Mining Multi-Dimensional and Multi-Level Sequential Patterns

Marc Plantevit, Anne Laurent, Dominique Laurent, Maguelonne Teisseire, Yeow Wei Choong

Mining Multi-Dimensional and Multi-Level Sequential Patterns

MARC PLANTEVIT
LIRMM - CNRS - Univ Montpellier 2, France
ANNE LAURENT
LIRMM - CNRS - Univ Montpellier 2, France
DOMINIQUE LAURENT
ETIS - CNRS - Univ Cergy Pontoise, France
MAGUELONNE TEISSEIRE
CEMAGREF Montpellier, France
and
YEOW WEI CHOONG
HELP University College - Kuala Lumpur, Malaysia

Multi-dimensional databases have been designed to provide decision makers with the necessary tools to help them understand their data. Compared to transactional data, this framework is particular as the datasets contain huge volumes of historized and aggregated data defined over a set of dimensions, which can be arranged through multiple levels of granularities. Many tools have been proposed to query the data and navigate through the levels of granularity. However, automatic tools are still missing to mine this type of data, in order to discover regular specific patterns. In this paper, we present a method for mining sequential patterns from multi-dimensional databases, taking at the same time advantage of the different dimensions and levels of granularity, which is original compared to existing work. The necessary definitions and algorithms are extended from regular sequential patterns to this particular case. Experiments are reported, showing the interest of this approach.

Categories and Subject Descriptors: H.2.8 [**Information Systems, Database Management**]: Database applications, data mining

General Terms: Algorithms, Design, Performance, Theory

Additional Key Words and Phrases: Sequential Patterns, Frequent Patterns, Multi-Dimensional Databases, Hierarchy, Multi-Level Patterns.

1. INTRODUCTION

Multi-dimensional databases have been studied for more than 10 years. They provide an easy-to-use interface for decision makers to navigate through their data. The modeling of such databases and the operations that can be applied are now well defined, and implemented by the main editors of database management systems (e.g. Oracle, Microsoft, IBM). However, only a few methods have been designed to automatically mine the relevant knowledge from these large amounts of historized data. In this framework, sequential patterns are suitable as they aim at discovering correlations between events through time. For instance, rules like *Many customers buy first a TV and a DVD player at the same time, and then a recorder* can be discovered. Scalable methods and algorithms to mine such rules have been proposed in the literature [Srikant and Agrawal 1996]. As for association rules,

the efficiency of the discovery is based on the *support* which indicates to which extend data from the database contains the patterns.

However, these methods cannot take advantage of the framework of multi-dimensional databases, because:

—they only consider one dimension to appear in the patterns, which is usually called the *product* dimension,

—they do not consider hierarchies.

Some studies claim to combine several dimensions [Pinto et al. 2001; de Amo et al. 2004; Yu and Chen 2005]. However, we argue here that they do not provide a complete framework for multi-dimensional sequential pattern mining. The way we consider multi-dimensionality is indeed generalized in the sense that patterns contain several dimensions combined over time [Plantevit et al. 2005]. Moreover, these dimensions are considered at *different levels* of granularity so as to automatically mine the most relevant rules. Note that mining rules at very high levels of granularity leads to trivial rules, whereas mining rules at very low levels of granularity is not always possible because the support value becomes too low.

In our approach, we aim at building rules like *When the sales of soft drinks are high in Europe, exports of Perrier in Paris and exports of soda in the US become high later on*. This rule not only combines two dimensions (Location and Product) but it also combines them over time and at different levels of granularity (as Perrier is considered as a kind of soft drink). As far as we know, no method has been proposed to mine such rules, except in our first proposal [Plantevit et al. 2006].

In order to mine the most relevant sequences, our approach is designed so as to take into account the most appropriate level of granularity, including partially instanciated tuples in sequences where the highest level is considered. More precisely, our algorithms are designed in order to mine frequent multi-dimensional sequences that may contain several levels of hierarchy, including the most general, usually referred to as the *ALL* value. However, in order to avoid mining non relevant patterns, only the most specific patterns (meant to be the most informative) are shown to the user.

The paper is organized as follows: Section 2 introduces a motivating example illustrating the goal of our work, and then, Section 3 presents existing work concerning multi-dimensional databases, multi-dimensional approaches and sequential pattern mining. Section 4 introduces basic definitions and Section 5 presents M^3SP algorithm for mining multi-dimensional and multi-level sequential patterns. Section 6 presents the results of the experiments performed on synthetic and real data. In Section 7, we conclude and present future research directions based on the present work.

2. MOTIVATING EXAMPLE

In this section, we present an example to illustrate our approach. This example will be used throughout the paper as a running example.

We consider a fact table T in which transactions issued by customers are stored. More precisely, we consider a set $\mathcal{D}$ containing six dimensions denoted by *D*, *Cat*, *Age*, *Loc*, *Prod* and *Qty*, where:

—*D* is the *date* of transactions (considering four dates, denoted by $1, 2, 3$ and 4),

—*Cat* is the customer *category* (considering two categories, denoted by $Educ$ and *Ret*, standing for educational and retired customers, respectively),

—*Age* is the *age* of customers (considering two discretized values, denoted by *Y* (young) and *O* (old)),

—*Loc* is the *location* where transactions have been issued (considering 6 locations, denoted by *NY* (New York), *LA* (Los Angeles), *SF* (San Francisco), *Paris*, *London* and *Berlin*),

—*Prod* is the *product* of the transactions (considering seven products, denoted by *chocolate*, *pretzel*, *whisky*, *wine*, *beer*, *soda*, $M1$ and $M2$), and

—*Qty* stands for the *quantity* of products in the transactions (considering nine quantities).

Table I shows the table T in which, for instance, the first tuple means that, at date 1, an educational young customer bought 50 units of beer in Berlin.

Let us now assume that we want to extract all multi-dimensional sequences that deal with products and the location where they have been bought, and that are frequent with respect to the groups of customers and their age. To this end, we consider three sets of dimensions as follows:

(1) the dimension *D*, representing the date,
(2) the two dimensions *Loc* and *Prod* that we call *analysis dimensions*, and which values appear in the frequent sequences,
(3) the two dimensions *Cat* and *Age*, which we call *reference dimensions*, according to which the support is computed.

Thus, tuples over analysis dimensions are those that appear in the items that constitute the sequential patterns to be mined. Moreover, the table is partitioned into blocks according to tuple values over reference dimensions and the support of a given multi-dimensional sequence is the ratio of the number of blocks supporting the sequence over the total number of blocks. Fig. 4 displays the corresponding blocks in our example.

In this framework, the multi-dimensional sequence $\langle\{(Berlin, beer), (Berlin, pretzel)\}, \{(London, wine)\}\rangle$ has support $\frac{1}{4}$, since the partition according to the reference dimensions contains four blocks, among which one supports the sequence. This is so because, in the first block shown in Fig. 4(1), $(Berlin, beer)$ and $(Berlin, pretzel)$ both appear at the same date (namely date 1), and $(London, wine)$ appears later on (namely at date 3).

The semantics of the *reference dimensions* is the following. In our example, *CG* and *A* are these reference dimensions. In this context, a frequent sequence $s = \langle\{(c1, p1), (c2, p1)\}, \{(c2, p2)\}\rangle$ means that in most cases, some customers sharing the same customer-group and age, first bought product $p1$ in city $c1$ and in city $c2$ at the same time, and *then* bought product $p2$ in city $c2$.

This kind of patterns is meant at discovering trends among customers according to their customer-group and age. However, it could be the case that the reference dimension is the identifier of a single customer. In this case, the patterns relate to the *same* customer.

It is important to note that, in the approach of the present paper, more general patterns can be mined, based on additional information provided by *hierarchies* over dimensions. The considered hierarchies allow for dealing with items at different levels of granularity, and this implies that further patterns can be mined.

To see this in the context of our running example, consider a support threshold of $\frac{1}{2}$ and two arbitrary locations λ and λ'. Then, no sequence of the form $\langle\{(\lambda, pretzel)\}, \{(\lambda', M2)\}\rangle$

is frequent. On the other hand, in the first and third blocks of Fig. 4, *pretzel* and $M2$ appear one after the other, according to the date of transactions, but in different locations. To take such a situation into account, we consider a specific constant, denoted by ALL_{Loc}, standing for any location and then, the sequence $\langle\{(ALL_{Loc}, pretzel)\},\{(ALL_{Loc}, M2)\}\rangle$ is frequent since its support is equal to $\frac{2}{4} = \frac{1}{2}$.

Similarly, $(Berlin, M2)$ appears only in block (2) of Fig. 4, and thus the sequence $\langle\{(Berlin, M2)\}\rangle$ is not frequent. But, if we consider the domain value EU as a *generalization* of the cities *Berlin*, *London* and *Paris*, then it is easy to see that the sequence $\langle\{(EU, M2)\}\rangle$ is frequent, since it can be considered as appearing in three of the blocks shown in Fig. 4, namely in blocks (1), (2) and (3).

Table I. Running example

D	Cat	Age	Loc	Prod	Qty
1	Educ	Y	Berlin	beer	50
1	Educ	Y	Berlin	pretzel	20
2	Educ	Y	London	M2	30
3	Educ	Y	London	wine	20
4	Educ	Y	NY	M1	40
1	Educ	O	LA	soda	20
2	Educ	O	Paris	wine	30
2	Educ	O	Berlin	pretzel	10
3	Educ	O	Paris	M2	20
1	Ret	Y	London	whisky	20
1	Ret	Y	London	pretzel	20
2	Ret	Y	Berlin	M2	30
1	Ret	O	LA	chocolate	50
2	Ret	O	Berlin	M1	20
3	Ret	O	NY	whisky	20
4	Ret	O	Paris	soda	30

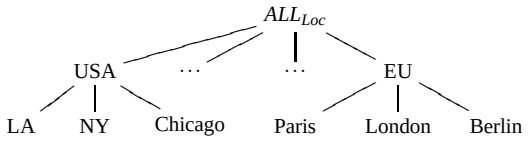


Fig. 1. Hierarchy over dimension *Loc*

3. RELATED WORK

In this section, we present the context of multi-dimensional databases and the existing work related to sequential patterns.

