HYPE: Mining Hierarchical Sequential Patterns

Marc Plantevit, Anne Laurent, Maguelonne Teisseire

HYPE: Mining Hierarchical Sequential Pattern Mining

Marc Plantevit
LIRMM-Univ. Montpellier2
Montpellier, France
marc.plantevit@lirmm.fr

Anne Laurent
LIRMM-Polytech'Montpellier
Montpellier, France
laurent@lirmm.fr

Maguelonne Teisseire
LIRMM-Polytech'Montpellier
Montpellier, France
teisseire@lirmm.fr

ABSTRACT

Mining data warehouses is still an open problem as few approaches really take into account the specifities of this framework (e.g. multidimensionnality, hierarchies, historized data). Multidimensional sequential patterns have been studied. However, they do not provide any way to handle hierarchies. In this paper, we propose an original method of extraction of sequential patterns taking into account the hierarchies. This method extracts more accurate knowledge and extends our preceding approach M^2SP. We define the concepts related to our problems as well as the associated algorithms. The experiments which we carried out show the interest of our proposal.

Keywords

Multidimensional sequential patterns, hierarchies, OLAP.

1. INTRODUCTION

The techniques of data mining can provide a considerable help in the OLAP framework ([5]) where the user must make the best adapted decisions in a minimum of time. More precisely, data mining is a key step in the decisional process considering large volumes of multidimensional data. Indeed, mined patterns or rules allow another view of the original data. However, discovering such rules needs some parameters. In particular, it needs minimal support which corresponds to the minimal frequency the patterns occur within the database. If the selected minimal support is too high, the number of rules discovered is weak and the rules are too general to be useful. If the support is too low, the number of mined rules is very important and makes difficult the analysis of those. The decision maker is then confronted with the following problem: how to lower the minimal support without generating the discovery of non-relevant rules? Or how to increase the minimal support without losing the useful rules? Is it then necessary to make a trade-off between the quality of extracted knowledge and the minimal support?

In this context, using hierarchies can help to solve this dilemma. It makes it possible to discover rules within several levels of hierarchies. Thus, even if a high support is used, important knowledge which support is weak in the database can be *included* in more general knowledge which will be considered frequent. We thus wish to extend our preceding proposal to mine multidimensional sequential patterns by taking hierarchies into account.

Sequential patterns have been studied for more than ten years [1], with a lot of research and industrial applications (*e.g.* user behavior, web log analysis, discovery of patterns from proteins' sequences, security). Algorithms have been proposed, based on the APriori-based framework [16, 10, 2], or on other approaches [11, 7]. Some other work has been done on the discovery of frequent episods [9]. Sequential patterns have recently been extended to multidimensional sequential patterns by Pinto et al. [12], Plantevit et al. [13], and Yu et al. [15]. They aim at discovering patterns that take time into account and that involve several dimensions. For instance in [13], rules like *A customer who bought a surfboard together with a bag in NY later bought a wetsuit in SF* are discovered.

Even if some approaches provide the taking into account of the hierarchies in the extraction of sequential patterns, there does not exist, to our best knowledge, any work combining the extraction of multidimensional sequential patterns and the management of the hierarchies. No current method can extract knowledge like: *When the sales of soft drinks increase in Europe, exports of Perrier increase in France and exports of soda increase in the USA,* where *Perrier* is a kind of French carboned soft drink. We propose a novel approach *HYPE* (HierarchY Pattern Extension) which is an extension of our previous proposition M^2SP [13]. The originality of our approach lies in the idea that no single level of hierarchy is fixed and that several levels can be mixed. The extracted sequential patterns are automatically associated to the most adequate levels of hierarchies.

In this paper, we present the concepts related to the traditional sequential patterns and multidimensional ones as well as the approaches managing hierarchies during the extraction of knowledge. We introduce then the fundamental concepts related to our approach HYPE as well as the algorithms allowing its implementation. Experiments carried out on synthetic data are reported and confirm the interest of our approach. We also show that using the hierarchies allows a better management of the joker values defined in the M^2SP approach.

2. HIERARCHIES AND DATAMINING

In this section, we present the sequential patterns as well as the approaches of the literature having dealt with the problem of the extraction of sequential pattern in a multidimensional framework

(several analysis dimensions). Then, we underline why it is relevant to use the hierarchies during the process of extraction of sequential patterns and wa make an overview of associated work.

2.1 Sequential Patterns

An early example of research in the discovering of patterns from sequences of events can be found in [4]. In this work, the idea is the discovery of rules underlying the generation of a given sequence in order to predict a plausible sequence continuation. This idea is then extended to the discovery of finding interesting patterns (or *rules*) embedded in a database of sequences of sets of events (items). A more formal approach in solving the problem of mining sequential patterns is the AprioriAll algorithm as presented in [9]. Given a database of sequences, where each sequence is a list of transactions ordered by transaction time, and each transaction is a set of items, the goal is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data-sequences that contain the pattern. In [1], the authors introduce the problem of mining sequential patterns over large databases of customer transactions where each transaction consists of customer-id, transaction time, and the items bought in the transaction. Formally, given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min support threshold, sequential pattern mining is to find all of the frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min support. Sequential pattern mining discovers frequent patterns ordered by time. An example of this type of pattern is *A customer who bought a new television 3 months ago, is likely to buy a DVD player now*. The main stake of sequential pattern mining methods is then the most effective extraction. Algorithms have been proposed, based on the APriori-based framework [16, 10, 2], or on other approaches [11, 7].

