HAL
open science

TED and EVA : Expressing Temporal Tendencies Among Quantitative Variables Using Fuzzy Sequential Patterns

Céline Fiot, Florent Masseglia, Anne Laurent, Maguelonne Teisseire

To cite this version:

Céline Fiot, Florent Masseglia, Anne Laurent, Maguelonne Teisseire. TED and EVA : Expressing Temporal Tendencies Among Quantitative Variables Using Fuzzy Sequential Patterns. WCCI4: World Congress on Computational Intelligence, Jun 2008, Hong Kong, China. pp.1861-1868, 10.1109/FUZZY.2008.4630623 . lirmm-00273907

HAL Id: lirmm-00273907

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00273907

Submitted on 18 Nov 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

TED and EVA: Expressing Temporal Tendencies among Quantitative Variables using Fuzzy Sequential Patterns

Céline Fiot, Florent Masseglia, Anne Laurent, and Maguelonne Teisseire

***Abstract*— Temporal data can be handled in many ways for discovering specific knowledge. Sequential pattern mining is one of these relevant approaches when dealing with temporally annotated data. It allows discovering frequent sequences embedded in the records. In the access data of a commercial Web site, one may, for instance, discover that *"5% of the users request the page register.php 3 times and then request the page help.html"*. However, symbolic or fuzzy sequential patterns, in their current form, do not allow extracting temporal tendencies that are typical of sequential data. By means of temporal tendency mining, one may discover in the same access data that *"an increasing number of requests to the register form preceeds an increasing number of accesses to the help page a few seconds later"*. It would be easy to conclude that the users either quickly succeed in registering or make several attempts before they look at the help page within a few seconds. In this paper, we propose the definition of evolution patterns that allow discovering such knowledge. We show how extracting evolution patterns thanks to fuzzy sequential pattern mining techniques. We introduce our algorithms TED and EVA, designed for evolution pattern mining. Our proposal is validated by experiments and a sample of extracted knowledge is discussed.**

I. INTRODUCTION

Many applications – network watching, web log analysis, customer management – record temporally annotated data that should be mined using specific data mining techniques, such as sequential pattern discovery algorithms, for instance.

Sequential patterns [1] are frequent sequences that can be found in sequence databases, i.e. datasets containing ordered sets of timestamped records, each of them consisting of a set of values. From a web server watching application, one could extract that *In 10% of sessions, identification failure preceeds request to forg_pwd.php followed later by access to my_account.php*.

As most of databases do not only contain binary attributes, but also numerical attributes such as connection duration, number of visitor per webpage, or download rate, generalizations of *sequential patterns* were designed.
Based on a discretization of numerical attribute domains into fuzzy sets, *fuzzy sequential patterns* [2], [3], [4] contain information about the numerical values frequently observed in the data and their correlations according to time. Thus these fuzzy sequences contain additional knowledge compared to crisp ones. For instance, the previous symbolic pattern could be more explicitly described by *In 10% of sessions, a lot of identification failures preceed one request to forg_pwd.php followed later by few requests to my_account.php*.

But these patterns model neither the evolution of the numerical values of the attributes in the dataset nor the correlations between evolutions of attribute values (what we will call *co-evolution* in this paper). Moreover a fuzzy sequential pattern describing that *ID_fail.htm has been repeatedly visited and then little visited* does not inform the end-user about the duration of the decrease nor about the dynamics or the intensity of change.

For this reason we introduce in this paper a method that mines for temporal trends using fuzzy sequential patterns, in order to extract rules like *First* A *quickly increases, then* B *slowly decreases while* C *increases*, defining *evolution patterns*.

Some works were done to describe such trends using linguistic summaries within the context of univariate time-series [5], [6], showing trends like "A *first quickly increases then slowly decreases*". Other ones extract common or recurrent patterns in multivariate time-series [7], [8], describing that several time-series have exactly the same profile. To the best of our knowledge there does not exist a data mining method that allows the discovery of typical co-evolutions, not necessary following the same trends, in sequence database.
Within the previous context, analysis of access logs from a website, such patterns could be for instance that *An increasing number of requests to registration.php during a short period preceeds an increasing number of requests to faq.html, after a very short period*. This knowledge would be explicit for the end-user (*is the registration-form easy to fill-in?*).

However, modeling such temporal knowledge requires to handle a very large number of elements – both in terms of attributes and records – during the mining task. Searching for evolution patterns indeed requires to compare each record of a data sequence to the following ones which leads to a combinatorial space complexity. So our goal is to design a tool that efficiently extracts temporal trends in quantitative data sequences.

The reminder of the paper is organized as follows. In the next section, we define the fundamental concepts associated with fuzzy sequential patterns and trend discovery in time-series. In Section III we introduce our approach, first defining evolution patterns, then detailing how to implement their discovery. Thus Section IV describes the algorithms underlying our approach. In Section V, we present some experiments on web access logs, showing the benefits of evolution pattern discovery. We finally conclude in Section VI.