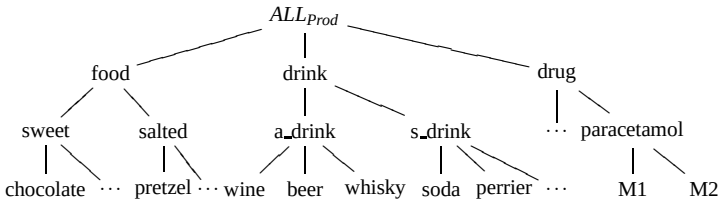


Fig. 2. Hierarchy over dimension *Prod*

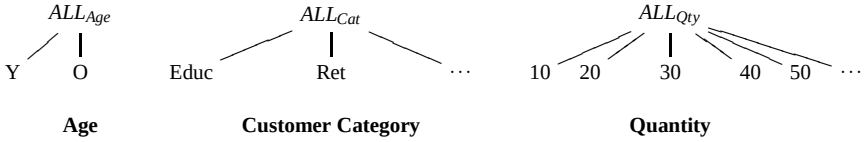


Fig. 3. Hierarchies over dimensions *Age*, *Cat* and *Qty*

(1) Block $(Educ, Y)$

D	Loc	Prod
1	Berlin	beer
1	Berlin	pretzel
2	London	M2
3	London	wine
4	NY	M1

(2) Block $(Educ, O)$

D	Loc	Prod
1	LA	soda
2	Paris	wine
2	Berlin	pretzel
3	Paris	M2

(3) Block (Ret, Y)

D	Loc	Prod
1	London	whisky
1	London	pretzel
2	Berlin	M2

(4) Block (Ret, O)

D	Loc	Prod
1	LA	chocolate
2	Berlin	M1
3	NY	whisky
4	Paris	soda

Fig. 4. Block partition of T (Table I) according to $\mathcal{D}_{\mathcal{R}} = \{Cat, Age\}$

3.1 Multi-Dimensional Databases

A multi-dimensional database Δ can be seen as a relational database defined over a particular schema, generally referred to as a *star schema* [Inmon 2003]. In general, a star schema consists of a distinguished table φ with schema F, called the *fact table*, and n other tables $\delta_1, \ldots, \delta_n$ with schemas $\Delta_1, \ldots, \Delta_n$, called the *dimension tables*, such that:

(1) If $K_1, \ldots, K_n$ are the (primary) keys of $\delta_1, \ldots, \delta_n$, respectively, then $K = K_1 \cup \ldots \cup K_n$ is the key of φ.

(2) For every $i = 1, \ldots, n$, $\pi_{K_i}(\varphi) \subseteq \pi_{K_i}(\delta_i)$ (thus each K_i is a foreign key in the fact table φ).

The attribute set $M = F \setminus K$ is called the *measure* of the star schema. Moreover, it is possible to consider *hierarchies* over attributes of the dimension tables, so as to query the database according to different levels of granularity.

When it comes to mine such a multi-dimensional database Δ, the relevant data are "extracted" from the database through a query, and data mining techniques are then applied to the answer of the query. In our approach, we follow this strategy in the following way: We assume that the table T to be mined is defined through a query on a given multi-dimensional database Δ. The attribute set over which T is defined is denoted by $\mathcal{D}$ and we assume that $\mathcal{D} = \{D_1, \ldots, D_n\}$. The set $\mathcal{D}$ may contain both dimension and measure attributes occurring in the schema of Δ, and we call these attributes *dimensions*. Moreover, dimensions in $\mathcal{D}$ are assumed to be such that:

(1) The hierarchies associated to dimensional attributes in the multi-dimensional database Δ can be considered so as to mine T at different levels of granularity.
(2) Among all dimensions in $\mathcal{D}$, one is associated with a *totally ordered* domain, referred to as the *time dimension*, and according to which sequences are constructed.
(3) A fixed subset of $\mathcal{D}$, whose elements are called *analysis dimensions*, along with the associated hierarchies allow the expression of sequential patterns. More precisely, in our approach, a sequential pattern is a sequence of sets of tuples defined over the analysis dimensions at different levels of granularity.
(4) Another subset of $\mathcal{D}$, whose elements are called *reference dimensions*, allows to partition T into blocks. Given a sequential pattern ς, the number of blocks that support ς over the total number of blocks is then defined as the *support* of ς.

We refer to the next section for formal definitions of the concepts just introduced, and we make the following remarks:

—Regarding the time dimension, it is possible to consider several dimensions, instead of a single dimension, provided that the cartesian product of the corresponding domain values be totally ordered. However, as this more general case introduces no further difficulty, but complicates notation, we restrict our appraoch to a single attribute in this respect.
—Hierarchies are explicitly used on analysis dimensions only. The level of granularity on the other dimensions is assumed to be *fixed* for a given mining task.
—For each analysis dimension D_i in $\mathcal{D}$, all values over D_i contained in T are assumed to be expressed at the *same* level of granularity, seen as the most specific one for the considered mining task. However, this level does not need to be the most specific one in the hierarchies defined in Δ.

3.2 Sequential Pattern Discovery

An early example of research in the discovering of patterns from sequences of events can be found in [Dietterich and Michalski 1985]. 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 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 presented in [Mannila et al. 1995]. 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 [Agrawal and Srikant 1995], 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 is a list of itemsets, and a user-specified minimum support threshold (minsup), the problem amounts to find all frequent subsequences, *i.e.*, the subsequences that appear a sufficient number of times. 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*.

Subsequently, many studies have introduced various methods for mining sequential patterns (mainly in time-related data), most of them being Apriori-like, *i.e.*, based on the Apriori property which states that any super-pattern of a nonfrequent pattern cannot be frequent. An example using this approach is the GSP algorithm [Srikant and Agrawal 1996], which has motivated a lot of research work, aiming at improving performance. The main approaches addressing this issue are SPADE [Zaki 2001], PrefixSpan [Pei et al. 2004], SPAM [Ayres et al. 2002], PSP [Masseglia et al. 1998], DISC [Chiu et al. 2004] and PAID [Yang et al. 2006].

3.3 Multi-Dimensional Sequential Patterns

As far as we know, three main propositions have dealt with several dimensions when building sequential patterns. We briefly recall these propositions below.

The approach of [Pinto et al. 2001] is the first work dealing with several dimensions in the framework of sequential patterns.

In this work, multi-dimensional sequential patterns are defined over a schema $A_1 \ldots A_m S$ where $A_1, \ldots, A_m$ are dimensions describing the data and S is the sequence of items purchased by the customers, ordered over time. A multi-dimensional sequential pattern is defined as a pair $((a_1, ..., a_m), s)$ where $a_i \in A_i \cup \{*\}$ and s is a sequence. In this case, $(a_1, ..., a_m)$ is said to be a multi-dimensional pattern. For instance, the authors consider the sequence $((*, NY, *), \langle b, f \rangle)$, meaning that customers from NY have all bought product b and then product f.

Sequential patterns are mined from such multi-dimensional databases either (i) by mining all frequent sequential patterns over the product dimension and then grouping them into multi-dimensional patterns, or (ii) by mining all frequent multi-dimensional patterns and then mining frequent product sequences over these patterns. Note that the sequences found by this approach do not contain several dimensions since the dimension time only concerns products. Dimension product 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. As our approach allows to mine such knowledge, it can be seen as a generalization of the work in [Pinto et al. 2001].

Several other proposals directly follow the seminal paper of [Pinto et al. 2001]. In [Rashad et al. 2007], the authors propose an algorithm called *MobilePrefixSpan* in order to discover patterns that describe the movements of mobile users. However, they only consider consecutive order in their framework. The work in [Stefanowski and Ziembinski 2005; Stefanowski 2007] shows the relevance of multi-dimensional sequential patterns for

Web Usage Mining. Moreover, the authors propose to use both numeric and symbolic data, with a specific handling of numeric data. According to this approach, a multi-dimensional sequence is supported by a multi-dimensional data sequence if they are closely similar. However, no algorithm is defined in this paper. In [Zhang et al. 2007], the authors propose the mining of multi-dimensional sequential patterns in distributed systems.

In [Yu and Chen 2005], the authors consider sequential pattern mining in the framework of Web Usage Mining. Even if three dimensions (namely, pages, sessions and days) are considered, these dimensions are very particular since they belong to a single hierarchized dimension. Thus, the sequences mined in this work describe correlations between objects over time by considering only one dimension, which corresponds to the web pages.

In [de Amo et al. 2004], the approach is based on first order temporal logic. This proposition is close to our approach, but more restricted since (i) groups used to compute the support are predefined, whereas we consider the fact that the user should be able to define them (see reference dimensions below), and (ii) several attributes cannot appear in the sequences. The authors claim that they aim at considering several dimensions but they have only shown one dimension for the sake of simplicity. However, the paper does not provide any complete proposition to extend this point to *real* multi-dimensional patterns, as we do in our approach.

3.4 Multi-Level Rules

The work in [Srikant and Agrawal 1996] introduces the hierarchy management in the extraction of association rules and sequential patterns. The authors suppose that the hierarchical relations between the items are represented by a set of *hierarchies*. They make it possible to extract association rules or sequential patterns according to several levels of hierarchy. Transactions are modified by adding, for every item, all ancestors in the associated hierarchy, and then, the frequent sequences are generated. However, this approach cannot be scalable in a multi-dimensional context, simply because adding the list of all ancestors for each item and each transaction is not efficient. Indeed, such a hierarchy management implies that the size of the database be multiplied by the maximum height of hierarchies to the number of analysis dimensions. We show in Section 6 (see Fig. 7) that such a hierarchy management is not tractable for more than three analysis dimensions.

The approach in [Han and Fu 1999] is quite different. The authors tackle the association rule extraction problem, in such a way that their approach can be adapted to sequential pattern extraction. Beginning at the highest level of the hierarchy, the rules on each level are extracted while lowering the support when going down in the hierarchy. The process is repeated until no rules can be extracted or until the lowest level of the hierarchy is reached. However, this method does not make it possible to extract rules containing items of different levels. For example *wine* and *drink* cannot appear together in such a rule. This approach thus deals with the extraction of *intra hierarchy level* association rules, and does not address the general problems of extracting sequences at *multiple levels of hierarchy*.

As can be seen from this section, the existing work, and especially the one described in [Pinto et al. 2001] is said to be *intra*-pattern, since sequences are mined within the framework of a single description (the so-called *pattern*). Moreover rules are only mined at the same level of granularity. On the other hand, in [Han and Fu 1999], the rules are said to be *intra*-level. In this paper, we propose to generalize these studies to *inter-level* and *inter-pattern* multi-dimensional sequences.

Compared to our previous work in [Plantevit et al. 2005], the main contribution of this paper is to take hierarchies into account, whereas in [Plantevit et al. 2005], only two levels have been considered, namely the most specific level (the values that appear in the table to be mined), and the most general level (denoted by * in [Plantevit et al. 2005] and referred to as *ALL* in the present paper). Compared to the preliminary version of this work in [Plantevit et al. 2006], the present paper offers a detailed and formal presentation of the approach (see Section 4 and Section 5), as well as computational improvements and further experiments on both synthetic and real datasets. We shall come back to this last issue at the end of Section 6.

4. BASIC DEFINITIONS

4.1 Background

Let $\mathcal{D} = \{D_1, \ldots D_n\}$ be a set of dimensions. Each dimension D_i is associated with a (possibly infinite) domain of values, denoted by $dom(D_i)$. For every dimension D_i, we assume that $dom(D_i)$ contains a specific value denoted by ALL_i.