In the traditional framework (only one analysis dimension) for association rule or sequential pattern extraction, there are several works which take into account the hierarchies in order to allow an extraction of accurate knowledge.

In [14], the beginnings of the hierarchy management in the extraction of association rules and sequential patterns are proposed. The authors suppose that the hierarchical relations between the items are represented by a set of *taxonomies*. They make it possible to extract association rules or sequential reasons according to several levels of hierarchy. They modify the transactions by adding for each item all of its ancestors in associated taxonomy. Then, they generate the frequent sequences while trying to filter with the maximun redundant information and by optimizing the process using several properties. However, this approach cannot be scalable in a multidimensional context. Indeed, to add on each dimension the list of the ancestors of one item in taxonomy, for each transaction, is unthinkable. That would be equivalent, in the worst case, to multiply the size of the database by the maximum depth of a hierarchy and this for each dimension of analysis, scan of this basis would be then too much expensive.

The approach of J. Han et al. [6] is quite different. The authors tackle the association rule extraction problem, but this approach can be adapted to sequential pattern extraction. Begining at the highest level of the hierarchy, they extract the rules on each level while lowering the support when going down in the hierarchy. The process is reiterated until no rules can be extracted or until the lowest level of the hierarchy. However, this method does not make it possible to extract rules containing items of different levels. For example $wine$ and $drinks$ cannot cohabit in such a rule. This method thus proposes the extraction of *intra level of hierarchy* association rules. It thus does not make it possible to answer the general problems of extraction of the sequences *on various levels of hierarchy*. Furthermore, the implementation of this approach in a multidimensional context can be discussed. If several taxonomies exist (one by dimension), does one have to move on the same levels of hierarchy on various taxonomies or to combine these levels? This type of extraction can be expensive in time, because the mechanism of extraction of knowledge can be reiterated several times (depth of taxonomy), which is not inconsiderable.

We have presented the sequential patterns as well as works allowing the taking into account of the hierarchies in the extraction of knowledge. Nevertheless the sequential patterns are sometimes quite poor compared to the data they describe. Indeed, the correlations are extracted within only one dimension (e.g. the *product* dimension) whereas a database can contain several other dimensions. This is why several works try to combine several analysis dimensions in the extraction of sequential patterns.

2.2 Multidimensional Sequential Patterns

Combining several analysis dimensions makes it possible to extract knowledge which describes the data in a better way. [12] is the first paper dealing with several dimensions in the framework of sequential patterns. For instance, purchases are not only described by considering the customer ID and the products, but also by considering the age, type of the customer (Cust-Grp) and the city where he lives. Multidimensional sequential patterns are defined over the schema $A_1, ..., A_m, S$ where the set of A_i stands for the dimensions describing the data and S stands for the sequence of items purchased by the customers ordered over time. A multidimensional sequential pattern is defined as (id_1,$(a_1, ..., a_m)$,s) where $a_i \in A_i \cup \{*\}$. id_1,$(a_1, ..., a_m)$ is said to be a multidimensional pattern. For instance, the authors consider the sequence $((*, NY, *)$,$\langle bf \rangle)$ meaning that customers from NY have all bought a product b and then a product f. Note that the sequences found by this approach do not contain several dimensions since the dimension time only concerns products. The product dimension is the only dimension that can be combined over time, meaning that it is not possible to have a rule indicating that when b is bought in $Boston$ then c is bought in NY.

[13] proposes to mine such *inter pattern* multidimensional sequences.

In [15], the authors consider sequential pattern mining in the framework of Web Usage Mining. Even if they consider three dimensions (pages, sessions, days), these dimensions are very particular since they belong to a single hierarchized dimension. Moreover, the sequences found describe correlations between objects over time by considering only one dimension, which corresponds to the web pages.

We can also quote work of [3], which proposes a first order temporal logic based approach for multidimensional sequential pattern mining. [8] also proposes a new method of generation of the multidimensional sequences embedded in a set of transactions.

To our best knowledge, there is not any approach proposing to take the hierarchies into account in the extraction of multidimensional sequential patterns. We thus propose to integrate the management of the hierarchies into M^2SP in order to allow a more complete extraction of knowledge, suitable in the OLAP framework.

2.3 Running Example

In order to illustrate the various concepts and definitions, we propose the running example. Table 1 describes the purchases of product carried out in various cities of the world. For the hierarchies, we choose two dimensions, the cities and the products, whose respective taxonomies are indicated in Figures 1 and 2.