Céline Fiot and Florent Masseglia are with the AxIS Research Team, INRIA Sophia-Antipolis, France (email: {celine.fiot, florent.masseglia}@sophia.inria.fr). Anne Laurent and Maguelonne Teisseire are with the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier, University of Montpellier, France (email: {laurent, teisseire}@lirmm.fr).

II. Patterns and Trends

Sequential patterns are often introduced as an extension of association rules in [9]. Initially proposed in [1], they highlight correlations between database records as well as their temporal relationships. Some generalizations were proposed to handle numerical attributes using fuzzy sets. In this paper we use fuzzy sequential patterns to mine evolution within quantitative data sequences.

A. Sequential Patterns

Sequential patterns are based on the idea of *maximal frequent sequences*.
Let $\mathcal{R}$ be a set of object 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 $\langle s_1 s_2 ... s_p \rangle$. An *n-sequence* is a sequence of n items (or of size n).

Example 1: Consider an example of market basket analysis. Objects are customers, and records are the transactions made by each customer. Timestamps are the dates of transactions. If a customer purchases products $e, a, k, u,$ and f according to the sequence $s = \langle (e)\ (a\ k)\ (u)\ (f) \rangle$, then all items of the sequence were bought separately, except products a and k which were purchased at the same time. In this example, s is a 5-sequence.

A sequence $S = \langle s_1\ s_2 ... s_p \rangle$ is a *subsequence* of another one $S' = \langle\ s'_1\ s'_2\ ... s'_m \rangle$ 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' = ⟨(a) (f)⟩ is a subsequence of s because (a) ⊆ (a k) and (f) ⊆ (f). However, ⟨(a) (k)⟩ is not a subsequence of s.

All records from the same object o are grouped together and sorted in increasing order of their timestamp, constituting 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).

Example 3: Within the context of web usage mining, an object would be for instance one IP and the data sequences would be the click sequences associated with each IP. Each URL would be encoded into an item.

The *frequency* of a sequence ($freq(s)$) is defined as the percentage of objects supporting s in the whole set of objects $\mathcal{O}$. 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 of which the frequency is greater than a specified threshold ($minFreq$) [1]. Each of these sequences represents a *sequential pattern*, also called a maximal frequent sequence.

Several extensions were proposed to handle numerical and quantitative values [2], [3], [4], to generalize sequential patterns with respect to various temporal parameters (time-interval between events of a sequence, grouping several records into a single itemset...) [10], [11], or even to deal with missing values [12].

B. Fuzzy Sequential Patterns

In order to allow for handling numerical or quantitative information several works proposed to partition each numerical attribute into several fuzzy sets. The quantitative database is thus converted into a membership degree database, which is then mined for fuzzy sequential patterns.

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

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: $([candy, lot][soda, little])$ is a fuzzy itemset and can also be denoted by $((candy, soda)(lot, little))$.

Last a g-k-**sequence** $S = \langle s_1 \cdots s_g \rangle$ is a sequence constituted by g fuzzy itemsets $s = (X, A)$ grouping together k fuzzy items $[x, a]$.

Example 6: The sequence $\langle ([soda, lot]\ \ [candy, lot])\ ([videogames, little]) \rangle$ 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 $\varrho[x]$ the value of attribute x in record ϱ. One record ϱ in a fuzzy sequence database (or membership degree database) consists of the membership degrees of attributes to each fuzzy set, e.g $r(x, a) = \mu_a(\varrho[x])$ is the

value of record r for the fuzzy item $[x, a]$. It represents the membership degree of the quantity $\varrho[x]$ of item/attribute x to the fuzzy set a in record ρ.

The frequency of a fuzzy sequence S is then computed by the formula 1:

$$FFreq(S) = \frac{\sum_{o \in \mathcal{O}} \varphi(S, o)}{|\mathcal{O}|} \quad (1)$$

where $\varphi(S, o)$ gives the degree to which S is included into the object o data sequence.

This degree is computed by considering the best appearance – i.e. the appearance with the highest degree – of the ordered list of itemsets of S. It is computed by formula 2:

$$\varphi(S, o) = \underline{\perp}_{\varsigma \subseteq \zeta_o | S = \varsigma = \langle s_1 \ldots s_i \ldots s_k \rangle} \overline{\top}_{s_1 \ldots s_k} (\overline{\top}_{j \in s_i} \mu(j)) \quad (2)$$

where k is the number of itemsets in S, ζ_o is the set of sequences included in the data sequence of object o and $\overline{\top}$ and $\underline{\perp}$ are the t-norm and t-conorm operators generalized to n-ary cases. In practice, we use the Zadeh t-norm and t-conorm, $\min$ and $\max$.

