporting $s$. In order to decide whether a sequence is frequent or not, a minimum frequency value ($minFreq$) is specified by the user and the sequence is said to be frequent if the condition $freq(s) \geq minFreq$ holds. Given a database of object records, the problem of sequential pattern mining is to find all maximal sequences whose the frequency is greater than a specified threshold ($minFreq$) [8]. Each of these sequences represents a sequential pattern, also called a maximal frequent sequence. Note that items are processed using a simple binary evaluation – present or not present.

### B. Fuzzy Sequential Patterns

In order to mine fuzzy sequential patterns, the universe of each quantitative item is partitioned into several fuzzy sets.
T.-P. Hong et al. [10] presented the first proposal of a fuzzy sequential pattern mining approach. Their proposal is based on discretization of data into fuzzy intervals. However, to minimize the number of items being processed, in a first step, the mining algorithm selects the fuzzy items that would be kept for mining sequential patterns. So it only keeps, for each item, the fuzzy set having the highest cardinality over the whole database (by $\Sigma$-count).

*Example 3:* Let us consider the database shown in Table I describing the purchases of 3 customers for the attribute $Candy$.

TABLE I
TRANSACTIONS FOR THE FUZZY PARTITION $Candy$

| Customer | Date | $little$ | $medium$ | $lot$ |
|---|---|---|---|---|
| C1 | d1 | 0.6 | 0.4 | 0 |
| C1 | d2 | 0 | 0.7 | 0.3 |
| C1 | d3 | 1 | 0 | 0 |
| **Max(C1)** | | 1 | 0.7 | 0.3 |
| C2 | d1 | 0 | 0 | 0 |
| C2 | d2 | 0 | 0.5 | 0.5 |
| C2 | d3 | 0 | 0.3 | 0.7 |
| **Max(C2)** | | 0 | 0.5 | 0.7 |
| C3 | d1 | 0 | 0.6 | 0.4 |
| C3 | d2 | 0.9 | 0 | 0 |
| C3 | d3 | 0 | 0.2 | 0.8 |
| **Max(C3)** | | 0.9 | 0.6 | 0.8 |
| **Count** | ($\Sigma(Max)$) | **1.9** | **1.8** | **1.8** |

In this case, the only column that will be taken into account in [10] for the item $Candy$ is $little$, whereas it is not the quantity that is the most present within the data sequences (it is actually $medium$ or $lot$).

It is thus obvious that such a decision is reductive and can lead to erroneous rules. The consequences of this strategy on the discovery of sequential patterns are highlighted in our experiments, section VI-D. If we consider that the selection step could be avoided, it is not clearly expressed. Moreover, it would certainly have consequences on the following part of the algorithm.

Y.-C. Hu et al. [11]–[14] adopted a theoretical approach presenting an algorithm without the underlying data structures nor any experimental assessment. Moreover, their proposal has ambiguous formalism and notations for the frequency computation of a fuzzy itemset and thus for a fuzzy sequence. It is especially hard to identify differences between the frequency computation of the sequences <(a)(b)(c)> and <(a)(b c)>. This point is, however, fundamental in the context of sequential pattern mining because the timestamps associated to the items play an important role.

Finally, clearly none of these previous works consider the different levels of information that could be obtained by considering a global framework for fuzzy sequential pattern mining and the different levels of fuzzification that could be used to assess the frequency of a sequence. Therefore we give a generalized definition for the frequency of a sequence and propose three algorithms to mine fuzzy sequential patterns.

## III. Proposal: from Crispness to Fuzziness

First, we detail the formalism for our fuzzy sequential pattern framework. We give a clear and non-ambiguous definition of the basic concepts (item, itemset, $g$-$k$-sequence and fuzzy frequency). The core of this paper is a proposal of three algorithms with three levels of fuzzification, depending on the counting method adopted to determine the frequency of sequences or itemsets. The counting methods considered are based on the four classical ways to compute fuzzy cardinality of a fuzzy subset A:

1) counting all elements for which the membership degree is not null ($card(supp(A))$);
2) considering only elements for which the membership degree is greater than a defined threshold. This counting method is called *thresholded count* ($card(supp(A_\alpha))$);
3) adding the membership degrees of each element. This counting method is called Σ-count ($\sum_{x \in supp(A)} \mu_A(x)$);
4) adding membership degrees greater than a defined threshold $\alpha$, this is called *thresholded Σ-count* ($\sum_{x \in supp(A)} \mu_{A_\alpha}(x)$).

Each algorithm is defined through a generic one, SMT-Fuzzy, presented Section V. Each of them corresponds to one precise frequency computation, thus allowing three levels of fuzzification during the mining of fuzzy sequential patterns, based on one approach given in [16], thus justifying the use of some fuzzy logical operators for fuzzy association rule frequency measurement.

### A. Preamble: item, itemset, $g$-$k$-sequence

The item and itemset concepts have been redefined relative to classical sequential patterns.

*Definition 1:* A **fuzzy item** is the association of one item and one corresponding fuzzy set. It is denoted by $[x, a]$ where $x$ is the item (also called attribute) and $a$ is the associated fuzzy set.

*Example 4:* $[candy, lot]$ is a fuzzy item where $lot$ is a fuzzy set defined by a membership function on the quantity universe of the possible purchases of the item $candy$.

*Definition 2:* A **fuzzy itemset** is a set of fuzzy items. It can be denoted as a pair of sets (set of items, set of fuzzy sets associated to each item) or as a list of fuzzy items.
We use the following notation: $(X, A)$, where $X$ is a set of items and $A$ is a set of corresponding fuzzy sets.

*Example 5:* $(X, A) = ([candy, lot][soda, little])$ is a fuzzy itemset and can also be denoted by $((candy, soda)(lot, little))$.

One fuzzy itemset only contains one fuzzy item related to one single attribute. For example, the fuzzy itemset $([candy, lot][candy, little])$ is not a valid fuzzy itemset because it contains twice the attribute $candy$.

Lastly, we define a $g$-$k$-sequence.

*Definition 3:* A **$g$-$k$-sequence** $S =< s_1 \cdots s_g >$ is a sequence constituted by $g$ fuzzy itemsets $s = (X, A)$ grouping together $k$ fuzzy items $[x, a]$.

*Example 6:* The sequence $S =< ([soda, lot][candy, lot])$ $([video\ games, little]) >$ groups together 3 fuzzy items into 2 itemsets. It is a fuzzy 2-3-sequence.

In the next sections of this article, we use the following notations: let $\mathcal{O}$ represent the set of objects and $\mathcal{R}_o$ the set of records for one object $o$. Let $\mathcal{I}$ be the set of attributes or items and $r[i]$ the value of attribute $i$ in record $r$. Each attribute $i$ is divided into fuzzy sets.

Here we apply generic definitions of objects and attributes to the supermarket basket analysis, with one object being a customer and the products standing for the attributes. For example, we use the dataset described in Table II, in which an empty cell indicates that the product has not been purchased.

First, the quantitative database is converted into a membership degree database. Each attribute is partitioned into several fuzzy sets, as shown in Fig. 1, which represents the membership functions for each attribute. These partitions are automatically built by dividing the universe of quantities into intervals. Each interval groups the same proportion of customers. It is then fuzzified in order to enhance generalization. From these membership functions, we get the membership degrees for each transaction and for each fuzzy set. Table III describes these values for customer 1.
This example will be used below to illustrate the computation of the different frequencies for the itemset $Ix = ([candy, little][soda, lot])$ and the sequence $Sx = < ([candy, little])([soda, lot]) >$.