In order to take into account the fact that items can be expressed according to different levels of granularity, we assume that each dimension D_i is associated with a *hierarchy*, denoted by H_i. Every hierarchy H_i is a tree whose nodes are elements of $dom(D_i)$ and whose root is ALL_i.

As usual, the edges of such a tree H_i can be seen as *is-a* relationships, and the *specialization* relation (respectively the *generalization* relation) corresponds to a top-down (respectively bottom-up) path in H_i, *i.e.*, a path connecting two nodes when scanning H_i from the root to the leaves (respectively from the leaves to the root).

In the case where no hierarchy is defined for a dimension D_i, we consider H_i as being the tree whose root is ALL_i and whose leaves are all the elements in $dom(D_i) \setminus \{ALL_i\}$. We recall in this respect that this paper is a generalization of our previous work [Plantevit et al. 2005], where, for every dimension D_i, no hierarchy is considered, and the symbol $*$ is used in place of ALL_i.

A fact table T over universe $\mathcal{D}$ is a finite set of tuples $t = (d_1, \ldots, d_n)$ such that, for every $i = 1, \ldots, n$, d_i is an element of $dom(D_i)$ that is a leaf of the associated hierarchy H_i. In other words, we assume that the table T to be mined contains only most specific values with respect to all hierarchies over dimensions.

We note that in our approach, the domains of attributes are not restricted to be discrete, even if each value occurring in the fact table T is treated as a nominal value. Moreover, it should be clear that discretizing continuous numerical attributes can be seen as defining a hierarchy on the corresponding domain.

Since we are interested in sequential patterns, we assume that $\mathcal{D}$ contains at least one dimension with a totally ordered domain, corresponding to the *time* dimension.

Moreover, given a fact table T over $\mathcal{D}$, for every $i = 1, \ldots, n$, we denote by $Dom_T(D_i)$ (or simply $Dom(D_i)$ if T is clear from the context) the *active domain* of D_i in T, *i.e.*, the set of all values of $dom(D_i)$ occurring in T along with their generalizations according to H_i. In the remainder of this paper, we consider only values in the active domains.

Given an element x in $Dom(D_i)$ and the associated hierarchy H_i, we introduce the following notation:

—$down(x)$ and $up(x)$ denote respectively the set of all *direct specializations* and the singleton containing the only *direct generalization* of x.

More precisiely, if x is not a leaf in H_i, then $down(x)$ is the set of all y in $Dom(D_i)$ such that H_i contains an edge from x to y; otherwise, $down(x)$ is set to be equal to the empty set. Similarly, if $x \neq ALL_i$, $up(x)$ is the set containing the only element y in $Dom(D_i)$ such that H_i contains an edge from y to x; otherwise, $up(x)$ is set to be equal to the empty set.

—We denote by $x^{\uparrow}$ (respectively $x^{\downarrow}$) the set containing x along with all generalizations (respectively specializations) of x with respect to H_i that belong to $Dom(D_i)$.

Clearly, for every $i = 1, \ldots, n$, we have $ALL_i^{\uparrow} = \{ALL_i\}$, $ALL_i^{\downarrow} = Dom(D_i)$, and, if x is a leaf of H_i then $ALL_i \in x^{\uparrow}$ and $x^{\downarrow} = \{x\}$. We also note that, for every x, we have $down(x) \subseteq x^{\downarrow}$, $up(x) \subseteq x^{\uparrow}$ and $\{x\} = x^{\uparrow} \cap x^{\downarrow}$.

EXAMPLE 1. *Referring back to our motivating example, the considered set of dimensions $\mathcal{D}$ is defined by $\mathcal{D} = \{D, Cat, Age, Loc, Prod, Qty\}$. Considering the fact table shown in Table I, we have for instance, $Dom(Prod) = \{beer, soda, wine, whisky, pretzel, chocolate, M1, M2\}$.*

Fig. 2 shows the hierarchy H_{Prod} associated to the dimension Prod. It can be seen from this figure that $down(drink) = \{s\_drink, a\_drink\}$ and $up(drink) = \{ALL_{Prod}\}$.

Moreover, Fig. 2 also shows that drink is a generalization of soda, that is $drink \in soda^{\uparrow}$ and $soda \in drink^{\downarrow}$. We also note that, as soda is a leaf of H_{Prod}, $down(soda) = \emptyset$ and $soda^{\downarrow} = \{soda\}$.

On the other hand, as no hierarchy is associated to the dimension D, the implicit associated hierarchy H_D is defined accordingly, that is, the root of H_D is ALL_D and all other values in $Dom(D)$ are leaves of H_D.

We end this section but pointing out that hierarchies on dimensions could be defined as *directed acyclic graphs* (DAGs). However, although this more general case can be considered in our approach, this would imply redundancies in the computation of frequent sequences. We shall come back to this point in Section 5.

4.2 Dimension Partitioning

For each table defined on the universe $\mathcal{D}$, we consider a partitioning of $\mathcal{D}$ into four sets:

—$\mathcal{D}_t$ contains a single dimension, called the *temporal* dimension,

—$\mathcal{D}_{\mathcal{A}}$ contains the *analysis* dimensions,

—$\mathcal{D}_{\mathcal{R}}$ contains the *reference* dimensions, and

—$\mathcal{D}_I$ contains the *ignored* dimensions.

Roughly speaking, in our approach, sequences are constructed according to the temporal dimension in $\mathcal{D}_t$ whose domain values are assumed to be *totally ordered*, and the tuples appearing in a sequence are defined over the analysis dimensions of $\mathcal{D}_{\mathcal{A}}$. Note that usual sequential patterns only consider one analysis dimension (generally corresponding to the products purchased or the web pages visited).

The set $\mathcal{D}_{\mathcal{R}}$ allows to identify the blocks of the database to be counted when computing supports. This is so because the support of a sequence is the proportion of those blocks that "support" the sequence. Note that, in the case of usual sequential patterns and of sequential patterns from [Pinto et al. 2001] and [de Amo et al. 2004], the set $\mathcal{D}_{\mathcal{R}}$ is reduced to one dimension, namely the *cid* dimension in [Pinto et al. 2001] and the *IdG* dimension in [de Amo et al. 2004].

The set $\mathcal{D}_I$ describes the ignored dimensions, *i.e.*, those dimensions that are used neither to define the date, nor the blocks, nor the patterns to be mined. Notice that $\mathcal{D}_I$ may be empty.

Based on such a partitioning of $\mathcal{D}$, and using a slight abuse of notation, each tuple $c = (d_1, \ldots, d_n)$ of T can be written as $c = (t, a, r, i)$ where t, a, r and i are the restrictions of c on $\mathcal{D}_t$, $\mathcal{D}_\mathcal{A}$, $\mathcal{D}_\mathcal{R}$ and $\mathcal{D}_I$, respectively.

Given a table T, the projection over $\mathcal{D}_t \cup \mathcal{D}_\mathcal{A}$ of the set of all tuples in T having the same restriction r over $\mathcal{D}_\mathcal{R}$ is called a *block*. More formally, given a tuple r in $\pi_{\mathcal{D}_\mathcal{R}}(T)$, the corresponding block, denoted by $B(T, r)$, or simply by $B(r)$ when T is understood, is defined by the relational expression $\pi_{\mathcal{D}_t \cup \mathcal{D}_\mathcal{A}}(\sigma_{\mathcal{D}_\mathcal{R} = r}(T))$. Moreover, we denote by $\mathcal{B}(T, \mathcal{D}_\mathcal{R})$, or simply by $\mathcal{B}(\mathcal{D}_\mathcal{R})$ when T is understood, the set of all blocks that can be constructed from T and $\mathcal{D}_\mathcal{R}$.

In our running example, we consider $\mathcal{D}_t = \{D\}$, $\mathcal{D}_\mathcal{A} = \{Loc, Prod\}$, $\mathcal{D}_\mathcal{R} = \{Cat, Age\}$, and $\mathcal{D}_I = \{Qty\}$. Fig. 4 shows the four blocks of $\mathcal{B}(\mathcal{D}_\mathcal{R})$ that can be constructed from table T of Table I and $\mathcal{D}_\mathcal{R}$ as specified just above.

4.3 Multi-Dimensional Items and Itemsets

Given a table T over a set $\mathcal{D}$ of n dimensions $D_1, \ldots, D_n$, over which hierarchies are assumed, we consider a fixed partitioning $\{\mathcal{D}_t, \mathcal{D}_\mathcal{A}, \mathcal{D}_\mathcal{R}, \mathcal{D}_I\}$ of $\mathcal{D}$ and we assume that $\mathcal{D}_\mathcal{A}$ is of cardinality m. In this setting, we define the fundamental concepts of item and itemset in the framework of multi-dimensional data.

DEFINITION 1 – MULTI-DIMENSIONAL ITEM. *A* multi-dimensional item *is a tuple* $a = (d_1, \ldots, d_m)$ *defined over* $\mathcal{D}_\mathcal{A}$, *that is, for every* $i = 1, \ldots, m$, $D_i \in \mathcal{D}_\mathcal{A}$ *and* $d_i \in Dom(D_i)$.

It is important to note that multi-dimensional items can be defined with values at any level of the hierarchies associated to analysis dimensions. For instance, in the context of our running example, $(drink, USA)$ and $(s\_drink, Paris)$ are multi-dimensional items.

Since multi-dimensional items are defined at different levels of hierarchies, it is possible to compare them using a specificity relation defined as follows.

DEFINITION 2 – ITEM SPECIFICITY RELATION. *For all multi-dimensional items* $a = (d_1, \ldots, d_m)$ *and* $a' = (d'_1, \ldots, d'_m)$, a' *is said to be* more specific *than* a, *denoted by* $a \preceq_I a'$, *if for every* $i = 1, \ldots, m$, $d'_i \in d_i^{\downarrow}$.

EXAMPLE 2. *Referring back to our running example, we have:*

—$(USA, drink) \preceq_I (USA, soda)$, *because* $USA \in USA^{\downarrow}$ *and* $soda \in drink^{\downarrow}$.
—$(EU, a\_drink) \preceq_I (Paris, wine)$, *because* $Paris \in EU^{\downarrow}$ *and* $wine \in a\_drink^{\downarrow}$.

However, $(Paris, wine)$ *and* $(USA, soda)$ *are not comparable according to* $\preceq_I$, *because Paris and USA are not comparable with respect to the corresponding hierarchy over the dimension Loc,* i.e., *neither* $Paris \in USA^{\uparrow}$ *nor* $Paris \in USA^{\downarrow}$ *holds.*

It is easy to see that the relation $\preceq_I$ is a partial ordering over the set of all multi-dimensional items. In other words, the relation $\preceq_I$ defined over $Dom(\mathcal{D}_\mathcal{A})$ is reflexive, anti-symmetric and transitive.