Table 1: Running Example

D (Date)	B ($Block_{ID}$)	Pl (Place)	P (Product)
1	1	Germany	beer
1	1	Germany	pretzel
2	1	Germany	M2
3	1	Germany	chocolate
4	1	Germany	M1
1	2	France	soda
2	2	France	wine
2	2	France	pretzel
3	2	France	M2
1	3	UK	whisky
1	3	UK	pretzel
2	3	UK	M2
1	4	LA	chocolate
2	4	LA	M1
3	4	NY	whisky
4	4	NY	soda

Figure 1: Taxonomy over the $Place$ dimension

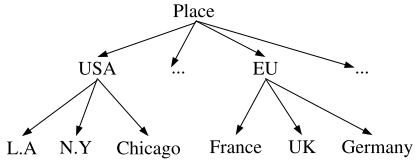
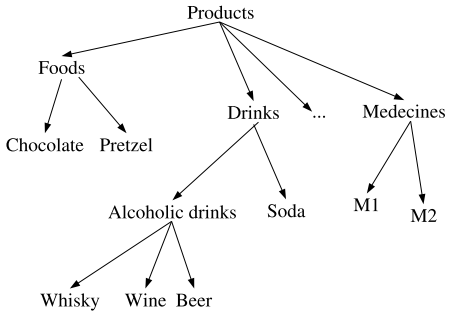


Figure 2: Taxonomy over the $Product$ dimension



3. CONTRIBUTIONS

In this section, we present our approach allowing the management of hierarchies in multidimensional sequential patterns. First, we define the concepts related to our approach. Then, we propose then the algorithms allowing the implementation of our approach.

3.1 Definitions

3.1.1 Dimension Set Partition

Let us consider a database DB where data are described with respect to n dimensions. We consider a 3-bin partitioning of the dimensions: the set of those dimensions that will be contained within the rules (*analysis dimensions*) is denoted by D_A; the set of those dimensions which the counting will be based on (*reference dimensions*) is denoted by D_R; and the set of those dimensions that are meant to introduce an order between events (*e.g.* $time$) is denoted by D_T. Each tuple $c = (d_1, \ldots, d_n)$ can thus be denoted by $c = (r, a, t)$ with r the restriction on D_R, a the restriction on D_A, t the restriction on D_T.

Given a table DB, the set of all tuples in DB having the same value r on D_R is said to be a *block* denoted by $\mathscr{B}_{DB,D_R}$ on the set of blocks from table DB. The concept of block is necessary to define the support of a multidimensional sequence. Its application in our running example is trivial since $|D_R| = 1$, the different blocks are described in the Figure 3.

Figure 3: Block Partition of DB (figure 1) according to $D_R = \{B\}$

Figure 4: block (1)

D	B	Pl	P
1	1	Germany	beer
1	1	Germany	pretzel
2	1	Germany	M2
3	1	Germany	chocolate
4	1	Germany	M1

Figure 5: block (2)

D	B	Pl	P
1	2	France	soda
2	2	France	wine
2	2	France	pretzel
3	2	France	M2

Figure 6: block (3)

D	B	Pl	P
1	3	UK	whisky
1	3	UK	pretzel
2	3	UK	M2

Figure 7: block (4)

D	B	Pl	P
1	4	LA	chocolate
2	4	LA	M1
3	4	NY	whisky
4	4	NY	soda

3.1.2 Taxonomies

In our multidimensional framework, we consider that there are hierarchical relations on each analysis dimension[1]. We consider that these hierarchical relations are materialized in the form of *taxonomy*. A taxonomy is a direct acyclic graph. The edges are *is-a* relation. The *Specialization* relation is then from root to leaves. Each analysis dimension thus has a taxonomy which makes it possible to represent the hierarchical relations between the elements of its domain.

Let $T_{D_A} = \{T_1, \ldots, T_m\}$ be the set of taxonomies associated to analysis dimensions where: **(i)** T_i is the taxonomy representing the hierarchical relations between the elements from the domain of the analysis dimension D_i; **(ii)** T_i is a direct acyclic graph; **(iii)** $\forall$ node $n_i \in T_i$, $label(n_i) \in Dom(D_i)$.

[1]This relation may be reduced to the tree of depth 1 where the root is labelled by * if no hierarchy is defined.

We write $\hat{x}$ an ancestor of x according to the associated taxonomy and $\check{x}$ one of its descendant. For instance, $drinks = \widehat{soda}$ means that $drinks$ is an ancestor of $soda$ according to the *Generalization/Specialization* relation. More precisely, $drinks$ is a more general instance than $soda$.

3.1.3 Hierarchies and Data

Each analysis dimension D_i from a transaction b of DB cannot be instanciated with a value d_i of which the node associated to the label d_i in the taxonomy T_i is a leaf. Formaly, $\forall d_i \in \pi_{D_i}(B)$,- $\forall$ node n_i *such that* $label(n_i) = d_i \nexists$nœud n' *such that* $n' = \check{n_i}$ (n_i leaf). For instance, the transaction database cannot contain the value $drinks$ since there are some more specific instances in the taxonomy $(soda, wine)$.