TABLE II
TRANSACTIONS GROUPED BY CUSTOMERS AND ORDERED BY THEIR TIMESTAMP

| Customers | Date | Items: candy | toothpaste | soda | ball | videogame |
|---|---|---|---|---|---|---|
| C1 | d1 | 4 | | 1 | | |
| | d2 | 3 | | | | |
| | d3 | 5 | 3 | 1 | | |
| | d4 | 2 | | 1 | | |
| | d5 | | | 1 | 5 | |
| | d6 | 2 | | 2 | | 2 |
| C2 | d1 | 2 | | 2 | 1 | |
| | d2 | | | 2 | | |
| | d3 | | 4 | 1 | | |
| | d4 | 4 | | | | |
| C3 | d1 | | | | | 3 |
| | d2 | 3 | 1 | 2 | | |
| | d3 | | | | 4 | 5 |
| | d4 | | | 2 | | |
| | d5 | | 2 | | | |
| C4 | d3 | 2 | | 2 | | 4 |
| | d4 | | | 3 | | |
| | d5 | | 2 | | | |
| | d6 | | | 2 | | |

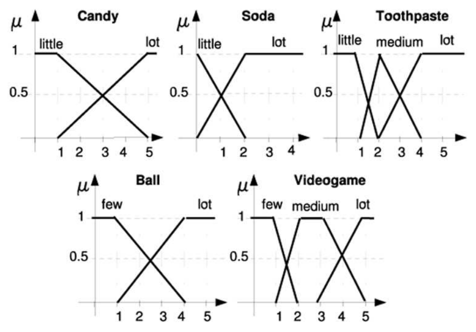

Fig. 1. Membership functions for fuzzy sets of the database attributes

TABLE III
MEMBERSHIP DEGREES FOR CUSTOMER 1

| | Items | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | candy | | toothpaste | | | soda | | ball | | videogame | | |
| D. | li. | L. | li. | m | L. | li. | L. | f. | L. | f. | m. | L. |
| d1 | 0.25 | 0.75 | | | | 0.5 | 0.5 | | | | | |
| d2 | 0.5 | 0.5 | | | | | | | | | | |
| d3 | | 1 | | 0.5 | 0.5 | 0.5 | 0.5 | | | | | |
| d4 | 0.75 | 0.25 | | | | 0.5 | 0.5 | | | | | |
| d5 | | | | | | 0.5 | 0.5 | | 1 | | | |
| d6 | 0.75 | 0.25 | | | | | 1 | | | | 1 | |

TABLE IV
MEMBERSHIP DEGREES FOR CUSTOMER 1 AND FUZZY FREQUENCIES FOR ITEMSET $Ix$

| | Items | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | candy | | toothpaste | | | soda | | ball | | videogame | | |
| D. | li. | L. | li. | m | L. | li. | L. | f. | L. | f. | m. | L. |
| d1 | $\underline{0.25}$ | 0.75 | | | | 0.5 | $\underline{0.5}$ | | | | | |
| d2 | 0.5 | 0.5 | | | | | | | | | | |
| d3 | | **1** | | 0.5 | 0.5 | 0.5 | **0.5** | | | | | |
| d4 | 0.75 | 0.25 | | | | 0.5 | 0.5 | | | | | |
| d5 | | | | | | 0.5 | 0.5 | | 1 | | | |
| d6 | $\underline{\underline{0.75}}$ | 0.25 | | | | | $\underline{\underline{1}}$ | | | | 1 | |

## B. Three Frequency Computation Methods

The **frequency of a fuzzy itemset** is computed as the number of objects supporting this fuzzy itemset compared to the total number of objects in the database:

$$FFreq_{(X,A)} = \frac{\sum_{o \in \mathcal{O}} \Big[ F(o,(X,A)) \Big]}{|\mathcal{O}|} \tag{1}$$

where the frequency degree $F(o,(X,A))$ marks whether the object $o$ supports the fuzzy itemset $(X,A)$ or not.
As previously presented, the cardinality of a fuzzy set depends on the counting method. We transpose here three of these techniques in the framework of fuzzy sequential patterns and propose three definitions for the fuzzy frequency. Fig. IV highlights the frequency found for $Ix$ in each case:

● SPEEDYFUZZY is based on the count "supports / does not support" (1). Computing the frequency of a fuzzy itemset consists of counting all objects that recorded the itemset at least once. Regardless of the membership degree of the record for the fuzzy itemset, if it is greater than zero, the object has the same weight:

$$F_{SF}(o,(X,A)) = \begin{cases} 1 & \text{if } \exists r \in \mathcal{R}_o | \forall [x,a] \in (X,A),\ \mu_a\big(r[x]\big) > 0 \\ 0 & \text{else} \end{cases} \tag{2}$$

*Example 7:* With the SPEEDYFUZZY count, the first customer supports the itemset $Ix$ since one transaction is found containing the regarded itemset with a membership degree greater than zero, this is transaction d1 on Table IV.

● MINIFUZZY is based on a thresholded count. This method only increments the number of objects supporting the fuzzy itemset when each item of the candidate sequence has a membership degree greater than a specified threshold in the data sequence of the object:

$$F_{MF}(o,(X,A)) = \begin{cases} 1 & \text{if } \exists r \in \mathcal{R}_o | \forall [x,a] \in (X,A),\ \mu_a\big(r[x]\big) > \omega \\ 0 & \text{else} \end{cases} \tag{3}$$

*Example 8:* With a MINIFUZZY count, customer 1 supports the itemset $Ix$ since a transaction is found containing the items with a membership degree greater than the threshold ($\omega$=0.49). These correspond to transaction d3 in Table IV.

● TOTALLYFUZZY carries out a thresholded $\Sigma$-count. In this approach, the importance of each fuzzy itemset in the data sequence is taken into account in the support computation. To do so, the threshold membership function $\alpha$ is defined as:

$$\alpha_a\big(r[x]\big) = \begin{cases} \mu_a\big(r[x]\big) & \text{if } \mu_a\big(r[x]\big) > \omega \\ 0 & \text{else} \end{cases} \tag{4}$$

The frequency counting formula becomes:

$$F_{TF}(o,(X,A)) = \underline{\perp}_{j=1}^{|\mathcal{R}_o|} \overline{\top}_{[x,a] \in (X,A)} \Big[ \alpha_a\big(r_j[x]\big) \Big] \tag{5}$$

where $\overline{\top}$ and $\underline{\perp}$ are the generalized t-norm and t-conorm operators.
Note that the $\Sigma$-count is a thresholded $\Sigma$-count, with a threshold $\omega$=0.

*Example 9:* With a TOTALLYFUZZY count, customer 1 supports the itemset $Ix$ if a transaction is found containing the fuzzy itemset, with a membership degree greater than the threshold $\omega$. The best value for the itemset is kept. In our example, this itemset is supported by d6 in Table IV.

The **frequency of a fuzzy $g$-$k$-sequence** is computed as the ratio of the number of objects supporting this fuzzy sequence compared to the total number of objects in the database:

$$FFreq_{(X,A)} = \frac{\sum_{o \in \mathcal{O}} \Big[ F(o, gS) \Big]}{|\mathcal{O}|} \qquad (6)$$

where the frequency degree $F(o, gS)$ indicates whether the object $o$ supports the fuzzy sequence $gS$. This frequency degree is computed using the algorithm *CalcFuzzyFreq*, Algorithm 1, which calls the function *FindFuzzySeq*, which is either $FindSpeedySeq$, $FindMiniSeq$ or $FindTotallySeq$ depending on the expected information level. These functions are detailed in Sections IV.

```
ALGORITHM 1 - CalcFuzzyFreq
Input: $gS$, candidate $g$-$k$-sequence ;
Ouput: $FFreq$ fuzzy frequency for the sequence $g$S ;

$FFreq, nbFreq, m \leftarrow 0$ ;
For each object $o \in \mathcal{O}$ do
    $m \leftarrow$ FindFuzzySeq($g$-S, $\mathcal{R}_o$);
    $nbFreq$ += $m$ ;
End For
$FFreq \leftarrow nbFreq/|\mathcal{O}|$;
return $FFreq$;
```

To illustrate these methods, we again use the membership degrees of customer 1's transactions. The different frequencies of the sequence $Sx$=$<([candy, little])([soda, lot])>$ are highlighted in Table V.

TABLE V
MEMBERSHIP DEGREES FOR CUSTOMER 1 AND FUZZY FREQUENCIES FOR SEQUENCE $Sx$

| | **Items** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | candy | | toothpaste | | | soda | | ball | | videogame | | | |
| **D.** | li. | L. | li. | m | L. | li. | L. | f. | L. | f. | m. | L. |
| d1 | $\underline{0.25}$ | 0.75 | | | | 0.5 | 0.5 | | | | | |
| d2 | **0.5** | 0.5 | | | | | | | | | | |
| d3 | | 1 | | 0.5 | 0.5 | 0.5 | $\underline{\mathbf{0.5}}$ | | | | | |
| d4 | $\underline{\underline{0.75}}$ | 0.25 | | | | 0.5 | 0.5 | | | | | |
| d5 | | | | | | 0.5 | 0.5 | | 1 | | | |
| d6 | 0.75 | 0.25 | | | | | $\underline{\underline{1}}$ | | | | 1 | |

## IV. COMPUTING A SEQUENCE FREQUENCY

Depending on the algorithm used to mine fuzzy sequential patterns, SPEEDYFUZZY, MINIFUZZY or TOTALLYFUZZY, the difference will depend on the frequency computation.

### A. Using SpeedyFuzzy and MiniFuzzy Algorithms

The way the algorithms SPEEDYFUZZY and MINIFUZZY work is quite similar to computing the sequence frequency. For each object, it is a matter of scanning the record set to find the candidate sequence. For each itemset in the sequence, it is necessary to check whether the membership degree is not zero for SPEEDYFUZZY or greater than the threshold $\omega$ for MINIFUZZY. As soon as the candidate sequence has been validated (the ordered set of items is supported by the object), scanning of the object records is stopped and the frequency of the sequence is incremented.