Moreover, in order to avoid redundancies in the extracted patterns, comparable items can not belong to the same itemset. Therefore, as stated in the following definition, two items

can belong to the same itemset only if they are not comparable according to the partial ordering $\preceq_I$.

DEFINITION 3 – ITEMSET. *An* itemset $s = \{a_1, \ldots, a_k\}$ *is a non-empty set of multi-dimensional items such that, for all distinct* i, j *in* $\{1, \ldots, k\}$, a_i *and* a_j *are not comparable with respect to* $\preceq_I$.

For instance, in the context of our running example, $\{(Paris, wine), (USA, a\_drink)\}$ is an itemset, whereas $\{(Paris, wine), (EU, a\_drink)\}$ is not. This is so because $(EU, a\_drink) \preceq_I (Paris, wine)$.

The basic notions of sequence and of support of a sequence are defined in the next section.

4.4 Multi-Dimensional Sequences and their Supports

DEFINITION 4 – SEQUENCE. *A* sequence $\varsigma = \langle s_1, \ldots, s_l \rangle$ *is a non-empty ordered list where, for every* $i = 1, \ldots, l$, s_i *is an itemset.*

A sequence of the form $\langle \{a\} \rangle$*, where* a *is a multi-dimensional item, is said to be an* atomic sequence.

For instance, in the context of our running example, $\langle \{(Paris, wine)\} \rangle$ is an atomic sequence, and $\langle \{(Paris, wine), (USA, a\_drink)\}, \{(EU, beer)\} \rangle$ is a non atomic sequence.

As will be seen later in the paper, atomic sequences play an essential role in our approach for mining frequent sequences while avoiding redundancies.

The following definition states that computing the support of a sequence amounts to count the number of blocks of $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$ that support the sequence.

DEFINITION 5 – SUPPORT OF A SEQUENCE. *A block* B *in* $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$ supports *the sequence* $\varsigma = \langle s_1, \ldots, s_l \rangle$ *if there exist* $t_1, \ldots, t_l$ *in* $Dom(D_t)$ *such that:*

(1) $t_1 < \ldots < t_l$

(2) *For every* $i = 1, \ldots, l$ *and every* α *in* s_i, B *contains a tuple* $c = (t_i, a)$ *such that* $\alpha \preceq_I a$.

The support *of* ς*, denoted by* $sup(\varsigma)$*, is the ratio of the number of blocks that support the sequence over the total number of blocks in* $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$.

Given a support threshold minsup, a sequence ς *is said to be* frequent *if its support is greater than or equal to minsup,* i.e., *if* $sup(\varsigma) \geq minsup$.

Based on Definition 5 above, an item a is said to be frequent if the sequence $\langle \{a\} \rangle$ is frequent. We recall from the introductory section that, in our approach, frequent sequences are mined based on the maximal atomic frequent sequences, referred to as *maf-sequences* and defined as follows.

DEFINITION 6 – MAF-SEQUENCES. *The atomic sequence* $\langle \{a\} \rangle$ *is said to be a* maximal atomic frequent sequence, *or an* maf-sequence *for short, if* $\langle \{a\} \rangle$ *is frequent and if for every* a' *such that* $a \prec_I a'$*, the sequence* $\langle \{a'\} \rangle$ *is not frequent.*

As will be seen in the next section, the frequent sequences to be mined in our approach are constructed based on maf-sequences. More precisely, maf-sequences provide all multi-dimensional items that occur in sequences to be mined.

We draw attention on the fact that multi-dimensional items occurring in maf-sequences are not comparable with respect to $\preceq_I$. Thus, according to Definition 3, any set containing such multi-dimensional items is an itemset.

The following example illustrates Definition 5 and Definition 6 in the context of our running example.

EXAMPLE 3. *For a support threshold* $minsup = \frac{1}{2}$, $\langle\{(EU, a\_drink)\}\rangle$ *is a frequent atomic sequence because, considering the blocks shown in Fig. 4, we have the following:*

(*1*) Block $B(Educ, Y)$ *(See Fig. 4(1)). According to the hierarchies, we have* $Berlin \in EU^{\downarrow}$ *and* $beer \in a\_drink^{\downarrow}$. *Thus,* $(EU, a\_drink) \preceq_I (Berlin, beer)$. *As* $(Berlin, beer)$ *is in* $B(Educ, Y)$, *this block supports* $\langle\{(EU, a\_drink)\}\rangle$. *Notice that a similar reasoning holds when considering the tuple* $(London, wine)$ *in this block.*

(*2*) Block $B(Educ, O)$ *(See Fig. 4(2)). As Paris is in* $EU^{\downarrow}$ *and wine is in* $a\_drink^{\downarrow}$, *for similar reasons as above, this block supports* $\langle\{(EU, a\_drink)\}\rangle$.

(*3*) Block $B(Ret, Y)$ *(See Fig. 4(3)). The same reasoning as in the previous two cases holds for this block, because London is in* $EU^{\downarrow}$ *and whisky is in* $a\_drink^{\downarrow}$. *Thus, this block also supports* $\langle\{(EU, a\_drink)\}\rangle$.

(*4*) Block $B(Ret, O)$ *(See Fig. 4(4)). This block does not support* $\langle\{(EU, a\_drink)\}\rangle$ *because it contains no multi-dimensional item more specific than* (EU, a_drink).

Since $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$ *contains* 4 *blocks, the support of* $\langle\{(EU, a\_drink)\}\rangle$ *is* $\frac{3}{4}$, *and thus, the sequence is frequent. Moreover, all multi-dimensional items a such that* $(EU, a\_drink) \prec_I a$, i.e., *all multi-dimensional items* $a = (d_1, d_2)$ *other than* (EU, a_drink) *and such that* $d_1 \in \{EU, Berlin, Paris, London\}$ *and* $d_2 \in \{a\_drink, beer, wine, whisky\}$, *lead to non frequent atomic sequences since these sequences are supported by at most one block. As a consequence, according to Definition 6,* $\langle\{(EU, a\_drink)\rangle$ *is an maf-sequence.*

It is also important to note that, although the table T contains four specializations of (EU, a_drink), *only three of them are counted in the computation of the support of* $\langle\{(EU, a\_drink)\}\rangle$. *This is so because* $(Berlin, beer)$ *and* $(London, wine)$ *are two specializations of* (EU, a_drink) *that belong to the same block,* i.e., *block* $B(Educ, Y)$, *which, according to Definition 5, is counted only* once *when computing* $sup(\langle\{(EU, a\_drink)\}\rangle)$.

Let us now consider the sequence $\varsigma = \langle\{(EU, a\_drink), (EU, pretzel)\}, \{(EU, M2)\}\rangle$ *and the same support threshold* $minsup = \frac{1}{2}$ *as above. It can be seen as above that* $\langle\{(EU, pretzel)\}\rangle$ *and* $\langle\{(EU, M2)\}\rangle$ *are also maf-sequences. Moreover, considering again successively the blocks shown in Fig. 4, we have:*

(*1*) Block $B(Educ, Y)$ *(See Fig. 4(1)). We have* $Berlin \in EU^{\downarrow}$, $London \in EU^{\downarrow}$ *and* $beer \in a\_drink^{\downarrow}$. *Thus,* $(EU, a\_drink) \preceq_I (Berlin, beer)$, $(EU, pretzel) \preceq_I (Berlin, pretzel)$ *and* $(EU, M2) \preceq_I (London, M2)$. *As the block* $B(Educ, Y)$ *contains the two tuples* $(1, Berlin, beer)$ *and* $(1, Berlin, pretzel)$ *along with the tuple* $(2, London, M2)$, *the sequence* ς *is supported by this block.*

(*2*) Block $B(Educ, O)$ *(See Fig. 4(2)). As Paris is in* $EU^{\downarrow}$ *and wine is in* $a\_drink^{\downarrow}$, *for similar reasons as above, the sequence* ς *is supported by this block.*

(*3*) Block $B(Ret, Y)$ *(See Fig. 4(3)). The same as in the previous two cases holds for this block, because London is in* $EU^{\downarrow}$ *and whisky is in* $a\_drink^{\downarrow}$. *Thus,* $B(Ret, Y)$ *also supports the sequence* ς.

(*4*) Block $B(Ret, O)$ *(See Fig. 4(4)). This block does not support the sequence* ς.

Since $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$ *contains* 4 *blocks, the support of* ς *is* $\frac{3}{4}$, *and thus, the sequence is frequent.*

In order to define sequence specialization in our approach, we first need to define when an itemset is more specific than another itemset.

DEFINITION 7 – ITEMSET SPECIFICITY RELATION. *Let s and s' be two itemsets. s' is said to be* more specific than *s, denoted by $s \preceq_{IS} s'$, if for every a in s, there exists a' in s' such that $a \preceq_I a'$.*

For instance, in the context of our running example, the itemsets $\{(Paris, wine)\}$ and $\{(Paris, wine), (USA, soda)\}$ are two specializations of the itemset $\{(EU, wine)\}$, *i.e.,* we have $\{(EU, wine)\} \preceq_{IS} \{(Paris, wine)\}$ and $\{(EU, wine)\} \preceq_{IS} \{(Paris, wine), (USA, soda)\}$. This is so because $(EU, wine) \preceq_I (Paris, wine)$.

We note that $\preceq_{IS}$ generalizes set inclusion, in the sense that for all itemsets s and s' such that $s' \subseteq s$, we have $s \preceq_{IS} s'$.

Moreover, the partial ordering $\preceq_{IS}$ defined above can be extended to sequences, so as to define sequence specialization as follows.

DEFINITION 8 – SEQUENCE SPECIFICITY RELATION. *Let $\varsigma = \langle s_1, \ldots, s_l \rangle$ and $\varsigma' = \langle s'_1, \ldots, s'_{l'} \rangle$ be sequences. ς' is said to be* more specific than *ς, denoted by $\varsigma \preceq_S \varsigma'$, if there exist integers $1 \leq i_1 < i_2 < \ldots < i_l \leq l'$ such that $s_1 \preceq_{IS} s'_{i_1}, s_2 \preceq_{IS} s'_{i_2}, \ldots, s_l \preceq_{IS} s'_{i_l}$.*

EXAMPLE 4. *In the context of our running example, the following are examples of sequence specializations:*

—$\langle \{(EU, wine)\} \rangle \preceq_S \langle \{(Paris, wine)\} \rangle$,
since, as seen earlier, $\{(EU, wine)\} \preceq_{IS} \{(Paris, wine)\}$.