C. Modeling Trends in Time-Series

Evolution and trend discovery within temporal data is an important research area, since it as lots of industrial applications. Thus, linguistic summaries or time-series segmentation and representation [13], [14] have led to interesting work. In particular, [5], [6] identify trends in time series to make them understandable for a human being.

In [15], as in many works, time-series are analysed to detect anomalies. Regarding multi-variate time-series, most of approaches aim at analysing parallel evolution of several time-series, each related to one numerical attribute using clustering or multiple alignments.

However the data we mined, web access logs, cannot be considered as time-series even multi-variate ones, since they are discontinuous and irregularly collected. Actually all the records do not necessary contain values for every attributes. Moreover time periods between events may be irregular and timestamps may be different from one object to the other.

Therefore we more specifically focused on proposals that discuss methods for trend discovery in data sequences. Apart from [16] that mines for emerging patterns and [17] that highlights trends in sentences of textual databases, there does not exist an approach for discovering frequent evolutions or duration analysis, within sequential data.

III. TED: MODELLING TRENDS IN QUANTITATIVE ATTRIBUTES

The objective of this work is to discover and express evolutions among the quantitative attributes of different objects in a sequence database. For instance, an evolution pattern could be *The number of requests to registration.php increases followed later by an increasing number of requests to faq.html*. Moreover we propose to use the fuzzy data sequence formalism to discover an additional information, expressing in the previous example what should be *"later"* for this pattern: *few minutes, half an hour, ...*

In order to mine such evolution patterns we thus propose to process the original quantitative sequence database into a trend database that will be mined by our algorithm EVA. In this section, we describe the concepts of evolution sequences and detail our approach for evolution pattern mining.

A. Overall Principle

The global process can be described by figure 1.

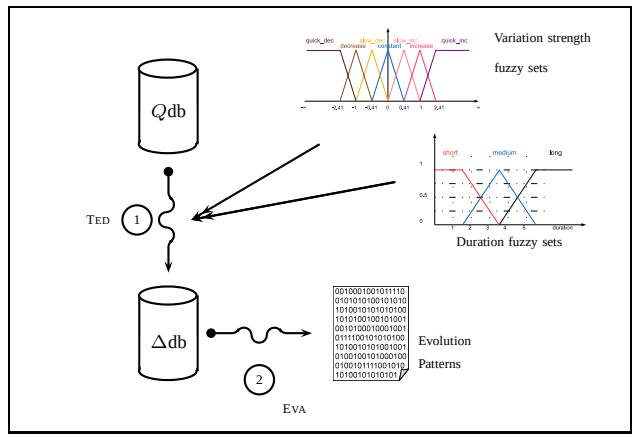

Fig. 1. Overall principle of our approach

First the quantitative database (Qdb, e.g. Table I) is converted into a *variation database*. Then, such as it is done for fuzzy sequential pattern mining, this dataset is converted into a membership degree database (Δdb on Figure 1, e.g. Table III), using predefined fuzzy sets – automatically or from expert knowledge designed. There are two fuzzy sets partitions: one gives the linguistic terms describing variation strength, the second describes the duration of the trend. These steps are detailed in subsections III-C and III-D.

This membership degree database Δdb is the *trend database*. It is the dataset mined for evolution patterns (step 3 in Figure 1) as it is described in subsection IV-A.

Example 7: $\langle([x, 4])([x, 3][y, 5][z, 8])([x, 2][y, 4][z, 10])([y, 6])\rangle$ is a sequence characterizing the number of connections to URL x, y and z during successive sessions of one identified IP o. Table I represents this sequence as a quantitative database, Qdb. It contains four ordered records for the IP o.

TABLE I
A QUANTITATIVE DATA SEQUENCE.

	date	x	y	z	
d1	4	4			r^1
d2	7	3	5	8	r^2
d3	8	2	4	10	r^3
d4	10		6		r^4

Then Table III, in Subsection III-C, gives the trend database Δdb drawned from Table I, after step 1, the execution of the algorithm TED.

B. Evolution Patterns

We propose to mine for evolution patterns. So data sequences are modelled such that each item will express a variation, for instance *"increasing number of requests"*.

Each record in this trend database represents the evolution that actually happened between two records related to the same object within the original dataset. Records in the trend database contain items that were created from the same start and end records of the initial quantitative dataset. We define a *trend dataset* as a set of data sequences made of *evolution items*. An evolution item denote a trend – increase, decrease or constancy – of a quantitative attribute. The variation strength may be considered using fuzzy sets: an evolution item is defined as a fuzzy item $[x, v]$ in which x is a quantitative attribute of the original dataset and v a fuzzy set representing both trend and strength of the variation of x value. For instance, using the trends granules described by Figure 2, from [5], an evolution item *[nb_faq.html, quick_inc* would for instance mean that *the number of requests to faq.html page quickly increases*. Each evolution item is associated with a membership degree that more precisely describes the strength of the variation. Details are given in the next subsection.