3.1.4 h-generalized Item, Itemset and Sequence

We now define the fundemental concepts of h-generalized item, itemset and sequence.

DEFINITION 1 (MULTIDIMENSIONAL H-GENERALIZED ITEM). *A multidimensional h-generalized item $e = (d_1, \ldots, d_m)$ is a tuple defined over the set of the m D_A dimensions such that $d_i \in \{label(T_i)\}$.*

Contrary to the transactions of DB, multidimensional h-generalized items can be defined with any value d_i which associated node in the taxonomy is not a leaf. For instance $(drinks, USA)$, $(soda, France)$ are some multidimensional h-generalized items.

Since multidimensional h-generalized items are instanciated on various levels of hierarchy, it is possible that two items are comparable, i.e. item is more *specific or general* than another. In order to no be heavy with the notations, we directly use the concept of ancestor on the item and the transaction without locating in the corresponding taxonomy.

DEFINITION 2 (HIERARCHICAL INCLUSION). *Let e and e' be two multidimensional h-generalized items, $e = (d_1, \ldots, d_m)$ and $e' = (d'_1, \ldots, d'_m)$, we say that:*

- *e is more general than e' ($e >_h e'$) if $\forall d_i, d_i = \hat{d'_i}$ or $d_i = d'_i$;*
- *e is more specific than e' ($e <_h e'$) if $\forall d_i, d_i = \check{d'_i}$ or $d_i = d'_i$;*
- *e and e' are incomparable if there is no relation between them ($e \not>_h e'$ and $e' \not>_h e$).*

For instance, we have:

- $(USA, drinks) >_h (USA, soda)$;
- $(France, wine) <_h (EU, Alcoholic\ drinks)$;
- $(France, wine)$ and $(USA, soda)$ are incomparable.

DEFINITION 3. *A transaction b supports an item e if $\Pi_{D_A}(b) <_h e$.*

As an example, the transaction $(1, 1, France, wine)$ supports the item $(EU, alcool)$.

DEFINITION 4 (MULTIDIMENSIONAL H-GENERALIZED ITEMSET). *A multidimensional h-generalized itemset $i = \{e_1, \ldots, e_k\}$ is a non-empty set of multidimensional h-generalized items where all items are incomparable.*

Two comparable items cannot be present in the same itemset since we adopt a ensemblist point of view. Moreover we prefer to represent the most precise possible information within an itemset. For instance, $\{(France, wine), (USA, soda)\}$ is a multidimensional h-generalized itemset whereas $\{(France, wine), (EU, Alcoholic\text{-}drinks)\}$ is not such an itemset because $(France, wine) <_h (EU, Alcoholic\ drinks)$.

DEFINITION 5 (MULTIDIMENSIONAL H-GENERALIZED SEQUENCE). *A multidimensional h-generalized sequence $s = \langle i_1, \ldots, i_j \rangle$ is a non-empty ordered list of multidimensional h-generalized itemsets.*

For instance, $\langle \{(France, wine), (USA, soda)\}, \{(Germany, beer)\} \rangle$ is a multidimensional h-generalized sequence. Multdimensional sequences can be included into another one:

DEFINITION 6 (SEQUENCE INCLUSION). *A multdimensional sequence $\varsigma = \langle a_1, \ldots, a_l \rangle$ is said to be a subsequence of $\varsigma' = \langle b_1, \ldots, b_{l'} \rangle$ if there exist integers $1 \leq j_1 \leq j_2 \leq \ldots \leq j_l \leq l'$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots, a_l \subseteq b_{j_l}$.*

The inclusion of the multidimensional sequences must respect the hierarchical inclusion of the multidimensional h-generalized items. As an example:

- The sequence $\langle \{(France, wine)\}, \{(Germany, beer)\} \rangle$ is a subsequence of $\langle \{(France, wine), (USA, soda)\}, \{(Germany, beer)\} \rangle$;
- The sequence $\langle \{(France, wine)\}, \{(Germany, beer)\} \rangle$ is a subsequence of $\langle \{(France, Alcoholic\ drinks), (USA\text{-}, drinks)\}, \{(EU, Alcoholic\ drinks)\} \rangle$;
- The sequence $\langle \{(EU, wine)\}, \{(Germany, beer)\} \rangle$ is not a subsequence of the sequence $\langle \{(France, wine), (USA, soda)\}, \{(Germany, beer)\} \rangle$ because $(EU, wine) \not\leq_h (France, wine)$, the hierarchical inclusion is not respected here.

3.1.5 Support

Computing the support of a multidimensional h-generalized sequence is equivalent to count the the number of blocks defined over the reference dimensions D_R which support the sequence. A block supports a multidimensional h-generalized sequence if it is possible to find a set of tuples which satisfies it. All the itemsets from the multidimensional h-generalized sequence must be found on various dates belonging to the domain of D_t such as the order of the itemsets respects the sequentiality.