The SPEEDYFUZZY count is based on the function $FindSpeedySeq$, Algorithm 2, which searches for a candidate sequence in the record set of an object. It involves searching for each itemset of the sequence, following their order. When the first one is found within a record with a degree greater than zero, the following itemset is searched for in the following record and so on.
In Fig. V, transactions that validate sequence $Sx$ with a SPEEDYFUZZY count are d1 followed by d3. This corresponds to the underlined items.

```
ALGORITHM 2 - FindSpeedySeq
Input: $gS$, candidate $g$-$k$-sequence; $\mathcal{R}$, record set to run
Ouput: $found$, 1 if $g$S is found, 0 else

$currIS \leftarrow g$S.first() ;
$found \leftarrow 0$ ;
$r \leftarrow \mathcal{R}$.first() ;
While ((($\underline{!}found$) || ($r$!=$\varnothing$))) do
    If (($\top_{[x,a] \in currIS} \alpha_a(r[x]) \geq 0$)) Then
        If (($currIS$ != $g$S.last())) Then
            $currIS \leftarrow g$S.next() ;
        Else
            $found \leftarrow 1$;
        End If
    End If
    $r \leftarrow \mathcal{R}$.next() ;
End While
return $found$
```

The MINIFUZZY count is based on the function $FindMiniSeq$, Algorithm 3, which searches for a candidate sequence in the record set of an object. It consists of searching for each itemset of the sequence, following their order. When the first one is found within a record with a degree greater than the threshold of relevancy $\omega$, the following itemset is searched for in the following record and so on.

```
ALGORITHM 3 - FindMiniSeq
Input: $gS$, candidate $g$-$k$-sequence; $\mathcal{R}$, record set to run
Ouput: $found$, 1 if $g$S is found, 0 else

$currIS \leftarrow g$S.first() ;
$found \leftarrow 0$ ;
$r \leftarrow \mathcal{R}$.first() ;
While ((($\underline{!}found$) || ($r$!=$\varnothing$))) do
    If (($\top_{[x,a] \in currIS} \alpha_a(r[x]) \geq \omega$)) Then
        If (($currIS$ != $g$S.last())) Then
            $currIS \leftarrow g$S.next() ;
        Else
            $found \leftarrow 1$;
        End If
    End If
    $r \leftarrow \mathcal{R}$.next() ;
End While
return $found$
```

In Table V, transactions that validate the sequence $Sx$ with MINIFUZZY count are d2 followed by d3. This corresponds to the boldfaced items.

## *B. TotallyFuzzy Frequency Computation*

TOTALLYFUZZY implements the fuzzy frequency computation using a thresholded $\Sigma$-count to calculate the number of objects supporting a sequence. In this section, we present the frequency calculation carried out by this algorithm.

*1) A Trade-Off between Space and Computational Complexity:*
When considering classical sequential patterns or even in a certain way SPEEDYFUZZY or MINIFUZZY, an object supports a sequence or not. So the scan can stop as soon as the sequence is found within the record set of an object. On the contrary, as TOTALLYFUZZY computes a thresholded $\Sigma$-count, for each object and each sequence, the best membership degree must be considered. This degree is computed as the aggregation of the itemset frequencies. The order of the fuzzy items must also be taken into account. This leads to an exhaustive scanning of the record set, as performed for association rule mining.
Let us consider, for instance, Table V. If we look for the same sequence as above, i.e. $Sx$, the first occurrence of this sequence is the one underlined, supported by transaction d1 then d3. However, the best occurrence of this sequence is given by transaction d4 followed by d6, underlined twice.

A naive approach could be that for each candidate $k$-sequence (likely frequent sequence) we scan all the database to find its frequency, which would involve $n^k$ scans of the database at most if $n$ is the number frequent items. The only structure kept in memory would thus be the list of candidate sequences. However, this would be very inefficient, as the computational time would explode.
To reduce this number of scans, we have defined a data structure enabling us to find all representations of all $k$-sequences in only $k$ scans of the database. The computational time is also lower but the used memory space is increased. However, some optimizations have been implemented to bound this spatial complexity.
So we present here an efficient implementation based on the notion of *path*. One path corresponds to a possible instantiation of the candidate sequence itemsets within the object record set. Several paths may be initialized for one object. For the global frequency computation, we only keep the complete one having the best degree.

*Definition 4:* One *path* is a triplet containing the already found sequence $seq$, the currently searched itemset $curIS$ (coming next in the candidate sequence) and the current membership degree $curDeg$.

The next subsections illustrate how TOTALLYFUZZY count works and detail the functions for the support calculation.

*2) Computing a sequence support. an Illustration:*
As an illustration, we run the TOTALLYFUZZY count function, $FindTotallySeq$, to compute the frequency of the candidate sequence $Sx$ <([candy, little])([soda, lot])> for customer 1 from Table IV, with a threshold $\omega$= 0.2. This is summarized in Table VI.

TABLE VI
ILLUSTRATION OF TOTALLYFUZZY ON C3

| | |
|---|---|
| | $pth1 : (\varnothing, ([candy, little]), 0)$ |
| **After d1** | $pth1 : (< ([candy, little]) >, ([soda, lot]), 0.25)$ |
| **After d2** | $pth1 : (< ([candy, little]) >, ([soda, lot]), 0.25)$ |
| | $pth2 : (< ([candy, little]) >, ([soda, lot]), 0.5)$ |
| **Opt.** | $pth1$ is deleted |
| **After d3** | $pth2 : (< ([candy, little]) ([soda, lot]) >, \varnothing, 0.5)$, closed |
| | $pth3 : (< ([candy, little]) >, ([soda, lot]), 0.5)$ |
| **After d4** | $pth2 : (< ([candy, little]) ([soda, lot]) >, \varnothing, 0.5)$, closed |
| | $pth3 : (< ([candy, little]) ([soda, lot]) >, \varnothing, 0.5)$, closed |
| | $pth4 : (< ([candy, little]) >, ([soda, lot]), 0.5)$ |
| | $pth5 : (< ([candy, little]) >, ([soda, lot]), 0.75)$ |
| **Opt.** | $pth3$ and $pth4$ are deleted |
| **After d5** | $pth2 : (< ([candy, little]) ([soda, lot]) >, \varnothing, 0.5)$, closed |
| | $pth5 : (< ([candy, little]) > ([soda, lot]) >, \varnothing, 0.63)$, closed |
| | $pth6 : (< ([candy, little]) >, ([soda, lot]), 0.75)$ |
| **Opt.** | $pth2$ is deleted |
| **After d6** | $pth5 : (< ([candy, little]) > ([soda, lot]) >, \varnothing, 0.63)$, closed |
| | $pth6 : (< ([candy, little]) > ([soda, lot]) >, \varnothing, 0.87)$, closed |
| **Opt.** | $pth5$ is deleted |
| **C3 deg.** | **0.87** |

First, the process is initialized by creating one empty path $pth1 = (\varnothing, ([candy, little]), 0)$ (Table VI, row 1). Then it begins by scanning the first record of customer 1, d1. The currently searched itemset $curIS$ is found with a degree *d1[candy, little]=0.25* $\geq \omega$. The path $pth1$ is so updated with $pth1.seq \leftarrow < ([candy, little]) >$, $pth1.curIS \leftarrow ([soda, lot])$ and $pth1.seq \leftarrow 0.25$ (Table VI, row 2).
Then transaction d2 contains the first sequence of the candidate sequence $g$-$S$, $([candy, little])$. So a new path is created, $pth2 \leftarrow (< ([candy, little]) >, ([soda, lot]), 0.5)$. As this transaction does not contain the next itemset for $pth1$, the scan of the dataset should continue. Before that, an optimization is carried out to avoid having to use to much memory space: for two paths at the same step, only the one with the best $curDeg$ value is kept. Here we have $pth1.curDeg < pth2.curDeg$ so $pth1$ is deleted and the scan of the next transaction continues with $pth2$ (Table VI, row 3).
Transaction d3 is then checked. It contains $pth2.curIS = ([soda, lot])$. It is thus modified to $pth2 \leftarrow (< ([candy, little]) \ ([soda, lot]) >, \varnothing, 0.5)$. This path is closed since it contains all itemsets of $g$-$S$. However, we plan to improve this solution. So, before having modified $pth2$, it is copied into $pth3$. At this step, we have two paths: $pth2$, closed with a frequency degree of 0.5 and $pth3 = (< ([candy, little]) >, ([soda, lot]), 0.5)$ (Table VI, row 4).
Transaction d4 is then checked. $pth3.curIS$ is found so it is copied, modified and then closed with a degree of 0.5. As d4 also contains the first itemset of $g$-$S$, a new path is created. At this point, we have four paths (Table VI, row 5). The $Optimize$ function deletes, for each step in the sequence, the path with the lowest degree, i.e. $pth3$ and $pth4$. Scanning then continues. At d5, $pth5$ is copied into $pth6$, modified and closed with a frequency degree of 0.63, so $pth2$ is deleted. Finally, at d6, $pth6$ is updated and closed with a degree of 0.87. This is the path kept for customer 3.