Then an evolution itemset can be defined as a non-ordered, non-empty set of evolution items. It represents the *co-evolution* of several attributes, i.e. the variation of several attributes over a given time-period. And an *evolution sequence* is an ordered list of evolution itemsets. It describes successive trends in the original quantitative dataset. An evolution itemset is denoted by parenthesis ($[x, inc][y, dec]$) and a sequence by angles $\langle([x, inc][y, dec])\ ([z, q_inc])\rangle$.

Thus a trend database is a membership degree database in which fuzzy items describe variation strength of quantitative attributes. However this dataset cannot be exactly considered as a sequence database that could be mined using fuzzy sequential pattern algorithms, as described in [2], [3], [4].
Each record is related to one object and to two different timestamps. One timestamp corresponds to the starting date of the observed variation, the second is the ending date. Moreover there can be several records that have the same starting date, since the evolution dataset stores the evolution between each pair of records of Qdb that have common quantitative attributes.

Example 8: Consider for instance the evolution of the number of requests to URL y in Tab. I, starting from d2: two variations will be observed, one between d2 and d3, the second between d2 and d4.

For this reason mining for evolution patterns is not a mere application of fuzzy sequential patterns after a tricky data preprocessing. In the next subsection, we describe how the evolution dataset is built using the algorithm TED, that handles trends and durations, and in section IV we detail the algorithm, EVA, that mines for evolution patterns.

C. Trend Databases

Each evolution item $[x, v]$ in the evolution database Δdb represents the variation of one quantitative attribute between two successive records r^1 and r^2, related to a same object o in the quantitative database Qdb. Then each record in Δdb is built by the combination of two records of the initial Qdb dataset. More specifically, for each ordered pair of records r^i and r^j of one data sequence, such that $r^i(x^i)$ and $r^j(x^j)$ are filled-in, a *trend record* $\Delta r_{r^i r^j}$ is created in Δdb containing the evolution item $[x, TrendGran]$, where $TrendGran$ is the trend granule corresponding to the variation strength.

Example 9: From records r^1 and r^2 in Table I, an evolution record $\Delta r_{r^1 r^2}$ containing the evolution item $[x, decreasing]$ can be created as the quantity recorded in database for x in r^1 is greater than the one recorded in r^2.

The temporal order initially existing within the quantitative database is kept, for each object o, by chronologically ordering the trend records according to the timestamps of r^i and r^j.

Example 10: Consider once again the variations of the number of requests to URL y in Tab. I. Three evolution records can be created for this item: one combining d2 and d3, the second from d3 and d4, the last from d2 and d4. These evolution records will be ordered first considering their starting date, then their ending date. So the evolution records for y would be ordered as follows : first the one combining d2 and d3, then the one combining d2 and d4, last the one combining d3 and d4.

To define the trend between two records, we assimilate a pair of records r^i, r^j to two points and we evaluate the slope of the line going fro r^i, defined by ($r^i[x^i]$, d_i), to r^j, defined by ($r^j[x^j]$, d_j). Then the linguistic terms representing the trend are chosen considering the trend granules on Figure 2, and the associated membership degrees are given by fuzzy sets, for instance described by Figure 3.

Note that there are many methods for constructing fuzzy granulation of directions [18]. The user may also define membership functions of particular linguistic terms depending on his/her needs.

Example 11: From records r^1 and r^2 in Table I, the evolution item created in Example 9 is actually associated to the trend *slowly decreasing* as the slope between r^1 and r^2 is equal to $\frac{3-4}{7-4} = -1/3$. Moreover, from Figure 3, this slope corresponds to a membership degree of 0.8 for slowly decreasing and of 0.2 for constant.

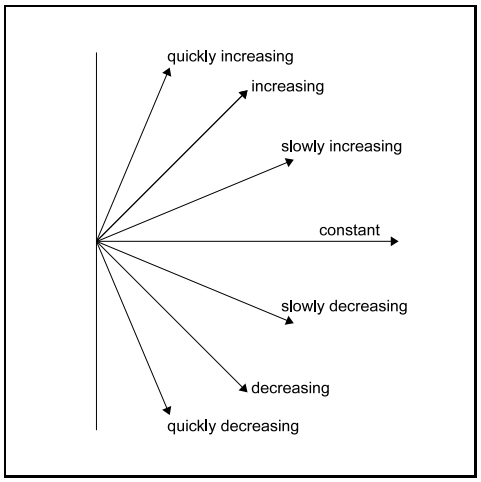

Fig. 2. Trend granules

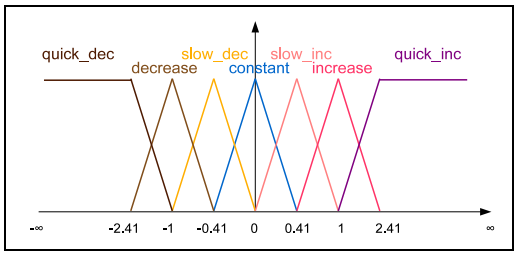

Fig. 3. Trend fuzzy sets

From records r^3 and r^4 in Table I, the evolution item created for attribute y corresponds to the slope $\frac{6-4}{10-8} = 1$. The trend record created in Δdb is then $\Delta r_{r^3 r^4}([x, increasing]) = 1$.