DEFINITION 7. *A block supports a sequence $\langle i_1, \ldots, i_l \rangle$ if $\forall j = 1 \ldots l,\ \exists d_j \in Dom(D_t)$, for each item e from i_j, $\exists t = (r, e, d_j)$ where $t = (r, \check{e}, d_j) \in T$ w.r.t. $d_1 < d_2 < \ldots < d_l$.*

Thus, the *support* of a multidimensional h-generalized sequence is the number of those blocks defined over D_R which contain this sequence.

According to a user-defined-minimal-treshold , a *multidimensional h-generalized sequential pattern* is a sequence which support is greater than the minimal treshold.

EXAMPLE 1. *According to our running example database DB, let us consider $D_R = \{B_{id}\}$, $D_A = \{Place, Product\}$, $D_T = \{Date\}$, $minimal\ support = 2$, and $\varsigma = \langle\{(EU, Alcoholic\text{-}drinks), (EU, pretzel)\}\{(EU, M2)\}\rangle$. The sequence is frequent if at least two blocks of the partition of DB support the sequence.*

1. block $(\mathbf{1})$ *(Fig. 4). According to the taxonomies, Germany is more specific than EU and beer is an Alcoholic drinks. Thus, at the date 1, there is the first itemset $\{(EU, Alcool), (EU, pretzel)\}$ of ς. A date later (2), the last itemset $\{(EU, M2)\}$ is contained. The sequence ς is supported by this block.*

2. block $(\mathbf{2})$ *(Fig. 5). France is an instance of EU and wine is more specific than Alcoholic drinks. The sequence ς is supported by this block.*

3. block $(\mathbf{3})$ *(Fig. 6). UK is an instance of EU and whisky is an instance of Alcoholic drinks. This block then supports the sequence ς.*

4. block $(\mathbf{4})$ *(Fig. 7). This block does not support the sequence ς since the $place$ dimension does not contain any instance of EU.*

The support of ς is 3. The sequence is thus frequent.

3.2 The HYPE Algorithm Proposal

3.2.1 Overview

Before presenting the algorithms allowing the extraction of multidimensional h-generalized sequential patterns, we briefly detail the general behavior of our approach.

HYPE is divided into two phases. Firstly, the maximaly specific items are extracted. We think the maximaly specific items are an alternative to the huge of extracted knowledge. Indeed, they make it possible to *factorize* knowledge. The user can infer more general knowledge in a post-processing. Secondly, the multidimensional h-generalized sequences are mined in a further step. These sequences are generated and validated from the frequent maximaly items.

However, the fact of using maximaly specific items to generate the frequent sequences does not enable us to extract all knowledge embedded in the database. Indeed, some sequences whose first items are not maximaly specific cannot be mined. Some longer sequences cannot then be mined (blocks fastly support more general knowledge). However, this deficiency is relative because these non-mined sequences often describe too general knowledge which do not provide any interest for the user.

It is not necessary to prune the taxonomies in a preprocessing step. Indeed, this operation can easily be carried out during the multidimensional h-generalized sequential pattern mining process.

3.2.2 Generation of frequent items

Multidimensional h-generalized items are the basis of the multidimensional h-generalized sequential pattern mining. They are the sequences which length is 1. In a will of scalbility, items cannot be mined in only one scan. Indeed, considering the cartesian product of analysis dimension domains is not possible in aplications where the number of dimensions and the cardinality of their domains can be very important. If the number of analysis dimensions is m, then the number of generated items χ is exponential according to m:

$$2^m \leq \chi \leq \sum_{i=1}^{m} \binom{m}{i} i^k \ where\ k = \max|Dom(D_i)|$$

We thus consider that such an approach can compromise the scalability of the extraction. It is thus necessary to define a method which limits the number of candidate items and the number of database scans. In order to focus on items which probability to be frequent is non-null, we adopt a levelwise algorithm. Indeed a levelwise algorithm is used in order to build the frequent multidimensional h-generalized items. To this end, we consider a lattice which lower bound is the $(*, \ldots, *)$ multidimensional item[2]. This lattice is partially built from $(*, \ldots, *)$ up to the frequent items[3] containing as few $*$ as possible. At level i, i values are specified. Then items at level i are combined to build a set of candidates at level $i+1$. The process is iterated m times until obtaining the complete set of multidimensional h-generalized items. Two frequent items are combined to build a candidate if they are $\bowtie$-compatible. That is to say they share a sufficient number of valus over analysis dimensions (Definition 8). To be $\bowtie$-compatible, two multidimensional items defined over n dimensions should share $n-2$ values. For instance, $(a, *, c)$ and $(*, b, c)$ share $3-2=1$ value, they are then $\bowtie$-compatible. The items $(a, b, *)$ and $(a, b, *)$ are not compatible.