—$\langle \{(EU, wine)\}, \{(EU, beer)\} \rangle \preceq_S \langle \{(Paris, wine), (USA, soda)\}, \{(Berlin, beer)\} \rangle$,
since we have $\{(EU, wine)\} \preceq_{IS} \{(Paris, wine), (USA, soda)\}$ *and* $\{(EU, beer)\} \preceq_{IS} \{(Berlin, beer)\}$.

—$\langle \{(Paris, a\_drink)\} \rangle \preceq_S \langle \{(Paris, wine), (USA, drink)\}, \{(Berlin, beer)\} \rangle$,
since $\{(Paris, a\_drink)\} \preceq_{IS} \{(Paris, wine), (USA, drink)\}$.

However, $\varsigma' = \langle \{(Paris, wine), (USA, soda)\}, \{(Berlin, beer)\} \rangle$ *is* not *more specific than* $\varsigma = \langle \{(USA, wine)\}, \{(Berlin, beer)\} \rangle$. *This is so because* $(USA, wine) \not\preceq_I (Paris, wine)$, $(USA, wine) \not\preceq_I (Paris, soda)$ *and* $(USA, wine) \not\preceq_I (Berlin, beer)$*; thus* $\{(USA, wine)\}$ *cannot be compared with any itemset in* ς'.

5. M^3SP: ALGORITHMS FOR *M*INING *M*ULTI-DIMENSIONAL AND *M*ULTI-LEVEL *S*EQUENTIAL *P*ATTERNS

In this section, we give fundamental properties of frequent multi-dimensional and multi-level sequential patterns and then, we detail the algorithms for their extraction. These algorithms are based on the following two-step process:

(1) *Step 1:* Maf-sequences are mined. To this end, we follow a strategy inspired from the BUC algorithm [Beyer and Ramakrishnan 1999].

(2) *Step 2:* All frequent sequences that can be built up based on the frequent sequences found in Step 1 are mined. To this end, we use the SPADE algorithm [Zaki 2001].

The correctness of each of these two steps is based on the following basic properties.

5.1 Basic Properties

In this section, we show the following:

(1) The support of sequences as defined in Definition 5 is anti-monotonic with respect to the partial ordering $\preceq_S$ (see Proposition 1 and Proposition 2). Such a monotonicity property is indeed required in Step 1 and Step 2 above.
(2) The partially ordered set $(Dom(\mathcal{D}_\mathcal{A}), \preceq_I)$ is shown to be a lattice, the elements of which can be generated in a non-redundant manner (see Proposition 4). With such properties at hand, the processing of Step 1 above can be achieved efficiently.

First, considering atomic sequences, we have the following proposition.

PROPOSITION 1. *For all multi-dimensional items a and a', if $a \preceq_I a'$ then $sup(\langle\{a'\}\rangle) \leq sup(\langle\{a\}\rangle)$.*

PROOF: Let a and a' be such that $a \preceq_I a'$ and let B be a block of T that supports $\langle\{a'\}\rangle$. Then, by Definition 5, B contains at least one tuple $t = (d, \alpha)$ where $d \in Dom(D_t)$, $\alpha \in Dom(\mathcal{D}_\mathcal{A})$ and $a \preceq_I \alpha$. Therefore, we have $a \preceq_I a' \preceq_I \alpha$ and thus, B supports $\langle\{a\}\rangle$. Hence $sup(\langle\{a'\}\rangle) \leq sup(\langle\{a\}\rangle)$, which completes the proof. □

Now, generalizing Proposition 1 above, the following proposition states that the support measure for sequences is anti-monotonic with respect to $\preceq_S$.

PROPOSITION 2. *For all sequences ς and ς', if $\varsigma \preceq_S \varsigma'$ then $sup(\varsigma') \leq sup(\varsigma)$.*

PROOF: Given two sequences ς and ς' such that $\varsigma \preceq_S \varsigma'$, based on Definition 8, and using similar arguments as in the previous proof, it is easy to see that every block B that supports ς' also supports ς. Therefore, we have $sup(\varsigma') \leq sup(\varsigma)$, and the proof is complete. □

We now turn to the properties required to see how to efficiently implement Step 1 above, *i.e.,* how to mine maf-sequences.

We first note in this respect that the partially ordered set $(Dom(\mathcal{D}_\mathcal{A}), \preceq_I)$ can be given a lattice structure provided a greatest element with respect to $\preceq_I$ be considered.

To see this formally, we consider an additional item, denoted by $\top_\mathcal{A}$, which is assumed to be more specific than any other multi-dimensional item in $Dom(\mathcal{D}_\mathcal{A})$, that is, for every a in $Dom(\mathcal{D}_\mathcal{A})$, we have $a \preceq_I \top_\mathcal{A}$.

In this setting, given two multi-dimensional items $a = (d_1, \ldots, d_m)$ and $a' = (d'_1, \ldots, d'_m)$, the least upper bound and the greatest lower bound of a and a', denoted by $lub(a, a')$ and $glb(a, a')$ respectively, are defined as follows:

—$lub(a, a') = (u_1, \ldots, u_m)$ where, for each $i = 1, \ldots m$, u_i is the most specific value in $d_i^{\uparrow} \cap (d'_i)^{\uparrow}$.
—$glb(a, a')$ is defined as follows:
If there exists i in $\{1, \ldots, m\}$ such that $d_i^{\downarrow} \cap (d'_i)^{\downarrow} = \emptyset$ then $glb(a, a') = \top_\mathcal{A}$
Otherwise, $glb(a, a') = (g_1, \ldots, g_m)$ where, for each $i = 1, \ldots, m$, g_i is the most general value in $d_i^{\downarrow} \cap (d'_i)^{\downarrow}$.

It is easy to see that for all a and a' in $Dom(\mathcal{D}_\mathcal{A})$, $lub(a, a')$ and $glb(a, a')$ are well defined, showing that $(Dom(\mathcal{D}_\mathcal{A}) \cup \{\top_\mathcal{A}\}, \preceq_I)$ is a lattice.

In our algorithms, we ignore the additional multi-dimensional item $\top_\mathcal{A}$ (as it is of no practical interest), and Step 1 mentioned above is achieved through a depth-first traversal of

the set $Dom(\mathcal{D}_{\mathcal{A}})$, starting from the most general multi-dimensional item $(ALL_1, \ldots ALL_m)$ that we denote by $ALL_{\mathcal{A}}$ in the remainder of the paper.

This algorithm follows the same strategy as BUC ([Beyer and Ramakrishnan 1999]) that we adapt to our particular context. Namely, for each atomic sequence $\langle\{a\}\rangle$ found to be frequent, we generate all the direct successors of a in $(Dom(\mathcal{D}_{\mathcal{A}}), \preceq_I)$ and, for each of them, we compute its support against the *subset* of the dataset consisting of those tuples that are more specific than a. In this way, as done in BUC algorithm, the more items are specialized, the smallest the dataset. We refer to Section 6 for a discussion on the performance of the corresponding algorithm.

In order to implement the corresponding algorithm, we need an effective and non redundant mean to characterize the direct successors in $(Dom(\mathcal{D}_{\mathcal{A}}), \preceq_I)$ of a given multi-dimensional item a.

To this end, we denote by $succ(a)$ the set of all direct successors of a given multi-dimensional item a. More formally, given two multi-dimensional items a and a', a' is in $succ(a)$ if:

—$a \prec_I a'$, and
—for every $\alpha \in Dom(\mathcal{D}_{\mathcal{A}})$, if $a \preceq_I \alpha \preceq_I a'$ then either $\alpha = a$ or $\alpha = a'$.

PROPOSITION 3. *For every $a = (d_1, \ldots, d_m)$ in $Dom(\mathcal{D}_{\mathcal{A}})$:*

$$succ(a) = \{(d'_1, \ldots, d'_m) \mid (\exists i \in \{1, \ldots, m\})(d'_i \in down(d_i) \wedge (\forall j \neq i)(d'_j = d_j))\}.$$

PROOF: Let $a' = (d'_1, \ldots, d'_m)$ be such that $(\exists i \in \{1, \ldots, m\})\ (d'_i \in down(d_i) \wedge (\forall j \neq i)(d'_j = d_j))$. Then, clearly, we have $a \prec_I a'$ and for every $\alpha = (\delta_1, \ldots, \delta_m)$ such that $a \preceq_I \alpha \preceq_I a'$, we have $\delta_i \in d_i^{\downarrow} \cap (d'_i)^{\uparrow}$ and for every $j \neq i$, $\delta_j = d_i = d'_j$. As $d'_i \in down(d_i)$, we have $d_i^{\downarrow} \cap (d'_i)^{\uparrow} = \{d_i, d'_i\}$. Thus, either $\alpha = a$ or $\alpha = a'$, which shows that $a' \in succ(a)$.

Conversely, let a' be in $succ(a)$. Then, we have $a \prec_I a'$, entailing that for every $i = 1, \ldots, m$, $d'_i \in d_i^{\downarrow}$. Assuming that $(\exists i \in \{1, \ldots, m\})(d'_i \in down(d_i) \wedge (\forall j \neq i)(d'_j = d_j))$ is not satisfied, means that for every $i \in \{1, \ldots, m\}$, we have (i) $d'_i \notin down(d_i)$ or (ii) there exists $j \neq i$ such that $d'_j \neq d_j$.

(i) In this case, there exists δ_i in $Dom(D_i)$ different than d_i and d'_i such that $\delta_i \in d_i^{\downarrow}$ and $d'_i \in \delta_i^{\downarrow}$. For $\alpha = (d_1, \ldots, d_{i-1}, \delta_i, d_{i+1}, \ldots, d_m)$, we have $a \prec_I \alpha \prec a'$, which is a contradiction with the fact that $a' \in succ(a)$.

(ii) In this case, d'_j is in $d_j^{\downarrow}$ and we can assume that d'_i is in $down(d_i)$. For $\alpha = (d_1, \ldots, d_{i-1}, d'_i, d_{i+1}, \ldots, d_m)$, we have $a \prec_I \alpha \prec_I a'$, which is again a contradiction with the fact that $a' \in succ(a)$. Thus, the proof is complete. □

EXAMPLE 5. *In the context of our running example, consider the multi-dimensional items $a = (EU, beer)$ and $a' = (Paris, a\_drink)$. According to Proposition 3 above, we have:*

—$succ(a) = \{(Paris, beer), (Berlin, beer), (London, beer)\}$,
because $down(EU) = \{Paris, Berlin, London\}$ and $down(beer) = \emptyset$,
—$succ(a') = \{(Paris, beer), (Paris, wine), (Paris, whisky)\}$,
because $down(Paris) = \emptyset$ and $down(a\_drink) = \{beer, wine, whisky\}$.

We note that, in this case, $succ(a) \cap succ(a') = \{(Paris, beer)\}$, meaning that if the set $succ(a) \cup succ(a')$ is to be computed based on a and a', $(Paris, beer)$ is generated twice.