Table II is the trend database obtained from Table I.

TABLE II

EVOLUTION DATA SEQUENCE OBTAINED FROM TABLE I.

		x			y				z		
		dec	s.dec	cst	dec	cst	s.inc	inc	inc	q.inc	
d1	d2		0.8	0.2							δ^1
d1	d3	0.15	0.85								δ^2
d2	d3	1			1				0.3	0.7	δ^3
d2	d4					0.2	0.8				δ^4
d3	d4							1			δ^5

Having this set of records for each data sequence o in $\mathcal{O}$, we could then go to the mining step. However one type of information, still available in a sequence database, would be lost. We are indeed in a context of temporally annotated records. So comparing two records to generate an evolution item with a membership degree could lead to an additional entry in the trend dataset: timestamps of both records may also be compared thus creating an additional fuzzy item in $\Delta r_{r^i r^j}$ describing duration between r^i and r^j. We express this duration item by linguistic terms computed from the difference between timestamps of r^i and r^j.

Example 12: Considering the fuzzy sets given by Figure 4 for duration, the final trend database obtained from Table I is described by Table III.

TABLE III

TREND DATA SEQUENCE OBTAINED FROM TABLE I.

x			y				z		duration		
dec	s.dec	cst	dec	cst	s.inc	inc	inc	q.inc	sh.	m.	lg.
	0.8	0.2							0.25	0.75	
0.15	0.85									0.25	0.75
1			1				0.3	0.7	1		
				0.2	0.8				0.25	0.75	
						1				0.75	0.25

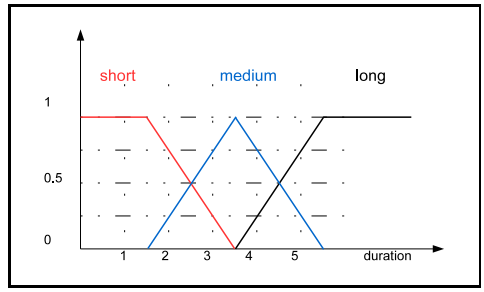

Fig. 4. Duration linguistic terms

Finally, the trend database consists of a set of trend records $\Delta r_{r^i r^j}$, each consisting of:

- one object-id o, corresponding to the id of r^i and r^j,
- r^i timestamp, denoted by $t(r^i)$,
- r^j timestamp, denoted by $t(r^j)$,
- a set of evolution items $[x, v]$
- fuzzy duration items

$$\Delta r_{r^i r^j}(\delta) = \mu_d(t(r^j) - t(r^i))$$

D. The Algorithm TED

The process described in the previous paragraphs (step 1 on figure 1) is done using the algorithm TED, described by figure 5.

It parses the records of the quantitative dataset. For each of them, r, the following records in the same data sequence are parsed to find the attributes of r. For each pair of records containing common attributes, a record is created in the trend database, consisting of the evolution items and fuzzy duration items. Each of them is associated with a membership degree.

TED ***Main*** - **Input:** Qdb
Ouput: Δbd

Δdb.*initialize*();
For each data sequence $o \in Q$db **do**
  For each record $r \in \mathcal{R}_o$ **do**
    For each record $r' \in \mathcal{R}_o$ / $t(r') > t(r)$ **do**
      Δr.*initialize*();
      For each attribute a **do**
        If (($r[a]$!= NULL) **AND** ($r'[a]$!= NULL)) **Then**
          Δr.*add*(a, v, $\mu_v(r[a] - r'[a])$);
          [where v gives the variation strength of the trend]
        End If
      End For
      Δdb.*add*(o,Δr, $t(r)$, $t(r')$, d, $\mu_d(t(r') - t(r))$);
      [where d gives the duration length fuzzy set]
    End For
  End For
End For
return Δdb ;

Fig. 5. TED algorithm

The algorithm TED creates the trend database within a temporal complexity of $O(n^2)$, with n the number of records in the quantitative database. In the worst case, the trend database contains $\sum_{o \in \mathcal{O}} \frac{|\mathcal{R}_o|(|\mathcal{R}_o|-1)}{2}$ records.

IV. EVA: AN ALGORITHM FOR EVOLUTION PATTERN MINING

We chose to implement our algorithm on the ground of a level-wise principle. This kind of algorithm uses the frequent sequences of size k to generate *candidate* – possibly frequent – sequences of size $k+1$. Then the frequency of these candidate sequences is calculated, only the frequent ones being stored. Mining the trend database for evolution patterns would then be done on the principles of TOTALLYFUZZY described in [4], for fuzzy sequential patterns.