*3) Computing a sequence support. the Algorithms:*
The algorithm TOTALLYFUZZY uses the function *FindTotallySeq* to carry out an ordered scanning in one object record set

and finds the best occurrence of a sequence in the records of one object.

**ALGORITHM 4** - FindTotallySeq

```
Input: $g$-$S$, candidate $g$-$k$-sequence;
       $R$, record set to run
Ouput: $m$, frequency degree of the best $g$-S representation
       instanciated in the record set $R$

$Paths$ : list of paths $\rightarrow$ (seq, curIS, curDeg)
$Paths$ $\leftarrow$Path($\varnothing$, $g$S.first, 0)
For each transaction $r \in R$ do
    For each path $pth \in Paths$, not updated at $r$ do
        If ($pth$ not closed) Then
            If ($pth$.curIS $\in r$) Then
                $pth$.curDeg $\leftarrow$ $pth$.curDeg $\mid \overline{\top}_{[x,a] \in pth.\text{curIS}} \alpha_a(r[x])$ ;
                Update($pth$);
            End If
        End If
        For j from 2 to $pth$.curIS -1 do
            [Search for an improvement of the current path]
            If (($g$S.get(j) $\in r$) &
            ($\overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(r[x]) > pth$.curDeg[j])) Then
                nCurIS $\leftarrow$ $g$S.get(j) ;
                For i from 1 to j-1 do
                    nSeq $\leftarrow$ nSeq $\mid$ $g$S.get(i) ;
                    nCurDeg $\leftarrow$ nCurDeg $\mid$ $pth$.curDeg[i];
                End For
                nCurDeg $\leftarrow$ nCurDeg $\mid \overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(r[x])$ ;
                $Paths \leftarrow Paths \cup$ Update((nSeq, nCurIS, nCurDeg)) ;
            End If
        End For
    End For
    If (($g$S.first $\in r$) & (not(FirstPass))) Then
        $pth$ $\leftarrow$Path($\varnothing$, $g$S.first, $\overline{\top}_{[x,a] \in gS.first} \alpha_a(r[x])$);
        $Paths \leftarrow Paths \cup pth$ ;
        Update($pth$) ;
    End If
    Paths.Optimize() ; [deletion of less pertinent paths]
End For
For each path $pth \in Paths$ do
    If ($pth$ not closed) Then
        delete($pth$); [deletion of paths not containing the whole sequence]
    End If
End For
$pth \leftarrow Paths.first$ ; [$Paths$ only contains the best complete path]
$m \leftarrow \odot(pth$.curDeg) ; [Aggregation to return the frequency degree]
return $m$;
```

When the first itemset of the sequence is found, one path is created with the itemset frequency. The next records are checked to find either the following part of the sequence or once again the beginning of the sequence or an improvement of the paths already created. All possible paths are thus completed step-by-step at each record. The frequency degree of the best path for the whole sequence is then aggregated and returned.

The $Update$ function allows an update of each path and closes the complete ones. The $Optimize$ algorithm enables deletion of unnecessary paths, which means, for two paths at the same point, to only keep the one with the greatest degree.

**ALGORITHM 5** - Update

```
Input: $pth$, path to update

$pth$.seq $\leftarrow$ $pth$.seq $\cup pth$.curIS ;
If ($pth$.curIS.next $\neq \varnothing$) Then
    $pth$.curIS $\leftarrow$ $pth$.curIS.next ;
Else
    close($pth$);
End If
```

**ALGORITHM 6** - Optimize

```
Input: $Paths$, list of paths to reduce

For each $pth \in Paths$ do
    While (($Paths$.hasnext())) do
        $pth' \leftarrow Paths$.next() ;
        If ($pth'.currIS = pth.currIS$) Then
            If ($\odot(pth.curDeg) \geq \odot(pth'.curDeg)$) Then
                delete($pth'$);
            Else
                delete($pth$);
            End If
        End If
    End While
End For
```

## V. MINING FREQUENT SEQUENCES

This section presents the overall algorithm, called SMT-Fuzzy which extracts the fuzzy sequential patterns. Our approach extends the level-wise approach generate-prune within the sequential pattern context and, more particularly, uses the prefix-tree structure described in [15], in order to improve the frequency computation process. Other data structures may be considered such as the one described in [17]. This structure is used to store both the candidate and frequent sequences. The tree in Fig. 2 represents the sequences *<([candy, little][soda, lot])([soda, lot])>*, *<([candy, little])([toothpaste, medium])>* and also the subsequences *<([toothpaste, medium])>* and *<([soda, lot])([soda, lot])>*.

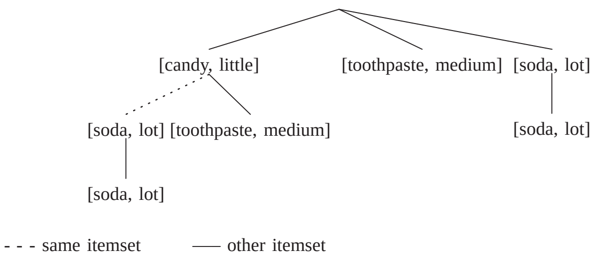

Fig. 2. Storage of sequences as a prefix-tree

The overall algorithm SMT-Fuzzy is presented by Algorithm 7. Whatever function is used to compute the frequency, the way of working is the same. First, the fuzzy frequency of all fuzzy items is computed and only items with a frequency greater than $minFreq$ are stored as frequent ones of size 1. Then, while a candidate generation still is possible, the $generate$ function builds the candidate sequences of size $k$ from the frequent sequences of size $k$-$1$. Then a scan of the database is run and frequencies are computed using $CalcFuzzyFreq$. Finally, all infrequent sequences of size $k$

are pruned and the process continues. At the end of the process, we obtain a Prefix-tree containing all frequent sequences of the database with their frequency at each leaf of the tree.

ALGORITHM **7** - SMT-Fuzzy

```
Input: $DB$, a database, $minSupp$
Ouput: $PT$, the prefix-tree of frequent sequences

find-1-Frequent() ;
While (#candidate $>$ 0) do
    $generate(PT$.depth+1) ;
    For each sequence $s$ in $PT$ do
        $CalcFuzzyFreq(s)$;
    End For
    $prune(PT)$;
End While
return ($PT$) ;
```

As for PSP, the overall computational complexity of SMT-Fuzzy depends only on the length of the candidate sequences, and thus of the frequency calculation. Computation of the $generate$ and $prune$ functions are indeed insignificant in comparison with that of $CalcFuzzyFreq$.

According to the fuzzification level expected by the end-user, the function $CalcFuzzFreq$ will call:

- $FindSpeedySeq$; SMT-Fuzzy will actually be run as SPEEDYFUZZY. It will return the fuzzy sequential patterns corresponding to a global view of the database, giving the frequency of attribute quantities for each one.
- $FindMiniSeq$; SMT-Fuzzy will be run as MINIFUZZY. It will return more accurate information, giving the fuzzy sequential patterns corresponding to a minimum relevancy of attribute quantities.
- $FindTotallySeq$; SMT-Fuzzy will be run as TOTALLYFUZZY. It will return accurate fuzzy sequential patterns, taking into account both the relevancy and frequency of fuzzy attributes within the dataset.

In the next paragraph, these different versions of SMT-Fuzzy, respectively called SPEEDYFUZZY, MINIFUZZY and TOTALLYFUZZY, are used to mine fuzzy sequential patterns within the database given in Table II.

First, the algorithm scans for frequent fuzzy items. Table VII gives the frequencies obtained through the different counts using $\omega = 0.49$ for MINIFUZZY and TOTALLYFUZZY. We obtain the 1-frequent ones boldfaced in this figure, for a minimum frequency of 52%.

TABLE VII
FUZZY FREQUENCIES $FFreq$ (%) FOR EACH FUZZY ITEM

| Count | candy | | toothpaste | | | soda | | ball | | videogame | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | li. | lot | li. | med. | lot | li. | lot | few | lot | few | med. | lot |
| $F_{TF}$ | **68.75** | **56.5** | 25 | **62.5** | 35 | 25 | **100** | 25 | 50 | 0 | **62.5** | 35 |
| $F_{MF}$ | **100** | **75** | 25 | **75** | 50 | 50 | **100** | 25 | 50 | 0 | **75** | 50 |
| $F_{SF}$ | **100** | **100** | 25 | **75** | 50 | 50 | **100** | 25 | 50 | 0 | **75** | 50 |

The different algorithms have found the same frequent items. Then these fuzzy items are combined into candidate 2-sequences whose the frequency is calculated. Frequent 2-sequences are then combined into candidate 3-sequences whose the frequency is calculated and so on.