As shown in the previous example, one important point in the computation of Step 1 is to make sure that every sequence $\langle\{a\}\rangle$ is considered at most once. In other words, we have to design a way to generate every multi-dimensional item a only once in our algorithm.

To this end, we assume that the dimensions in $\mathcal{D}_\mathcal{A}$ are ordered according to a fixed total ordering; let $D_1 < \ldots < D_m$ be this ordering. Given a multi-dimensional item $a = (d_1, \ldots, d_m)$, we denote by $\rho(a)$ the integer defined as follows:

—If $a = ALL_\mathcal{A}$, then $\rho(a) = 0$

—Otherwise, $\rho(a)$ is the integer in $\{1, \ldots, m\}$ such that $d_{\rho(a)} \neq ALL_{\rho(a)}$ and for every $j > \rho(a)$, $a_j = ALL_j$.

Then the *generated multi-dimensional items* from a given multi-dimensional item a are defined as follows.

DEFINITION 9 – GENERATED MULTI-DIMENSIONAL ITEMS. *Let $a = (d_1, \ldots, d_m)$ be a multi-dimensional item. The set of multi-dimensional items* generated from *a, denoted by $gen(a)$, is defined by:*

—*If $\rho(a) = 0$ then $gen(a) = succ(a)$*

—*Otherwise,*
$gen(a) = \{(d'_1, \ldots, d'_m) \mid (\exists i \in \{\rho(a), \ldots, m\})((d'_i \in down(d_i)) \wedge (\forall j \neq i)(d'_j = d_j))\}$.

As a consequence of Definition 9, it is easy to see that if a is such that $\rho(a) = 1$ then we have $gen(a) = succ(a)$. The following example illustrates Definition 9 in the context of our running example.

EXAMPLE 6. *Referring back to Example 5, we assume that the dimensions Loc and Prod are such that Loc $<$ Prod. For $a = (EU, beer)$ and $a' = (Paris, a\_drink)$, we have $\rho(a) = \rho(a') = 2$. Thus, according to Definition 9 above, we obtain the following:*

—*$gen(a) = \emptyset$, because $down(beer) = \emptyset$, and*

—*$gen(a') = \{(Paris, beer), (Paris, wine), (Paris, whisky)\}$,*
because $down(a\_drink) = \{beer, wine, whisky\}$.

Therefore, when considering generated multi-dimensional items instead of successors, the multi-dimensional item $(Paris, beer)$ is considered only once.

On the other hand, for $a = (EU, ALL_{Prod})$, we have $\rho(a) = 1$, and thus, the set $gen(a)$ is equal to $succ(a)$.

In order to show an exhaustive computation of generated multi-dimensional items, for the sake of simplification, let us consider the following simplified hierarchies taken from Fig. 1 and Fig. 2, respectively.

$$ALL_{Loc} \rightarrow EU \rightarrow Paris \qquad ALL_{Prod} \rightarrow drink \rightarrow a\_drink \rightarrow beer$$

In this case, the generated multi-dimensional items are shown in Fig. 5.

The following proposition states that, using generated multi-dimensional items as defined above, all multi-dimensional items but $ALL_\mathcal{A}$ are generated in a non redundant manner.

PROPOSITION 4. *(1)* $\bigcup_{a \in Dom(\mathcal{D}_\mathcal{A})} gen(a) = Dom(\mathcal{D}_\mathcal{A}) \setminus \{ALL_\mathcal{A}\}$.
(2) For all a and a' in $Dom(\mathcal{D}_\mathcal{A})$, if $a \neq a'$ then we have $gen(a) \cap gen(a') = \emptyset$.

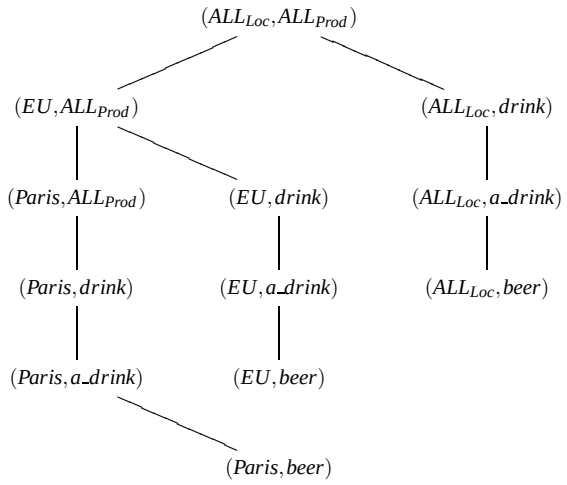


Fig. 5. Tree of generated multi-dimensional items

PROOF: (1) It is easy to see from Proposition 3 and Definition 9 that for every a in $Dom(\mathcal{D}_{\mathcal{A}})$, $gen(a) \subseteq succ(a)$. Thus, we have $\bigcup_{a \in Dom(\mathcal{D}_{\mathcal{A}})} gen(a) \subseteq Dom(\mathcal{D}_{\mathcal{A}}) \setminus \{ALL_{\mathcal{A}}\}$, because $ALL_{\mathcal{A}}$ cannot belong to any set $gen(a)$.

Conversely, let $a = (d_1, \ldots, d_m) \in Dom(\mathcal{D}_{\mathcal{A}}) \setminus \{ALL_{\mathcal{A}}\}$. Thus we have $\rho(a) > 0$ and $d_{\rho(a)} \neq ALL_{\rho(a)}$, which implies that $up(d_{\rho(a)}) \neq \emptyset$. Let $a' = (d'_1, \ldots, d'_m)$ be such that $d'_{\rho(a)} = up(d_{\rho(a)})$ and, for every $i \neq \rho(a)$, $d'_i = d_i$. Then, it is easy to see from Definition 9 that $a \in gen(a')$.
(2) Let $a = (d_1, \ldots, d_m)$ and $a' = (d'_1, \ldots, d'_m)$ be such that $a \neq a'$ and $gen(a) \cap gen(a') \neq \emptyset$ and let $\alpha = (\delta_1 \ldots, \delta_m)$ be in $gen(a) \cap gen(a')$. By Definition 9 and item (1) above, we have $\rho(a) > 0$ and $\rho(a') > 0$, and furthermore, there exist i in $\{\rho(a), \ldots, m\}$ and i' in $\{\rho(a'), \ldots, m\}$ such that:
– $\delta_i \in down(d_i)$ and for every $j \neq i$, $\delta_j = d_j$
– $\delta_{i'} \in down(d'_{i'})$ and for every $j \neq i'$, $\delta_j = d'_j$.
Assuming that $i \neq i'$, let us consider that $i < i'$. Then, $i' > \rho(a)$, and thus $d_{i'} = ALL_{i'}$. On the other hand, as $\delta_{i'} \in down(d'_{i'})$, $\delta_{i'} \neq ALL_{i'}$. Since we also have $\delta_{i'} = d_{i'}$, we obtain a contradiction. As the case $i' < i$ is similar, we have that $i = i'$, which implies that $\delta_i \in down(d_i) \cap down(d'_i)$. Since H_i is a tree, we have $d_i = d'_i$, which implies that $a = a'$. Thus, the proof is complete. □

It is important to note from the proof above that item (2) of Proposition 4 does not hold if H_i is a DAG but not a tree. This is so because in this case, it is possible that δ_i be in $down(d_i) \cap down(d'_i)$ while $d_i \neq d'_i$. This implies that, if one of the hierarchies H_i is not a tree, there might exist a and a' such that $gen(a) \cap gen(a') \neq \emptyset$.

5.2 Mining Maf-Sequences

We now focus on the first step of the computation of frequent sequences, namely the computation of maf-sequences. This step, referred to as Step 1 at the beginning of the current section, is achieved according to Algorithms 1 and 2, inspired from BUC [Beyer and Ramakrishnan 1999].

Algorithm 1 performs the following tasks:

(1) Computing all frequent minimal atomic sequences, *i.e.*, all sequences of the form $\langle\{(ALL_1,\ldots,ALL_{i-1},d_i,ALL_{i+1},\ldots,ALL_m)\}\rangle$ where d_i is such that $up(d_i)=\{ALL_i\}$. The set of all such sequences is denoted by L_1.

(2) Pruning from the hierarchies H_i ($i\in\{1,\ldots,m\}$) all values d_i such that $\langle\{(ALL_1,\ldots,ALL_{i-1},d_i,ALL_{i+1},\ldots,ALL_m)\}\rangle$ is not frequent and where d_i belongs to any level of H_i.

(3) Based on L_1 and the pruned hierarchies, computing all maf-sequences using a recursive function called *get_rec_maf_seq*, given in Algorithm 2. The set of all maf-sequences is denoted by *MAFS* in the two algorithms.

We note that the first two tasks in Algorithm 1 are achieved through one scan of the blocks by associating each domain element d_i of H_i with the number of blocks containing at least one tuple c such that $c.D_i\in d_i^{\downarrow}$. In Algorithm 1, this number is denoted by $count(d_i)$.

The recursive function shown in Algorithm 2 in order to compute maf-sequences operates on the set L_1, using *pruned* hierarchies. It is important to note in this respect that considering the pruned hierarchies H_i ($i=1,\ldots,m$) allows to avoid redundant candidates that are known in advanced not to be frequent.

We also note that given a candidate atomic sequence $\langle\{a\}\rangle$, $sup(\langle\{a\}\rangle)$ is computed by scanning the reduced blocks that only contain the tuples $c=(t,d_1,\ldots,d_m)$ that support a, *i.e.*, the tuples $c=(t,d_1,\ldots,d_m)$ such that $a\preceq_I(d_1,\ldots,d_m)$. The union of all these reduced blocks is denoted by $\sigma_a(\mathcal{B}(\mathcal{D}_{\mathcal{R}}))$ in Algorithm 2.

It is also important to note that, in Algorithm 2, the successors of frequent atomic sequences are generated in a non redundant manner, based on Definition 9. Moreover, in order to make sure that only maf-sequences are mined, before inserting a given atomic sequence $\langle\{a\}\rangle$ in the set *MAFS*, it is necessary to check whether *MAFS* already contains an atomic sequence more specific than $\langle\{a\}\rangle$. If such is the case, *MAFS* is not changed, otherwise, $\langle\{a\}\rangle$ is inserted in *MAFS*.