However, the direct application of this algorithm or of any sequential pattern mining algorithm is not possible due to the specific format of the trend data. Therefore we designed the algorithm EVA to mine for evolution patterns.

In this section we first explain why the specific format of trend data does not allow us to use existing algorithms. Then we introduce our solution to handle the record chronology. Last we detail the overall mining algorithm EVA.

A. About duration

Since each trend record $\Delta r_{r^i r^j}$ has been built from two records of the initial dataset, it contains two timestamps $t(r^i)$ and $t(r^j)$ that describe a duration. Since we search for sequences we need to define an order and/or constraints to be satisfied between trend records, taking into account the original timestamps $t(r^i)$ and $t(r^j)$.

Several trend records may have the same starting timestamp $t(r^1)$, so they cannot be included into the same sequence. But they correspond to the same object in the dataset. Moreover, one record covering the time-period from $t(r^1)$ to $t(r^3)$ does not preceed or follow a record including r^2 that happened between r^1 and r^3 in the original Qdb [19]. Example 13 illustrates these cases based on Table III.

Example 13: Figure 6 represents each gradual record of Table III. The second trend record overlaps the first and third ones and the fourth record overlaps the third and fifth ones.

To take into account the possible overlaps of itemsets, the sequence database Δdb should be parsed with lots of forward and backward phases during examination of data sequences. To avoid such expensive parsing of the data, we designed a method that skips overlapping itemsets for one candidate sequence.

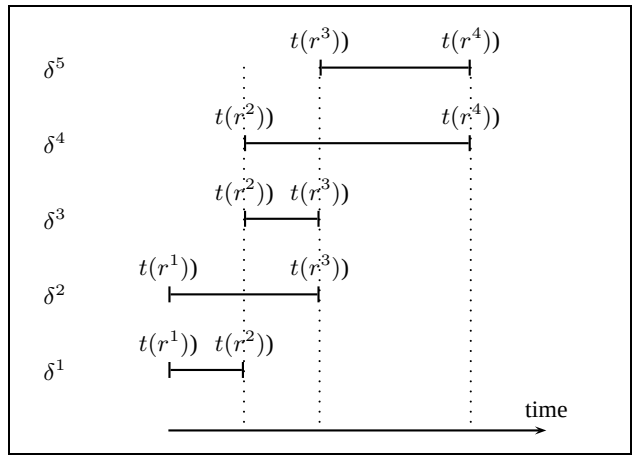

Fig. 6. Overlapping gradual records

B. Handling Record Chronology

Our approach uses a graph structure to represent allowed sequences in a data sequence. The principles of this model are quite similar to those developed in [11] to handle time constraints in sequential pattern mining. Vertices in the sequence graph are trend records and edges represents sequences.

Thus before extracting the evolution patterns, EVA preprocesses each data sequence of the trend database into a sequence graph. Then these sequence graphs are parsed to discover evolution patterns.

For each object o in Δdb, the sequence graph is built by the function *createGraph*, Figure 7, called by EVA main function. The records in Δdb are chronologycally ordered according to their start timestamp then to their end timestamp. The function *createGraph* scans the ordered list of records. First one vertex is created for each record within the data sequence; the functions v.end() and v.start() respectively return the end and start timestamps of the record associated to vertex v.

During the second step edges are created. For each vertex in the graph, *createGraph* creates the edges that correspond to allowed sequences, i.e. for two vertices v_i and v_j, an edge is built from v_i to v_j iff v_j.start() $> v_i$.end().

For this reason, when they are created, vertices are attached to one set according to their start-time. Thus for creating edges, *createGraph* only links each vertex v to the vertices in the first set l whose stat-time l.start is greater than or equal to v.end().

Figure 8 represents the sequence graph obtained from the trend sequence given by Table III. From the data sequences in Table III we can build four longest sequences to mine evolution patterns: $\langle \delta^1 \delta^3 \delta^5 \rangle$, $\langle \delta^2 \delta^5 \rangle$ and $\langle \delta^1 \delta^4 \rangle$.

Once the sequence graphs are created, the extraction of evolution patterns starts.

createGraph - **Input:** δs, one data sequence of Qdb
Ouput: *($\mathcal{GV}, \mathcal{GE}$)*, the sequence graph for δs

```
GV.initialize();
GE.initialize();
L.initialize();
r ← δs.first();
L.start_time ← start(r);
While (δs.hasNext()) do
    r ← δs.next();
    If (start(r) == L.start_time) Then
        L.addVertex(new_Vertex(r));
    Else
        GV.add(L);
        L.initialize();
        L.start_time ← start(r);
        L.addVertex(new_Vertex(r));
    End If
End While
GV.add(L);
For each vertex u ∈ GV do
    tmpL ← u.getSet();
    While (tmpL.start_time < u.end()) do
        tmpL ← GV.getNextSet();
    End While
    If (tmpL.notEmpty()) Then
        For each vertex v ∈ tmpL do
            GE.addEdge(u,v);
        End For
    End If
End For
return (GV, GE);
```