Table VIII shows the sequential patterns (maximal frequent sequences) respectively extracted by SPEEDYFUZZY, MINIFUZZY and TOTALLYFUZZY, for a minimum frequency $minFreq$=52% and $\omega$=0.49.

TABLE VIII
EXTRACTED SEQUENTIAL PATTERNS

| | | |
|---|---|---|
| **Sequential patterns with** SPEEDYFUZZY | <([video game, medium])> | 75% |
| | <([candy, little])([toothpaste, medium])> | 75% |
| | <([candy, lot])([toothpaste, medium])> | 75% |
| | <([candy, lot][soda, lot])([soda, lot])([soda, lot])> | 75% |
| | <([candy, little][soda, lot])([soda, lot])([soda, lot])> | 75% |
| **Sequential patterns with** MINIFUZZY | <([video game, medium])> | 75% |
| | <([candy, lot])> | 75% |
| | <([candy, little])([toothpaste, medium])> | 75% |
| | <([candy, little][soda, lot])([soda, lot])([soda, lot])> | 75% |
| **Sequential patterns with** TOTALLYFUZZY | <([video game, medium])> | 62.5% |
| | <([candy, lot])> | 56.5% |
| | <([candy, little])([toothpaste, medium])> | 53.5% |
| | <([candy, little][soda, lot])([soda, lot])([soda, lot])> | 57% |

Note that the frequent items are the same for all counting methods. The difference is in the number and length of the sequences. For a same $minFreq$, the number and length of the mined patterns are indeed greater with MINIFUZZY or SPEEDYFUZZY than with TOTALLYFUZZY. This is due to the thresholded $\Sigma$-count.

This reduction in the number of patterns can be used for a database containing very high amounts of frequent patterns to find the most relevant ones. The advantage of this method is to be more selective and therefore to find the most relevant sequential patterns. The user will thus be provided with a selection of patterns, and not only have to assess a selection of patterns or a really large quantity of them.

## VI. EXPERIMENTS

In this section, we present a comparison of performances of the three algorithms SPEEDYFUZZY, MINIFUZZY and TOTALLYFUZZY and the algorithm PSP [15], a generate-prune approach for crisp sequential pattern mining. We also compare TOTALLYFUZZY with an implementation of the [10]'approach. These experiments were carried out on a PC - Linux 2.6.7 OS, CPU 3 GHz with 512 MB of memory. All the algorithms were implemented in Java on the PSP principle. In particular, the Prefix-Tree structure was used to store the candidate and frequent sequences.

### A. Synthetic Datasets

We used synthetic datasets which have been created in several steps. In a first step, quantitative databases were generated using an enhancement of DatGen [18]. Then an automatic partitioning was carried out using a tool based on a module of Weka [19] using the object-timestamp-item-quantity file. This

*DiscretizeFilter* module enables us to divide a given interval into several smaller intervals of an equivalent number of elements (equi-depth). Lastly, the membership degree database was created.

TABLE IX
PARAMETERS FOR SYNTHETICAL DATABASES GENERATION

| | |
|---|---|
| R | Number of records in the database |
| O | Number of objects in the database |
| I | Average number of items by transaction |
| X | Number of possible items in the database |
| Q | Maximum quantity value |
| P | Number of fuzzy sets by item |

Table IX gives the list of parameters used for the data generation method and Table X shows the databases used and their properties. For these experiments, the number of fuzzy sets by attribute was set at 3, the maximal quantity by item at 10 and the number of different items in the database at 100. These databases are rather dense.

TABLE X
PARAMETER VALUES FOR SYNTHETIC DATASETS IN EXPERIMENTS

| Name | O | R | I |
|---|---|---|---|
| ob5rec100I10 | 5,000 | 100,000 | 10 |
| ob5rec200I10 | 5,000 | 200,000 | 10 |
| ob5rec250I10 | 5,000 | 250,000 | 10 |
| ob5rec350I10 | 5,000 | 350,000 | 10 |
| ob5rec450I10 | 5,000 | 450,000 | 10 |
| ob5rec500I10 | 5,000 | 500,000 | 10 |
| ob10rec250I10 | 10,000 | 250,000 | 10 |
| ob10rec500I10 | 10,000 | 500,000 | 10 |

## B. Operators

We have chosen to implement our version of TOTALLYFUZZY with *min* as a t-norm operator and *max* as a t-conorm operator. Let us consider an item. To be considered frequent, this item must be relevant enough for enough objects (regarding the value of $minFreq$). This means that its degree must be greater than $\omega$. Now let $IS$ be an itemset. If we consider that each item of $IS$ is relevant, this itemset could be considered as relevant. The frequency of this itemset should thus not be lower than the minimum frequency of the items. The operator for computing this frequency must thus be idempotent. For this reason, our t-norm operator is $min$.

The aggregation operator we have chosen is the average operator. Each itemset in the sequence will thus be considered as equivalent to the others. However, we could have used the min operator to compute the frequency of sequences. A sequence is indeed the intersection, taking into account the temporal order of itemsets. This can be computed using $min$ as explained above. The results described in Fig. 3 show that the aggregation by $min$ is too selective and that it prunes too many sequential patterns.

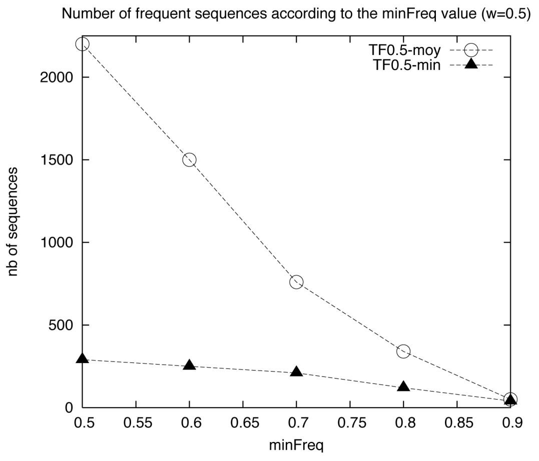

Fig. 3. Number of frequent sequences according to $minFreq$ depending on the aggregation operator for ob5rec100I10

## C. Fuzzy Algorithms vs. PSP Algorithm

The aim of these experiments is to compare the fuzzy algorithms with PSP [15], a sequential pattern algorithm working with the same efficient structure as our fuzzy algorithms. Fig. 4 shows the mining time according to $minFreq$. Note that the extraction time increases as $minFreq$ decreases. The number of frequent patterns and generated candidates is indeed higher for a low $minFreq$. So the algorithms scan the dataset more times.

It can also be noted that SPEEDYFUZZY is as fast as PSP despite the fact that it scans three times more items. One should indeed take into account that each PSP item has been split into three (it could have been more) fuzzy items to meet our algorithm needs. TOTALLYFUZZY and MINIFUZZY run slightly slower.

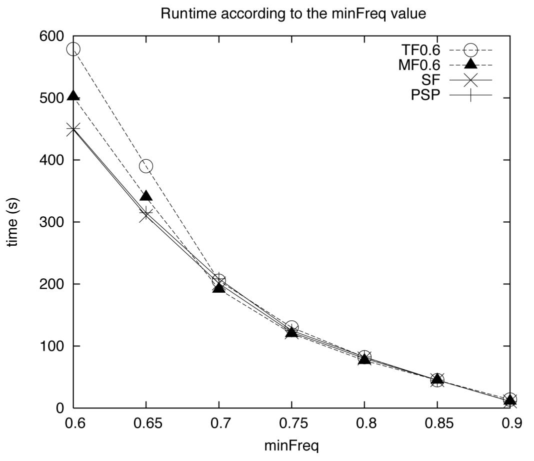

Fig. 4. Runtime according to $minFreq$ for ob5rec100I10

If we compare the number of frequent sequences for the same minimum frequency, Fig. 5 shows that MINIFUZZY and TOTALLYFUZZY definitely extract less frequent sequences than SPEEDYFUZZY and PSP. This is due to the frequency evaluation. In fact, these two fuzzy algorithms, MINIFUZZY and TOTALLYFUZZY, only keep items which have a degree greater than $\omega$ and so which are considered relevant by the user. The number of frequent sequences is then necessarily reduced compared to SPEEDYFUZZY or PSP.

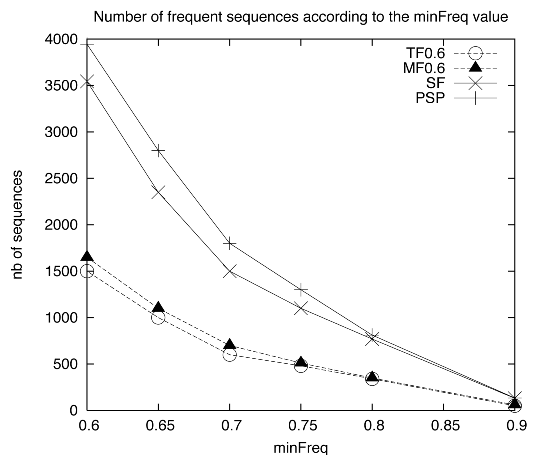

Fig. 5. Number of frequent sequences according to $minFreq$ for ob5rec100I10

Finally, Fig. 6 shows the extraction time according to the number of records in a database describing records for 5000 objects, for $minFreq = 0.6$. Note that the fuzzy algorithms have the same behavior as PSP.