EXAMPLE 7. *Fig. 6 illustrates the output of Algorithm 1 in the context of our running example, assuming that* $minsup=\frac{3}{4}$. *The computation of frequent atomic sequences is represented by a tree in which the nodes are of the form* $(d_1,d_2)_s$, *meaning that* $\langle\{(d_1,d_2)\}\rangle$ *is an atomic sequence with support* $\frac{s}{4}$. *Moreover, maf-sequences are displayed as boxed nodes in this tree.*

We note that leaves in this tree are not necessarily maf-sequences. For example, $(ALL_{Loc},pretzel)_3$ *is a leaf, but not an maf-sequence. This is so because* $(ALL_{Loc},pretzel)\preceq_I(EU,pretzel)$ *and* $\langle\{(EU,pretzel)\}\rangle$ *has been identified as being an maf-sequence.*

5.3 Mining Multi-Dimensional Sequences

We now turn to Step 2 mentioned at the beginning of the current section, *i.e.*, the computation of non atomic frequent sequences. We recall in this respect that maf-sequences are

Algorithm 1: Mining Minimal Atomic Frequent Sequences and Pruning Hierarchies

Data: The set $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$, the minimum support threshold *minsup*
Result: The set *MAFS* of all maf-sequences
begin
  foreach $i = 1, \ldots, m$ **do**
    Associate every value d in H_i with $count(d) = 0$;
  foreach $B \in \mathcal{B}(\mathcal{D}_{\mathcal{R}})$ **do**
    foreach *tuple* $c = (t, d_1, \ldots, d_m)$ *in* B **do**
      Update all values $count(\delta_i)$, for every $i = 1, \ldots, m$ and $\delta_i \in d_i^{\uparrow}$;
  /*All counts are computed*/
  $L_1 \leftarrow \emptyset$;
  foreach $d_j \in \bigcup_{i=1}^{i=m} H_i$ **do**
    if $count(d_j) < minsup$ **then**
      /*Pruning H_j*/
      Remove d_j from H_j
    else
      /*Computing L_1*/
      if $up(d_j) = \{ALL_j\}$ **then**
        $L_1 \leftarrow L_1 \cup \{\langle\{(ALL_1, \ldots, ALL_{j-1}, d_j, ALL_{j+1}, \ldots, ALL_m)\}\rangle\}$;
  $MAFS \leftarrow \emptyset$;
  /*Recursive computation of maf-sequences*/
  foreach $a \in L_1$ **do**
    call $get\_rec\_maf\_seq(a, minsup, \sigma_a(\mathcal{B}(\mathcal{D}_{\mathcal{R}})))$;
  return *MAFS*
end

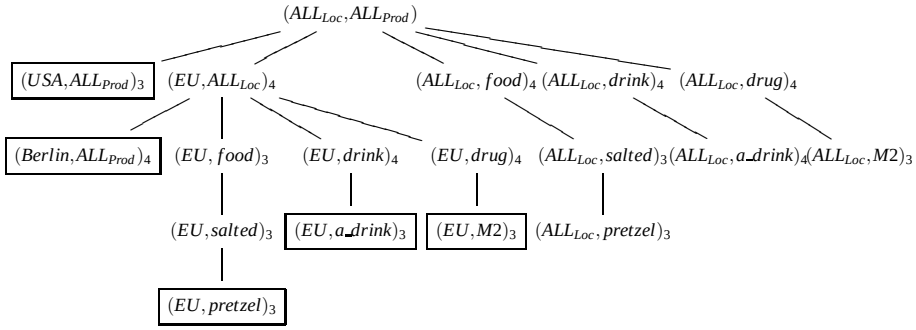


Fig. 6. Tree of frequent atomic sequences

Algorithm 2: Routine *get_rec_maf_seq*

Data: A multi-dimensional item a, the minimum support threshold *minsup*, the set of blocks $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$

Result: The current set *MAFS*

begin

  $Cand \leftarrow \{\alpha = (\delta_1, \ldots, \delta_m) \in gen(a) \mid (\forall i = 1, \ldots, m)(count(\delta_i) \geq minsup)\}$;

  $Freq \leftarrow \{\alpha \in Cand \mid sup(\alpha) \geq minsup\}$;

  if $Freq = \emptyset$ **then**

    if *for every* $\langle\{a'\}\rangle \in MAFS$, $a' \not\preceq_I a$ **then**

      /*$\langle\{a\}\rangle$ `is an maf-sequence` */

      $MAFS \leftarrow MAFS \cup \{\langle\{a\}\rangle\}$;

  else

    foreach $\alpha \in Freq$ **do**

      /*`Recursive call where` $\sigma_\alpha(\mathcal{B}(\mathcal{D}_{\mathcal{R}}))$ `denotes the selection of all tuples` c `in the blocks of` $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$ `such that` $\alpha \preceq_I c.\mathcal{D}_{\mathcal{A}}$ */

      call $get\_rec\_maf\_seq(\alpha, minsup, \sigma_\alpha(\mathcal{B}(\mathcal{D}_{\mathcal{R}})))$;

end

the basic ingredients from which all candidate sequences are built up. More precisely, the multi-dimensional items a occurring in any candidate sequence are such that $\langle\{a\}\rangle$ is an maf-sequence.

Frequent sequences are mined using any classical sequential pattern mining algorithm from the literature ([Masseglia et al. 1998; Zaki 2001; Ayres et al. 2002; Pei et al. 2004]). In our experiments, the algorithm SPADE ([Zaki 2001]) has been used.

Since in standard algorithms, the dataset to be mined is a set pairs of the form (c, seq) where c is a customer identifier and *seq* is a sequence of itemsets, the underlying table T in our approach is transformed as follows:

—Every block B in $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$ is assigned a unique identifier $ID(B)$, playing the role of the customer identifiers in standard algorithms.

—Every maf-sequence $\langle\{a\}\rangle$ is associated with a unique identifier denoted by $id(a)$, playing the role of the items in standard algorithms.

—Every block B in $\mathcal{B}(\mathcal{D}_{\mathcal{R}})$ is transformed into the pair $(ID(B), \varsigma(B))$ where $\varsigma(B)$ is a sequence playing the role of the sequence of itemsets in standard algorithms.

The sequence $\varsigma(B)$ is of the form $\langle(\mu_{11}, \ldots, \mu_{1n_1}), \ldots, (\mu_{p1}, \ldots, \mu_{pn_p})\rangle$ such that, for every $j = 1, \ldots, p$, B contains the tuples $(t_j, a_1), \ldots, (t_j, a_{n_j})$ where $\mu_{jk} = id(a_k)$ $(k = 1, \ldots, n_j)$ and $t_1 < \ldots < t_p$.

The following example illustrates the transformation.

EXAMPLE 8. *Referring back to our running example and considering a support threshold* $minsup = \frac{3}{4}$, *Table II shows the identifiers assigned to the maf-sequences given in Example 7. The blocks of Fig. 4 are displayed in their transformed format in Table III. Moreover, Table IV displays all frequent sequences in their transformed format as well in their multi-dimensional format in which identifiers are replaced with their actual values.*

Maf-sequence $\langle\{a\}\rangle$	$id(a)$
$\langle\{(USA, ALL_{Prod})\}\rangle$	1
$\langle\{(Berlin, ALL_{Prod})\}\rangle$	2
$\langle\{(EU, Pretzel)\}\rangle$	3
$\langle\{(EU, Al.\ Drinks)\}\rangle$	4
$\langle\{(EU, M2)\}\rangle$	5

Table II. Assigning identifiers to maf-sequences

$ID(B)$	$\varsigma(B)$
1	$\langle(2,3,4),(5),(4),(1)\rangle$
2	$\langle(1),(2,3,4),(5)\rangle$
3	$\langle(3,4),(2,5)\rangle$
4	$\langle(1),(2),(1)\rangle$

Table III. Transformed database

Transformed frequent sequences	Frequent multi-dimensional sequences
$\langle(3,4),(5)\rangle$	$\langle\{(EU, pretzel),(EU, a\_drink)\},\{(EU, M2)\}\rangle$
$\langle(1)\rangle$	$\langle\{(USA, ALL_{Prod})\}\rangle$
$\langle(2)\rangle$	$\langle\{(Berlin, ALL_{Prod})\}\rangle$

Table IV. Frequent multi-dimensional sequences for $minsup = \frac{3}{4}$

We end this section by discussing possible extensions on our algorithms regarding the way frequent sequences could be presented to the users.

First, due to the fact that we consider multi-dimensional patterns, one could think of presenting multi-dimensional sequences in a condensed way, by not repeating dimension values occurring at several places. For instance, the sequence $\langle\{(EU, pretzel),(EU, a\_drink)\}, \{(EU, M2)\}\rangle$ could be displayed as $\langle(EU)\{(pretzel),(a\_drink)\},\{(M2)\}\rangle$. However, this process is not always applicable, as for example, the sequence $\langle\{(EU, pretzel),\ (US,\ a\_drink)\},\ \{(EU, M2)\}\rangle$ cannot be condensed.

Second, knowing the blocks that support a given frequent multi-dimensional sequence could be considered relevant. As seen in Example 4, the multi-dimensional sequence $\varsigma = \langle\{(EU, pretzel),\ (EU, a\_drink)\},\ \{(EU, M2)\}\rangle$ is frequent because it is supported by the blocks $B(Educ, Y)$, $B(Educ, O)$ and $B(Ret, Y)$. Thus, it is possible to output this set of blocks, associated with the sequence.

However, as this set might be huge, one could think of *generalizing* the associated tuples, using the hierarchies defined on the reference dimensions in $\mathcal{D}_{\mathcal{R}}$. Then, this option would lead to results of the form (r, ς) where r is a tuple over $\mathcal{D}_{\mathcal{R}}$ and ς is a frequent multi-dimensional sequence, which is precisely what has been considered in [Pinto et al. 2001].

Unfortunately in some cases, such a generalization might be reduced to *ALL*-values, which contains no information at all. Such is the case for the sequence ς above, as the most specific generalization of $(Educ, Y)$, $(Educ, O)$ and (Ret, Y) is (ALL_{Cat}, ALL_{Age}).

Notice in this respect that considering the blocks $B(Educ, Y)$ and $B(Educ, O)$ only, leads to the generalization $(Educ, ALL\_Age)$ and that, displaying $((Educ, ALL\_Age), \varsigma)$ with support 0.5 as a result, means that 50% of the blocks are characterized by specializations of $(Educ, ALL\_Age)$ and support ς.

However, the generalization as shown above is not unique. Indeed, the generalization

(ALL_Cat, Y) of $B(Educ, Y)$ and $B(Ret, Y)$ would also lead to a similar result, namely $((ALL\_Cat, Y), \varsigma)$ with support 0.5.

Therefore, the two main shortcomings of such an extension are that (i) it requires further computation for obtaining the possible generalizations, and that (ii) several results exist, which might be confusing for a naive user.

6. EXPERIMENTS

The algorithms of our approach have been implemented in Java 1.5. Experiments have been carried out both on synthetic and real data, aiming at studying the scalability issues with respect to the main parameters when mining multi-dimensional databases (*e.g.*, the support threshold, the number of dimensions, the number of elements in every dimension).

Firstly, we study the impact of taking hierarchies into account. Namely, we show that considering the different levels of hierarchies as separate attributes and running our previous M^2SP algorithm in this case, is less efficient than running M^3SP. Secondly, we discuss the behavior of our algorithms in the case of various synthetic datasets, and thirdly, the case of a real dataset is considered.