DEFINITION 8 ($\bowtie$-COMPATIBILITY). *Let $e_1 = (d_1, \ldots, d_n)$ and $e_2 = (d'_1, \ldots, d'_n)$ be two distinct multidimensional items where d_i and $d'_i \in dom(D_i) \cup \{*\}$. e_1 and e_2 are said to be $\bowtie$-compatible if $\exists \Delta = \{D_{i_1}, \ldots, D_{i_{n-2}}\} \subset \{D_1, \ldots, D_n\}$ such that for every $j \in [1, n-2]$, $d_{i_j} = d'_{i_j} \neq *$ with $d_{i_{n-1}} = *$ and $d'_{i_{n-1}} \neq *$ and $d_{i_n} \neq *$ and $d'_{i_n} = *$.*

The join operation is defined as follows:

DEFINITION 9 (JOIN). *Let $e_1 = (d_1, \ldots, d_n)$ and $e_2 = (d'_1, \ldots, d'_n)$ be two $\bowtie$-compatible multidimensional items. We define $e_1 \bowtie e_2 = (v_1, \ldots, v_n)$ where $v_i = d_i$ if $d_i = d'_i$, $v_i = d_i$ if $d'_i = *$ and $v_i = d'_i$ if $d_i = *$. Let E and E' be two sets of multidimensional items of size n, we define $E \bowtie E' = \{e \bowtie e' \ s.t.\ (e, e') \in E \times E' \wedge$ e and e' are $\bowtie$-compatible$\}$*

3.2.3 Generation of Frequent Sequences

The frequent items give all frequent sequences containing one itemset consisting of a single item. To mine the frequent multidimensional h-generalized sequences, we follow the Apriori-like paradigm. Indeed, the multidimensional framework keeps the antimonotony of the support (All subsets of a frequent set are frequent). Once 1-frequent items are mined, the candidate sequences of size k ($k \geq 2$)

[2] $*$ on dimension D_i can be seen as the root of the taxonomy T_i.

[3] By definition, an h-generalized item is instanciated over all its dimensions. By misnomer, we use the term of item to describe the frequent tuples which are instanciated in a levelwise method in order to mine multidimensional h-generalized items.

are generated and validated to keep the frequent items. To efficiently maintain the set of frequent sequence, we use a prefixed-tree-like structure as [10].

3.2.4 Counting the Support of a Sequence

The support counting is one of the main operation of datamining process.

The reference dimensions enables to identify all the blocks which may support a sequence ς. The enumeration of all the blocks is essential to compute the support of a sequence and thus define if the sequence is frequent. The algorithm 1 checks whether each block of DB supports the sequence by calling the function *supportBlock* (Algorithm 2). If the sequence is supported then its support is incremented. The algorithm then returns the relative support of the sequence.

The algorithm 2 checks if a sequence ς is supported by a block B. To achieve it, the algorithm combines *recursivity and anchoring operation*. The anchoring operation is used to reduce the space search. This algorithm attempts to find a tuple from the block that matches the first item of the first itemset of the sequence in order to *anchor* the sequence. This operation is repeated recursively until all itemsets from the sequence are found (return true) or until there is no way to go on further (return false). Several possible anchors may be tested if the first ones do not fit.

Algorithm 1: SupportCount: Compute the support of a sequence

Data: Sequence ς, database DB, reference dimension set D_R
Result: The support of the sequence ς
begin
  $Integer\ support \longleftarrow 0$;
  $Boolean\ supportedSeq$;
  $\mathscr{B}_{DB,D_R} \longleftarrow \{blocks\ identified\ over\ D_R\}$;
  foreach $B \in \mathscr{B}_{DB,D_R}$ **do**
    $supportedSeq \longleftarrow supportBlock(\varsigma, B)$;
    if *supportedSeq is true* **then**
      $support \longleftarrow support + 1$;
  return $\left(\frac{support}{|B_{DB,D_R}|}\right)$;
end

In order to study the complexity of these algorithms, we adopt the following notations: n_B is the number of tuples in B, $m = |D_A|$ is the number of analysis dimensions, P_{max} is the maximal depth of a taxonomy.

SupportBlock (algorithm 2) The block B is ordered w.r.t. D_T, the anchoring operation is realizable in $O(\log n_C)$. It is enough to carry out a dichotomic seeking to find all the tuples w.r.t. the date condition. Checking if a tuple supports an item take, in the worst case, $O(P_{max} \times m)$. We should compare the m dimensions of the item to tuple's ones. In the worst case, the complexity is $O(n_B \times P_{max} \times m \times \log n_B)$.

SupportCount (algorithm 1) The previous function is called for each the l blocks B_i from $\{B_{DB,D_R}\}$. Let $n_{max} = \max n_{B_i}$, the complexity in the worst case is then: $O(l) \times O(n_{max} \times P_{max} \times m \times \log n_{max}) = O(l \times n_{max} \times P_{max} \times m \times \log n_{max})$