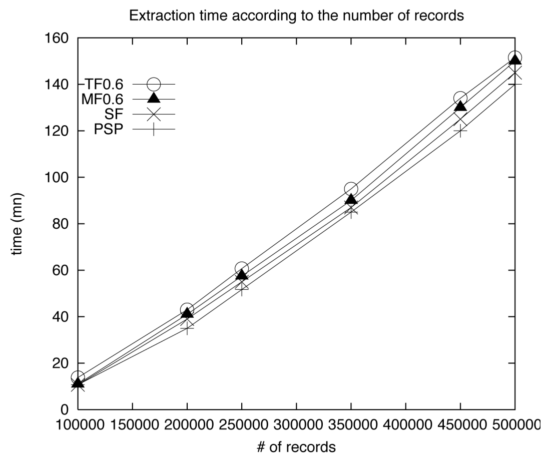

Fig. 6. Runtime according to the number of records in the datasets, $minFreq$=0.6

## D. TOTALLYFUZZY*vs. Hong and al. approach*

The aim of this paragraph is to show the benefits of our method in comparison to the method of Hong et al. [10] presented in Section II-B. In particular, we study the loss of information resulting from their approach.

In our experiments, we compared our algorithm TOTALLYFUZZY (the closest to their approach) with an implementation of the approach of Hong et al.

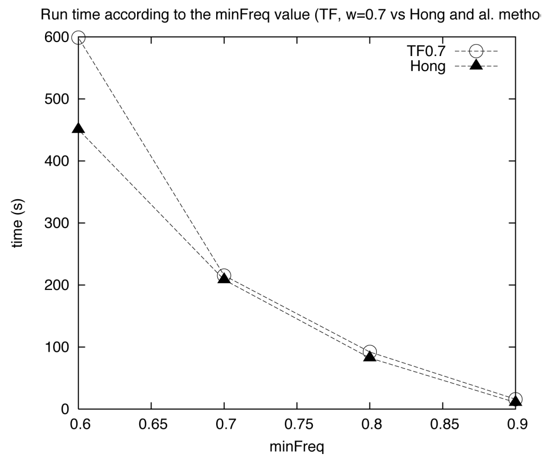

Fig. 7. Runtime according to $minFreq$ for TOTALLYFUZZY ($\omega$ = 0.7) and [10] for ob5rec100I10

As shown in Fig. 7, the selection of the item having the highest cardinality carried out by [10] efficiently decreases the runtime of the fuzzy sequential pattern mining. This algorithm runs faster than TOTALLYFUZZY. However, it can be noted from Table XI that some items (e.g. *[1000, 3]* and *[990, 3]*) with a very relevant frequency (resp. *88.44%* and *91.46%*) are found by TOTALLYFUZZY and not by the method of Hong et al., they are in fact deleted during the preprocessing step.

TABLE XI
NUMBER OF FREQUENT ITEMS EXTRACTED BY TOTALLYFUZZY ($\omega$=0.7) AND HONG AND AL. METHOD, FOR $minFreq = 60\%$ ON OB5REC100I10 DATABASE.

| Frequent fuzzy items | TOTALLYFUZZY($\omega$=0.7) | Hong and al. |
|---|---|---|
| [960, 1] | 65.12% | 65.12% |
| [970, 1] | 74.32% | 74.32% |
| [980, 1] | 85.36% | 85.36% |
| [**980, 3**] | **73.18%** | **not found** |
| [990, 1] | 97.24% | 97.24% |
| [**990, 2**] | **70.28%** | **not found** |
| [**990, 3**] | **91.46%** | **not found** |
| [1000, 1] | 95.94% | 95.94% |
| [**1000, 2**] | **66.76%** | **not found** |
| [**1000, 3**] | **88.44%** | **not found** |
| Nb of frequent items | 10 | 5 |

Consequently, the number of maximal frequent sequences found by TOTALLYFUZZY is higher than that obtained by the method of Hong et al. method. The frequent sequences resulting from the five ignored frequent items are indeed never generated and thus cannot be found. Fig. 8 shows this loss of information. In fact, for $minFreq$=70%, the approach of Hong et al. only finds two thirds of the sequential patterns mined by TOTALLYFUZZY.

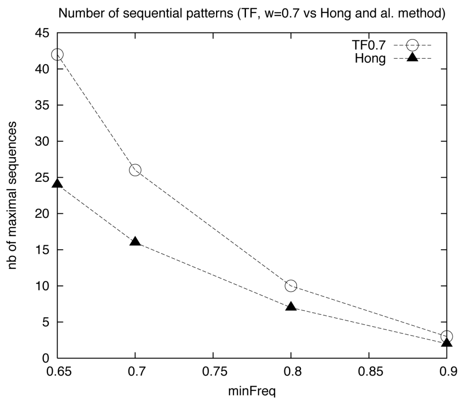

Fig. 8. Number of sequential patterns extracted by TOTALLYFUZZY ($\omega$=0.7) and by [10] for ob5rec100I10

## E. Response to $\omega$ Variation

Parameter $\omega$ introduced for the algorithms MINIFUZZY and TOTALLYFUZZY regulates the counting. It is indeed used to filter the relevancy of each record according to the membership degree of the fuzzy attributes. A high value for this parameter enables the user to extract sequential patterns which are both supported by many objects and well-represented in each ones' sequence. On the contrary, a low value for parameter $\omega$ allows extraction of sequential patterns supported by more objects. But these patterns do not provide information on the extent to which the objects support them.

The advantage of using this parameter combined with the minimum frequency is that it gives two information elements in one. Choosing a parameter $\omega$ greater than $minFreq$ will indeed enable the user to more precisely select frequent sequences within the database, keeping only the ones that are both frequent and relevant.

In order to show the response to the variation of the $\omega$ parameter, we built a new synthetic dataset (ob5rec100I10Q100) using a larger set for quantities (between 1 and 100) and we built fuzzy sets, the same for each possible quantity, enabling more precise variation for membership degrees, as shown in Fig. 9.

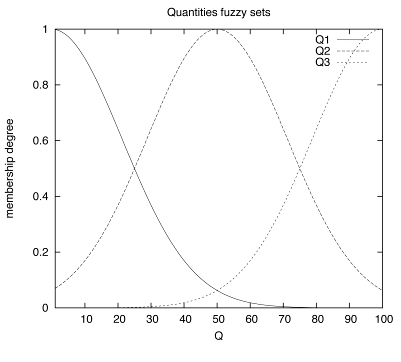

Fig. 9. Progressive fuzzy sets for quantities between 1 and 100 in ob5rec100I10Q100

We then ran experiments varying the value of parameter $\omega$ to show its influence on the runtime and found sequences. As we can see in Fig. 10 we extract more sequences as $\omega$ decreased. We indeed enable the mining algorithm to keep less relevant sequences, with lower membership degree as supporting sequences, so we extract more frequent sequences. Consequently the runtimes also increase as $\omega$ decreases, as shown in Fig. 11. The programme indeed has to scan the database for more candidate sequences.

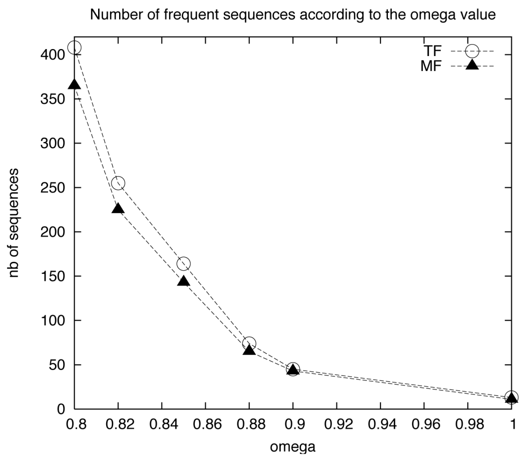

Fig. 10. Number of sequential patterns extracted by TOTALLYFUZZY and MINIFUZZY for ob5rec100I10Q100

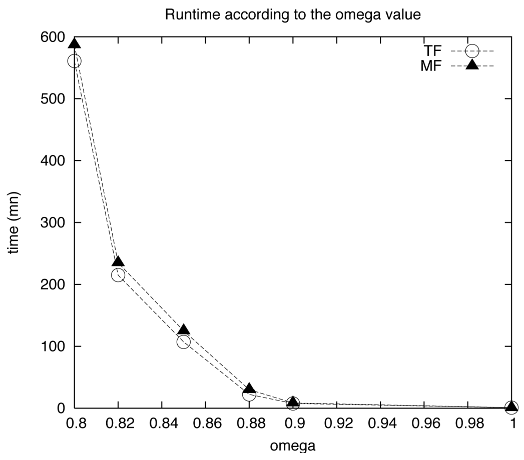

Fig. 11. Runtime of TOTALLYFUZZY and MINIFUZZY for ob5rec100I10

## F. Experiments on Real-World Datasets

Several experiments have been carried out on real-world datasets. The first ones were aimed at web access log mining. The purpose of the experiments was to increase knowledge obtained through crisp sequential pattern mining for the number of accesses to the same page during a session. Other experiments have been carried out on registered trademarks in order to mine for knowledge about the building of trademarks.