Fig. 7. *createGraph* function

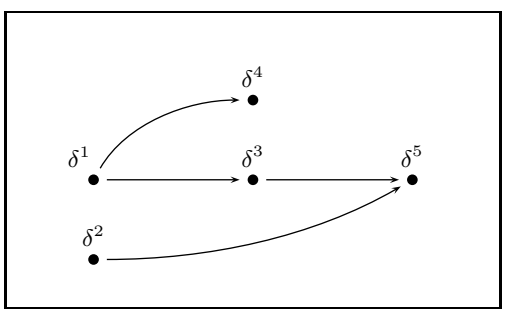

Fig. 8. Sequence graph for the data sequence in Table III.

C. The Algorithm EVA

The global algorithm for mining EVolution pAtterns, Figure 9, can be described as follows.

Once the trend database has been generated by TED, the function *createGraph* is called to process each data sequence into a graph, thus handling time constraints due to duration.

Then the sequence graphs are parsed to discover frequent evolution items, according to one user-defined minimal frequency threshold $minFreq$. After this step, the frequent elements of size k are combined into candidate sequences of size $k+1$. These sequences are searched within the graphs and EVA computes their frequency using the algorithm TotallyFuzzy from [4] that implements the formula 1 given in section II. EVA stops when no more candidate sequences has been found frequent.

EVA ***Main*** - **Input:** $minFreq$, Δdb
Ouput: F, frequent evolution sequences

```
F_0 ← ∅ ; k ← 1 ;
F_1 ← {{< i >}/i ∈ I&freq(i) > minFreq};
For each trend data sequence δS ∈ Δdb do
    graphDB ← createGraph(δS) ;
End For
While (Candidate(k) ≠ ∅) do
    For each sequence graph g ∈ graphDB do
        [countFrequency is a version of the TotallyFuzzy algorithm]
        [adapted to sequence graph parsing]
        countFrequency(Candidate(k), minFreq, g) ;
    End For
    F_k ← {s ∈ Candidate(k)/freq(s) > minFreq};
    Candidate(k + 1) ← generate(F_k) ;
    k++;
End While
return F ← ⋃_{j=0}^{k} F_k
```

Fig. 9. EVA: Main algorithm

The soundness and completeness of such approaches, based on sequence graphs, have been proved [20]: all the supported sequences and only them are created in the sequence graphs. So at the end of the process we obtained all the evolution patterns contained in the trend dataset.

V. EXPERIMENTS

The aim of these experiments is to apply gradual sequential patterns for web usage analysis.

A. Data

In our case, access logs from a laboratory website have been prepared and mined to find repeatedly visited pages. Records contain the number of access to one page, the same half-day by one user. For example, record *"1500 5067 10 6"* means that *"visitor 1500"* on half-day *5067* visited *6* times the URL coded by *10*. This dataset contains 27209 web pages visited by 79756 different IPs during 16 days (32 half-days). The translation into the formalism given in sections II and III is given by Tab. IV.

TABLE IV
DATA SEQUENCES FOR WEB USAGE MINING

Object	↔	IP
Timestamp	↔	halfday
Quantitative items	↔	# of accesses to each web page
Evolution items	↔	variation of the # of accesses to each web page
Duration	↔	time period between two accesses to one web page

As detailed in section III-A, quantities are compared and, thanks to our algorithm TED, the dataset is converted into a trend database that contain evolution items and the fuzzy sets membership degrees. Then these data are mined by EVA.

B. Results

The runtime performances of our algorithm are similar to fuzzy sequential pattern algorithms, Figure 10. As the minimum frequency of the gradual patterns wanted decreases. This is due to the increasing number of frequent sequences that leads to a proportional increase of the number of scans on the dataset.

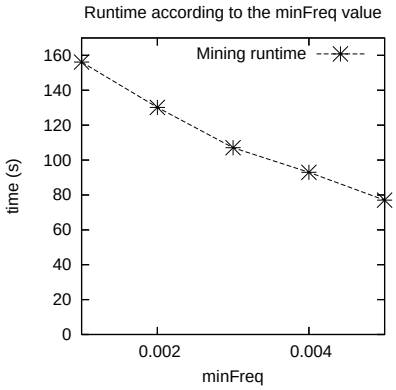

Fig. 10. Runtime according to $minFreq$.

Regarding the qualitative analysis, discovered patterns are relevant. One typical evolution pattern we discovered on the log file of INRIA Sophia is related to the Koala project and expresses the following temporal tendency *"A slow increase of the number of connections to page KBM preceeds a long period of an increase of the connections to KOML, that occurs after a short period. Then follows a slow increase of the number of connections to DJAVA"*.