3.3 HYPE Against M²SP

Algorithm 2: SupportBlock: Checking if a sequence is supported by a block

Data: Sequence ς, block B
Result: Boolean
begin
  /* *--initialization--* */
  $boolean\ foundItemSet \longleftarrow faux$
  $sequence \longleftarrow \varsigma$
  $itemset \longleftarrow sequence.first()$
  $item \longleftarrow itemset.first()$
  /* Condition to stop the recursivity */
  si $\varsigma = \emptyset$ **alors**
    retourner (vrai)
  /* Scanning the block */
  while $tuple \longleftarrow B.next \neq \emptyset$ **do**
    if *tuple supports item* **then**
      $followingItem \longleftarrow itemset.second()$
      si $followingItem = \emptyset$ **alors**
        $foundItemSet \longleftarrow true$
      /* Searching for all the items of the itemset */
      else
        /* Anchoring w.r.t. item (date) */
        $B' \longleftarrow \sigma_{date=cell.date}(B)$
        while $tuple' \longleftarrow B'.next() \neq \emptyset \wedge foundItemSet = faux$ **do**
          if *tuple′ supports followingItem* **then**
            $followingItem \longleftarrow itemset.next()$
            if $followingItem = \emptyset$ **then**
              $foundItemSet \longleftarrow vrai$
      if *foundItemSet is true* **then**
        /* Searching for the other itemsets */
        return $(SupportBlock(sequence.tail(), \sigma_{date>tuple.date}(B)))$
      else
        $itemset \longleftarrow sequence.first()$
        /* Reducing the search space */
        $C \longleftarrow \sigma_{date>cell.date}(B)$
  /* ς is not supported */
  return (false)
end

Managing hierarchies can be seen as a better way to manage the joker values, previously defined in [13]. Indeed, in M^{2}SP, the root of a taxonomy is the joker value * over the associated dimension. Then, if there is no possible value instanciation, no leaf label can be suitable. So, we directly go from the leaf to the root of the taxonomy (Figure 8). Thanks to HYPE, more accurate knowledge is mined. Indeed, taxonomies are an alternative when M^{2}SP is not able to instanciate a dimension. We do not directly go from leaf to root. We try to instanciate the dimension with the most specific ancestor of the leaf (9).

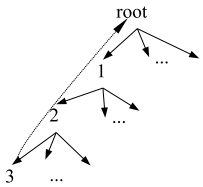


Figure 8: Joker value (*) management with M^{2}SP

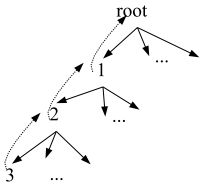


Figure 9: Hierarchy management with HYPE

EXAMPLE 2 (COMPARISON WITH M^{2}SP). *Let a user-defined threshold, the taking into account of the hierarchies (HYPE) makes it possible to mine knowledge which is not mined by M^{2}SP.*

M^{2}SP:

- $(*, chocolate)$, $(*, pretzel)$, $(*, M1)$, $(*, soda)$, $(*, M2)$, $(*, whisky)$
- $\langle\{(*, chocolate)\}\{(*, M1)\}\rangle$, $\langle\{(*, pretzel)\}\{(*, M2)\}\rangle$

HYPE:

- $(Place, chocolate)$,$(EU, pretzel)$, $(Place, M1)$, $(Place$,-$soda)$, $(EU, M2)$, $(Place, Whisky)$, $(EU, Alcoholic\,drinks)$,
- $\langle\{(Place, chocolate)\}\{(Place, M1)\}\rangle$ $\langle\{(EU, pretzel)\}\{(EU, M2)\}\rangle$ $\langle\{(EU, Alcoholic\,drinks)\}\{(EU, M2)\}\rangle$
- $\langle\{(EU, Alcoholic\,drinks), (EU, Pretzel)\}\{(EU, M2)\}\rangle$

Taking into account hierarchies makes it possible to mine more finer sequences than in M^{2}SP's approach.

4. EXPERIMENTS

In this section, we report experiments performed on synthetic data. These experiments aim at showing the interest of our approach, especially in the hierarchy management. The synthetic database contains $5,000$ tuples defined over 5 analysis dimensions. These first experiments compare the number of frequent mined sequences over the depth of the taxonomies (specialisation level) and the user

Figure 10: Number of frequent sequences over the depth of the taxonomies (minsup=0.3, $D_A = 5$, deg = 3)

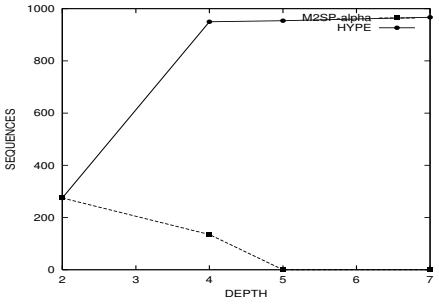


Figure 11: Number of frequent sequences over the depth of the taxonomies (minsup=0.4, $D_A = 5$, deg = 4)

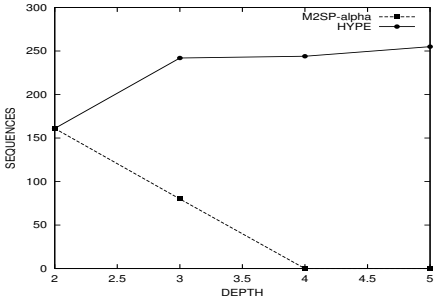


Figure 12: Number of frequent sequences over the minimal support ($D_A = 5$, deg = 3, higly correlated data)