### 1) *Web Access Log Mining:*

The aim of these experiments is to show that soft sequential pattern mining brings more relevant information compared to crisp sequential pattern mining. In our case, access logs from a laboratory website were prepared and mined to find frequently visited pages – such as in crisp sequential pattern mining – but also repeatedly visited pages. These experiments are developed in [20].

The access logs were pre-processed and the dataset recorded the number of accesses to one page, the same half-day by one user. It contained 27209 web pages visited by 79756 different IPs over 16 days (32 half-days). As explained previously, quantities are converted into fuzzy set membership degrees and mined with fuzzy algorithms. These are partitioned into 3 fuzzy sets per URL using the same tool as that used for synthetical datasets.

We ran the access log database with the different algorithms, starting with SPEEDYFUZZY, to first extract overall information concerning which URLs were often accessed several times in the same session. Then we mined for more accurate knowledge using TOTALLYFUZZY. These experiments revealed that the behavior of our algorithms on synthetical datasets was confirmed on real-world datasets.

Fig. 12 shows the runtime time according to $minFreq$. Note that the extraction time increases as $minFreq$ decreases.

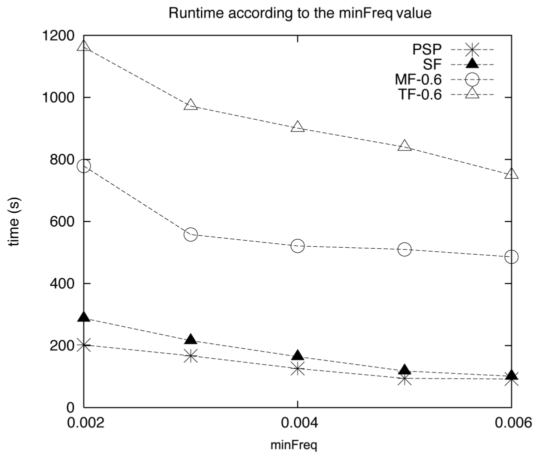

Fig. 12. Runtime according to $minFreq$ for 79756 sequences

If we compare the number of frequent sequences for the same minimum support, Fig. 13 shows that MINIFUZZY and TOTALLYFUZZY extract less frequent sequences than SPEEDY-FUZZY and PSP due to the support definition. In fact, these two fuzzy algorithms, MINIFUZZY and TOTALLYFUZZY, only keep items which have a degree greater than $\omega$ and thus which are considered as relevant by the user. The number of frequent sequences is then necessarily reduced compared to SPEEDYFUZZY or PSP.

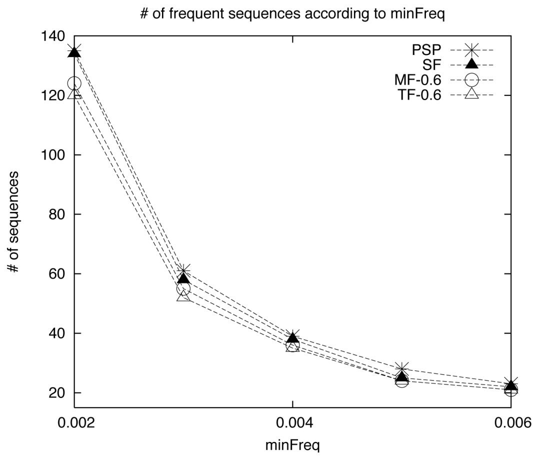

Fig. 13. Number of frequent sequences according to $minFreq$ for 79756 sequences

Figure 14 shows the extraction time according to the number of records in a database for $minFreq = 0.2\%$, and the fuzzy algorithms have the same behavior as PSP.

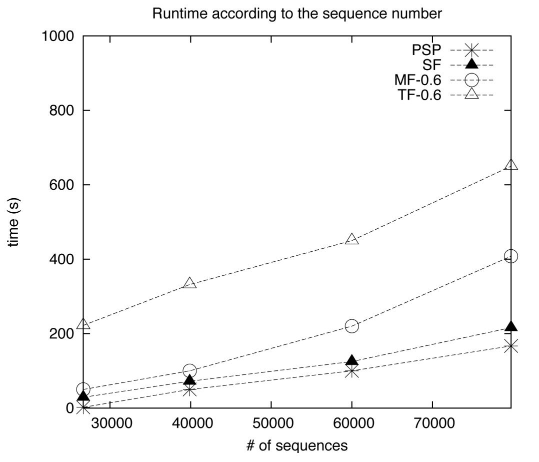

Fig. 14. Runtime according to the number of sequences in the datasets, for $minSupp$=0.2%

These behaviors confirm the good results obtained in experiments on synthetic datasets described in previous Section VI-C.

Furthermore, we obtained the same frequent URLs using crisp or soft sequential pattern mining. The advantages of our method concern the additional knowledge supplied by quantities. Indeed, while the crisp algorithm PSP found that one URL (e.g. 139) was frequently accessed, TOTALLYFUZZY found that it was frequently accessed between 2 and 5 times during the same period. Our patterns also revealed that the root page was only a step in the browsing sequences and often accessed only once during the same session.

### 2) A Trademark Database Summarization:

INPI is the French *National Institute for the Industrial Property*. It manages and registers industrial title deeds, communicates information on industrial property and devises and adapts French industrial property rights.
The INPI trademark database registered around 2 million names between 1961 and 2004. These trademarks have been registered in one or several categories corresponding to industrial fields such as “toys, games” or “clothing, shoes, hat industry”. Each record is composed of several attributes, for instance: idnumber, trademark, registration date, registering company, registering class. The global goal of analysing such a database is, for the linguistic expert, to understand how trademarks act on consumers.

These experiments aim at discovering word form in registered tradenames. These experiments were run in several steps. First some statistical results were used to select trademarks that would be interesting to investigate. Then we used TOTALLYFUZZY to sum up this database.

A statistical analysis revealed that, compared to other registration fields, an appreciable amount of trademarks registered in telecommunication fields contain some figures and symbols. As these results were atypical, we decided to search for an overall description of the composition of trademarks registered in this field. This dataset contains more than 480,000 trademarks consisting of one or several words. The mined patterns would, for example, be “Almost one third of registered names are first composed of one part containing a lot of letters then of a part with a few figures and then a part with a few punctuation marks and a few letters”. These proposals would be built from a sequence *<([letter, lot])([figure, few])([punct_symb, few],[letter, few])>*.
In this case, we can consider trademarks consisting of one or several words. In fact, a name consisting of only one word will appear as an itemset instead of a sequence for a name consisting of several words. In order to mine such patterns, the INPI database is converted into the format [ID_TM,#WORD, FUZZY ITEM,DEGREE], as shown Table XII, whereas the fuzzy items are described in Fig. 15. These fuzzy sets are given by the linguistic expert and are built on common word composition.

TABLE XII
DATABASE TRANSLATION FOR INTRA-NAME PATTERN MINING

| **Mining formalism** | | **trademark database** |
|---|---|---|
| object | ↔ | a trademark |
| timest amp | ↔ | number of the word in the trademark |
| fuzzy item | ↔ | # of signs, letters, figures, punctuations or symbols in this word |

Extended results are presented in [21], but some revealing patterns show that “Almost one half of these trademarks contain three words with a lot of signs, the first and last ones with a lot of signs and letters, the second one with few letters or a medium amount of signs” (<([#sign, lot][#letter, lot]) ([#letter, few]) ([#sign, lot][#letter, lot])> (42%), <([#sign, lot][#letter, lot]) ([#sign, medium]) ([#sign, lot][#letter, lot])> (42%)).

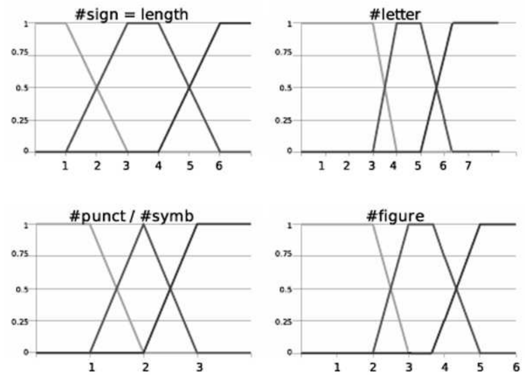

Fig. 15. Fuzzy sets for fuzzy items for intra-name pattern mining

By decreasing the value of $minFreq$ and $\omega$, we did not find patterns with figures or symbols using TOTALLYFUZZY. As these results did not confirm the statistical analysis results, we ran SPEEDYFUZZY and MINIFUZZY, which then confirmed these statistical results.