VI. Conclusion

In this paper we introduced an approach for discovering typical evolutions and durations in numerical sequence databases. This approach is based on fuzzy sequential patterns that are used to mine trend data sequences.

In contrast to time-series analysis, in the context of data sequences, some attributes may be unfilled in some records. Moreover duration between two consecutive records is not necessarily regular. Therfore, describing trends in such numerical sequence datasets requires specific approaches.

For this reason, we proposed a process based on two algorithms. TED converts a numerical database into a trend database, describing evolution of numerical attribute values, according to time for several objects. These evolutions are represented as trend sequences. Then EVA searches for frequent evolution sequences in this trend sequence dataset.

Discovered evolution patterns would for instance inform that *An increasing number of requests to registration.php during a short period preceeds an increasing number of requests to faq.html, after a very short period*. These temporal relations among web page browsing could then be used to improve web site architecture and quality of services. We actually applied an implementation of this process to access logs of a website and discovered relevant knowledge.

Extensions of this work could lead to temporal implication, describing causal relationships between evolution of attributes. It would then include some statistical results implying search for dependencies based on linear regressions. The discovered knowledge would for instance help in explaining some web server failures.

References

[1] R. Agrawal and R. Srikant, "Mining sequential patterns," in *11th International Conference on Data Engineering*, 1995, pp. 3–14.

[2] T. Hong, K. Lin, and S. Wang, "Mining Fuzzy Sequential Patterns from Multiple-Items Transactions," in *Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, 2001, pp. 1317–1321.

[3] R.-S. Chen, G.-H. Tzeng, C.-C. Chen, and Y.-C. Hu, "Discovery of Fuzzy Sequential Patterns for Fuzzy Partitions in Quantitative Attributes," in *ACS/IEEE Int. Conf. on Computer Systems and Applications*, 2001, pp. 144–150.

[4] C. Fiot, A. Laurent, and M. Teisseire, "From crispness to fuzziness: Three algorithms for soft sequential pattern mining," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 6, pp. 1263–1277, 2007.

[5] J. Kacprzyk, A. Wilbik, and S. Zadrozny, "Capturing the essence of a dynamic behavior of sequences of numerical data using elements of a quasi-natural language," in *IEEE International Conference on Systems, Man and Cybernetics (SMC'06)*, 2006, pp. 3365–3370.

[6] ——, "Linguistic summaries of time series via an owa operator based aggregation of partial trends," in *IEEE International Conference on Fuzzy Systems (FuzzIEEE'07)*, 2007, pp. 1–6.

[7] F. Duchêne, C. Garbay, and V. Rialle, "Learning recurrent behaviors from heterogeneous multivariate time-series," *Artificial Intelligence in Medicine*, vol. 39, no. 1, pp. 25–47, 2007.

[8] F. Mörchen and A. Ultsch, "Discovering temporal knowledge in multivariate time series," in *28th Conference of the German Classification Society*, 2005, pp. 272–279.

[9] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *ACM SIGMOD International Conference on Management of Data*, 1993, pp. 207–216.

[10] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *5th International Conference on Extending Database Technology*, 1996, pp. 3–17.

[11] F. Masseglia, P. Poncelet, and M. Teisseire, "Pre-Processing Time Constraints for Efficiently Mining Generalized Sequential Patterns," in *11th International Symposium on Temporal Representation and Reasoning*, 2004, pp. 87–95.

[12] C. Fiot, A. Laurent, and M. Teisseire, "Approximate sequential patterns for incomplete sequence database mining," in *16th IEEE International Conference on Fuzzy Systems (FuzzIEEE'07)*, 2007.

[13] J. Sklansky and V. Gonzalez, "Fast polygonal approximation of digitized curves," *Pattern Recognition*, vol. 12, no. 5, pp. 327–331, 1980.

[14] B. Hugueney and B. Bouchon-Meunier, "Time-series segmentation and symbolic representation, from process-monitoring to data-mining," in *Computational Intelligence. Theory and Applications : International Conference, 7th Fuzzy Days*, 2001, pp. 1611–3349.

[15] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *5th IEEE International Conference on Data Mining (ICDM '05)*, 2005, pp. 226–233.

[16] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *5th International Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999, pp. 43–52.

[17] B. Lent, R. Agrawal, and R. Srikant, "Discovering trends in text databases," in *3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*, 1997, pp. 227–230.

[18] I. Batyrshin, "On granular derivatives and the solution of a granular initial value problem," *International Journal Applied Mathematics and Computer Science*, vol. 12, no. 3, pp. 403–410, 2002.

[19] J. F. Allen, "Maintaining knowledge about temporal intervals," *Readings in qualitative reasoning about physical systems*, pp. 361–372, 1990.

[20] C. Fiot, A. Laurent, and M. Teisseire, "Extended time constraints for generalized sequential patterns," LIRMM, Tech. Rep. 6051, 2005.