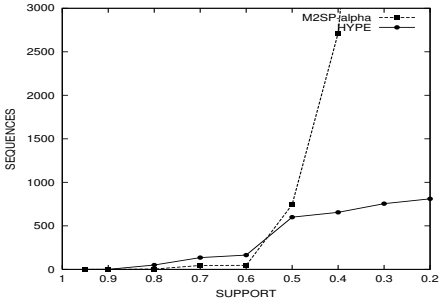
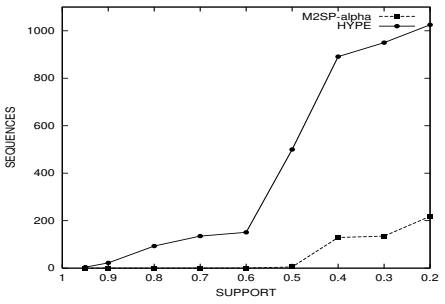


Figure 13: Number of frequent sequences over the minimal support (D_15, deg = 4, depth= 4)



defined threshold. We compared our results to M^2SP's results. in order to study the quality of the mined knowledge.

The figures 10 and 11 show the number of frequent mined sequences over the depth of the taxonomies. Growing the taxonomies generates an additional specialization level (drinks become alcoholic drinks or sodas). When the data becomes more specific, M^2SP mined less frequent sequences until it cannot mine any more knowledge. Taking into account hierarchies gives a robustness front of the specialization phenomena. Indeed, the sequences are mined among several levels of hierarchies.

The figure 12 shows the number of frequent mined sequences over the user-defined threshold in a highly correlated database (lower cardinality of analysis dimensions). As soon as the minimal support becomes too low, M^2SP extracts too many frequent sequences. Taking into account the hierarchies introduces a powerful ability of subsumption which prevents HYPE from mining too many sequences.

Furthermore, in lowly correlated database, the number of frequent mined sequences is simular to this one in highly correlated database whereas M^2SP mined a very low number of sequences. This highlights the interest of our approach front of the data quality (highly or lowly correlated database).

5. CONCLUSIONS

In this paper, we have defined the multidimensional h-generalized sequential patterns. We integrate the taking into account of the hierarchies thanks to taxonomies on analysis dimensions. It makes possible the building of multidimensional sequences defined over several level of hierarchies.

We have defined the different concepts (multidimensional h-generalized item, itemset and sequence) and the algorithms allowing the implementation of our approach. Experiments on synthetic data are reported and show the interest of HYPE. These experiments particulary show its ability to subsume knowledge and its strength in front of the data diversity (density, specialization, . . .).

This work offers several perspectives. The efficiency of the extraction could be enhanced by using condensed representation of mined knowledge (closed, free, non-derivable). The use of condensed representation can allow an additional pruning and thus enhance the extraction process. Other proprositions can be done about the hierarchy management. We can imagine modular management of the hierarchies where some dimensions would not have the same benhiavour than other ones in order to suit to the user's needs (prohibition to exceed the hierarchy level λ over the dimension ξ, . . .). The hierarchy management can allow us to define a novel automatic method to help user to navigate in the data cubes.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering (ICDE'95)*, pages 3–14, 1995.

[2] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *KDD*, pages 429–435. ACM, 2002.

[3] S. de Amo, D. A. Furtado, A. Giacometti, and D. Laurent. An apriori-based approach for first-order temporal pattern mining. In *XIX Simpósio Brasileiro de Bancos de Dados, 18-20 de Outubro, 2004,Brasília, Distrito Federal, Brasil, Anais/Proceedings*, pages 48–62. 2004.

[4] T. Dietterich and R. Michalski. Discovering patterns in sequences of events. *Artificial Intelligence*, 25(2):187–232, 1985.

[5] J. Han. Olap mining: Integration of olap with data mining. In *DS-7*, pages 3–20, 1997.

[6] J. Han and Y. Fu. Mining multiple-level association rules in large databases. *IEEE Trans. Knowl. Data Eng.*, 11(5):798–804, 1999.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pages 1–12, 2000.

[8] C.-H. Lee. An entropy-based approach for generating multi-dimensional sequential patterns. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *Knowledge Discovery in Databases: PKDD 2005, 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005, Proceedings*, volume 3721 of *Lecture notes in Computer Science*, pages 585–592. Springer, 2005.

[9] H. Mannila, H. Toivonen, and A. Verkamo. Discovering frequent episodes in sequences. In *Proc. of Int. Conf. on Knowledge Discovery and Data Mining*, pages 210–215, 1995.

[10] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proc. of PKDD*, volume 1510 of *LNCS*, pages 176–184, 1998.

[11] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(10), 2004.

[12] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *CIKM*, pages 81–88. ACM, 2001.

[13] M. Plantevit, Y. W. Choong, A. Laurent, D. Laurent, and M. Teisseire. M^2sp: Mining sequential patterns among several dimensions. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2005.

[14] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, 1996.

[15] C.-C. Yu and Y.-L. Chen. Mining sequential patterns from multidimensional sequence data. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):136–140, 2005.

[16] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.