Our conclusion on these experiments is that the fuzzy partitioning intuitively given by the expert from common language does not mesh well with the trademark composition. Further experiments will be carried out with automatically built fuzzy sets from the database using a clustering algorithm as presented in [2] and [22].

## VII. DISCUSSION AND PROSPECTS

In this paper, we present a complete and efficient fuzzy approach for sequential pattern mining which enables us to process historical numerical data such as demographic or sensor data. Extracting sequential patterns in such databases is highly interesting for event sequence detection. However, existing algorithms either do not allow numerical data processing or only extract one information element from the several which could be obtained from quantitative databases.

Our proposal clearly and completely defines different concepts and principles associated with fuzzy sequential patterns, which are presented through a global framework for fuzzy sequential pattern mining. This framework enables the user to choose between three fuzzyfication levels and thus between three information levels: presence and frequency of certain quantitative attributes or sequences, the frequency of these sequences with minimum relevancy or weighted frequency according to the relevancy of the quantitative sequence. Note that this framework also generalizes the crisp sequential pattern definition and that it allows mining of binary data.

The algorithms we present, i.e. SPEEDYFUZZY, MINIFUZZY and TOTALLYFUZZY, have been implemented and tested, thus highlighting the benefits of our novel approach and the feasibility with respect to the fuzzy methods described. The user can thus handle three fuzzification levels thanks to our three different algorithms. This choice allows extraction of frequent sequences through a trade-off between relevancy and performance. Experiments have highlighted that this work could be applied to different kinds of data and there are many prospects.

Several extensions could indeed be investigated. More particularly, we plan to apply this framework to the various proposals of incremental fuzzy sequential pattern mining presented in [23]. We also plan to study the extension of our framework to take temporal constraints proposed in [9] into account to mine fuzzy generalized sequential patterns.

## REFERENCES

[1] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, 1996, pp. 1–12.

[2] A. Fu, M. Wong, S. Sze, W. Wong, and W. Yu, "Finding fuzzy sets for the mining of fuzzy association rules for numerical attributes," in *Proc of the 1st Int. Symp. on Intelligent Data Engineering and Learning*, 1998, pp. 263–268.

[3] C. M. Kuok, A. W.-C. Fu, and M. H. Wong, "Mining fuzzy association rules in databases," *SIGMOD Record*, vol. 27, no. 1, pp. 41–46, 1998.

[4] T.-P. Hong, K.-Y. Lin, and S.-L. Wang, "Fuzzy data mining for interesting generalized association rules," *Fuzzy Sets and Systems*, vol. 138, pp. 255–269, 2003.

[5] W.-H. Au and K. Chan, "An effective algorithm for discovering fuzzy rules in relational databases," in *Proc. of the IEEE World Congress on Computational Intelligence*, 1998, pp. 1314–1319.

[6] G. Chen, Q. Wei, and E. Kerre, "Fuzzy data mining: Discovery of fuzzy generalized association rules," *Recent Research Issues on Management of Fuzziness in Databases*, 2000.

[7] M. Delgado, D. Sanchez, and M.-A. Vila, "Acquisition of fuzzy association rules from medical data," *Fuzzy Logic in Medicine*, vol. Studies in Fuzziness and Soft Computing Series (83), pp. 286–310, 2002.

[8] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. of the 11th IEEE Int. Conf. on Data Engineering*, 1995, pp. 3–14.

[9] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proc. of 5th Int. Conf. on Extending Database Technology*, 1996, pp. 3–17.

[10] T.-P. Hong, K.-Y. Lin, and S.-L. Wang, "Mining fuzzy sequential patterns from multiple-items transactions," in *Proc. of the Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, 2001, pp. 1317–1321.

[11] R.-S. Chen, G.-H. Tzeng, C. Chen, and Y.-C. Hu, "Discovery of fuzzy sequential patterns for fuzzy partitions in quantitative attributes," in *Proc. of the ACS/IEEE Int. Conf. on Computer Systems and Applications*, 2001, pp. 144–150.

[12] Y.-C. Hu, R.-S. Chen, G.-H. Tzeng, and J.-H. Shieh, "A fuzzy data mining algorithm for finding sequential patterns," *International Journal of Uncertainty Fuzziness Knowledge-Based Systems*, vol. 11, no. 2, pp. 173–193, 2003.

[13] G.-H. T. Y.-C. Hu and C.-M. Chen, "Deriving two-stage learning sequences from knowledge in fuzzy sequential pattern mining," *Information Sciences*, vol. 159, pp. 69–86, 2004.

[14] R.-S. Chen and Y.-C. Hu, "A novel method for discovering fuzzy sequential patterns using the simple fuzzy partition method," *Journal of the American Society for Information Science*, vol. 54, no. 7, pp. 660–670, 2002.

[15] F. Masseglia, F. Cathala, and P. Poncelet, "The PSP approach for mining sequential patterns," in *Principles of Data Mining and Knowledge Discovery*, 1998, pp. 176–184.

[16] D. Dubois, E. Hüllermeier, and H. Prade, "A note on quality measures for fuzzy association rules," in *10th IFSA World Congress on Fuzzy Sets and Systems*, 2003, pp. 346–353.

[17] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach." *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 11, pp. 1424–1440, 2004.

[18] G. Melli, "Synthetic classification data set generator."

[19] I. H. Witten and E. Frank, "Data mining: Practical machine learning tools with java implementations," 2000.

[20] C. Fiot, A. Laurent, and M. Teisseire, "Web access log mining with soft sequential patterns," in *Proc. of the 7th Int. FLINS Conf. on Applied Artificial Intelligence*, 2006.

[21] C. Fiot, A. Laurent, M. Teisseire, and B. Laurent, "Why fuzzy sequential patterns can help data summarization: An application to the inpi trademark database," in *15th IEEE Int. Conf. on Fuzzy Systems*, 2006.

[22] A. Gyenesei and J. Teuhola, "Multidimensional fuzzy partitioning of attribute ranges for mining quantitative data: Research articles," *Int. J. Intell. Syst.*, vol. 19, no. 11, pp. 1111–1126, 2004.

[23] R. S. ans A. Goswami, "A fuzzy data mining algorithm for incremental mining of quantitative sequential patterns," *Int. J. of Uncertainty Fuzziness Knowledge-Based Systems*, vol. 13, no. 6, pp. 633–652, 2005.

**Céline Fiot** (fiot@lirmm.fr) completed a Master in Computer Engineering at the Ecole des Mines de Nantes, with a specialization in computer engineering for decision support systems. In 2004, she started a Ph. D. in Computer Science at the Laboratory of Computer Science, Robotics and Microelectronics in Montpellier, France. This Ph. D. work aims at mining approximative frequent sequences from imprecise, uncertain or incomplete data. Research topics are data mining, sequential patterns, association rules, fuzzy logic, missing values, incomplete databases and approximative knowledge discovery.

**Anne Laurent** (laurent@lirmm.fr) completed her PhD at the Computer Science Lab of Paris 6 in the Department of Learning and Knowledge Extraction, under the supervision of Bernadette Bouchon-Meunier. Her PhD research interests covered fuzzy data mining and fuzzy multidimensional databases. During the year 2002-2003, she joined the University Provence/Aix-Marseille I as a postdoctoral researcher and lecturer, working in the Database and Machine Learning group of the Fundamental Computer Science laboratory in Marseille, France. She has been an assistant professor in the University Montpellier 2 at the LIRMM laboratory since September, 2003, as a member of the Data Mining Group. She is interested in fuzzy data mining, multidimensional databases and sequential patterns, investigating and proposing new methods to tackle the problem of remaining scalable when dealing with fuzziness and complex data.

**Maguelonne Teisseire** (teisseire@lirmm.fr) received a Ph.D. degree in Computing Science from the Méditerrané University, France, in 1994. Her research interests focused on behavioral modeling and design. She is currently an Assistant Professor of Computer Science and Engineering in Montpellier II University and Polytech'Montpellier, France. She is the head of the Data Mining Group at the LIRMM Laboratory Lab, Montpellier, France, since 2000. Her research interest focus on advanced data mining approaches when considering that data are time ordered. Particularly, she is interested in text mining and sequential patterns. Her research takes part on different projects supported by either National Government (RNTL) or regional project. She has published numerous papers in refereed journals and conferences either on behavioral modeling or data mining.

## List of Tables

## List of Figures