6.1 Hierarchy Management: M^2SP vs. M^3SP

As mentioned previously, taking hierarchies into account is one of the main features of the current approach, compared to that of [Plantevit et al. 2005]. On the other hand, instead of considering herarchies as predefined trees over the domain values, it is possible to consider every non root level of every hierarchy as a separate attribute and to incorporate the corresponding values into the dataset accordingly.

For example the tuple $(1, Berlin, beer)$ in the block $(Educ, Y)$ of Fig. 4 can be replaced with $(1, EU, Berlin, drink, a\_drink, beer)$ as $Berlin^{\uparrow} = \{Berlin, EU, ALL_{Loc}\}$ and $beer^{\uparrow} = \{beer, a\_drink, drink, ALL_{Prod}\}$. However, it should be noticed in this respect that such a transformation requires that all hierarchies be perfectly height-balanced trees (*i.e.*, trees in which all paths from the root to a leaf have the same length).

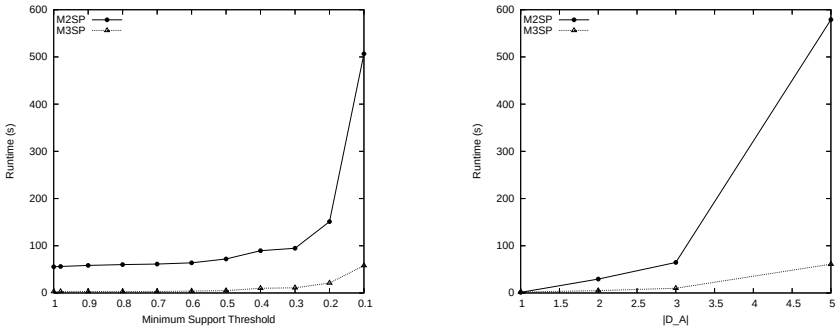
We recall that this way of managing hierarchies has been used by [Srikant and Agrawal 1996] in the case of a single analysis dimension (see Section 3.4). However, in the case of multiple analysis dimensions such a transformation results in a significant increase of the number of analysis dimensions, namely the number of analysis dimensions in the transformed dataset is the sum of the heights of all hierarchies defined on the dimensions of $\mathcal{D}_{\mathcal{A}}$. Then, with the transformed dataset at hand, M^2SP can be run in order to compute all frequent multi-dimensional sequences.

The experiments reported below show that such a transformation is much less efficient than a direct hierarchy management with M^3SP. Fig. 7(a) and Fig. 7(b) show that M^3SP outperforms M^2SP when applied on the transformed dataset. We note that, in these experiments, the cost of the transformation is not taken into account in the reported runtimes.

Considering a synthetic dataset with $50,000$ blocks and 4 analysis dimensions, Fig. 7(a) describes the behavior of the runtimes of the two approaches (*i.e.*, M^3SP and the simulation of hierarchy management with M^2SP) for values of the minimum support threshold ranging from 100% down to 10%. Moreover, in this experiment, the average height of hierarchies is 4, implying that taking hierarchies into account with M^2SP requires the transformation of the original dataset with 4 analysis dimensions into a new dataset over 16 analysis dimensions.

It can be seen that the runtime of M^2SP is significantly greater than that of M^3SP, as the runtime of M^3SP for $minsup = 10\%$ is less than that of M^2SP for $minsup = 100\%$.

Next, we consider a synthetic dataset with $30,000$ blocks and a support threshold equal to 50%. Fig. 7(b) describes the behavior of the runtimes of M^3SP and the simulation of hierarchy management with M^2SP for values of the number of analysis dimensions ranging from 1 to 5. The figure exhibits an important difference of runtime values when the number of analysis dimensions increases, showing that it is not possible to *efficiently* manage hierarchies using M^2SP.



(a) Runtime over minimum support threshold

(b) Runtime over the number of analysis dimensions

Fig. 7. Comparison between M^3SP and M^2SP

6.2 Experiments on Synthetic Datasets Using M^3SP

In this section, we study the scalability of M^3SP according to several parameters, namely the minimum support threshold, the size of the dataset (expressed in number of blocks), the number of analysis dimensions, the height and the degree (*i.e.,* the average number of direct successors) of the hierarchies.

The results of these experiments are summarized in Figures 8-12. The default values of the parameters, when fixed, are as follows:

—the minimum support threshold is set to 30%,

—the number of blocks is set to $250,000$,

—the number of analysis dimensions is set to 5, and

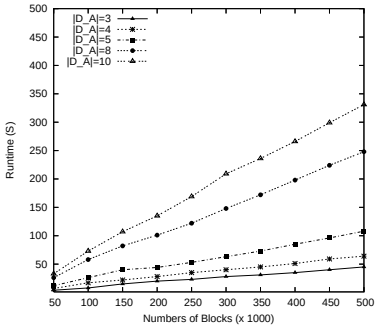
—the height and the degree of the hierarchies are respectively set to 5 and 3.5, which results in about $1,250$ distinct domain values per analysis dimension.

Fig. 8 describes the behavior of M^3SP in terms of runtime and memory usage according to the size of the dataset, expressed in number of blocks, considering that each block contains 10 tuples.

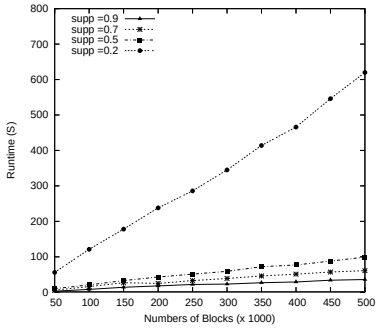
Fig. 8(a) and Fig. 8(b) show how the runtime of M^3SP behaves according to the size of the dataset. In Fig. 8(a), the number of analysis dimensions ranges from 3 to 10 and in

Fig. 8(b), the support threshold ranges from 90% down to 20%. It can be seen from these two figures that the runtime roughly increases proportionally to the size of the dataset, but still keeps with acceptable values. Indeed, the runtime does not exceed 600 seconds in the worst cases, *i.e.*, when the dataset contains $500,000$ blocks (that is $5,000,000$ tuples), and when 10 analysis dimensions or a minimum support threshold of 20% are considered.

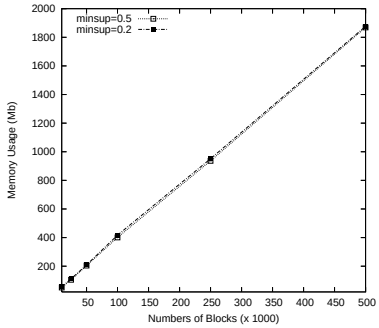
On the other hand, Fig. 8(c) describes the memory usage of M^3SP according the size of the dataset for two different values of the minimum support threshold (50% and 20%). It can be seen that memory usage roughly increases proportionally to the size of the dataset, but is independent from the minimum support threshold value.



(a) Runtime over the size of the dataset (different settings of $|\mathcal{D}_{\mathcal{A}}|$)



(b) Runtime over the size of the dataset (different settings of *minsup*)



(c) Memory usage over the size of the dataset

Fig. 8. Experiments for the size of the dataset

Fig. 9 describes the behavior of M^3SP in terms of runtime, memory usage and number of maf-sequences according to the minimum support threshold.

Fig. 9(a) and Fig. 9(b) show how the runtime of M^3SP behaves according to the minimum support threshold for different numbers of analysis dimensions (*i.e.*, respectively 2, 5 and 10 dimensions for Fig. 9(a), and 15 and 20 dimensions for Fig. 9(b)). Obviously, the runtime increases when *minsup* decreases and when the number of analysis dimensions increases. However, we observe that, for up to 10 analysis dimensions, the runtime does not exceed 25 seconds, even when the minimum support threshold value is below 20%. On the other hand, for 20 analysis dimensions, the runtime increases very fastly for minimum support values below 50%, but still, remains less than 600 seconds.

Fig. 9(c) shows how the numbers of frequent atomic sequences and maf-sequences behave with respect to the minimum support threshold. The number of maf-sequences obviously increases exponentially when the minimum support threshold decreases, but we notice that the reduction of the number of frequent atomic sequences, compared to the number of maf-sequences, is about 50%. This shows that considering maf-sequences is an important issue of our approach, regarding efficiency.

Fig. 9(d) shows how the memory usage of M^3SP behaves according to the support threshold. We point out that M^3SP is robust according to this parameter. Indeed, even if the memory usage of M^3SP increases when the support threshold decreases, the slope has low values and does not significantly increase, even when the minimum support threshold value is below 20%.

Fig. 10 describes the behavior of M^3SP according to the number of analysis dimensions. Fig. 10(a) shows the behavior of the runtime of M^3SP according to the number of analysis dimensions. Although increasing significantly, the runtime is still less that $1,000$ seconds for up to 10 analysis dimensions and a low minimum support threshold value (20%). As shown in Fig. 10(b), this increase mainly comes from the fact that the number of atomic sequences increases significantly with the number of analysis dimensions.

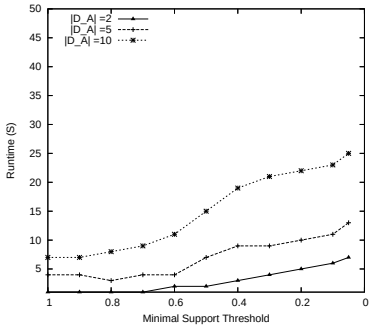
Fig. 10(c) describes the memory usage of M^3SP according to the number of analysis dimensions. It should be noticed that memory usage does not increase significantly when the value of the minimum support threshold decreases. On the other hand, we observe that the slopes of the two plots are decreasing when the number of analysis dimensions increases, which shows that, even for large numbers of analysis dimensions (more than 15), the memory usage should still be acceptable (*i.e.*, about $1,500$ Mb).

To study the behavior of M^3SP according to the "topology" of the hierarchies, we carried out experiments on 10 datasets with the same parameter default values as given at the beginning of the current subsection, but with different values for the degree and the height of the hierarchies associated to analysis dimensions.

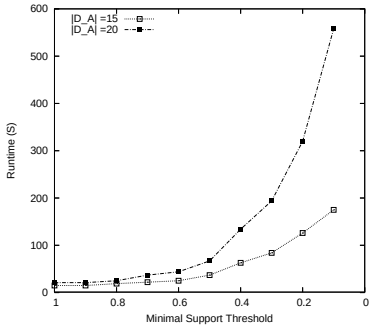
Fig. 11 describes the behavior of M^3SP according to the degree of the hierarchies. Fig. 11(a) shows the runtime of M^3SP when the degree of hierarchies changes. Obviously, decreasing the value of the minimum support threshold implies that runtime increases significantly. However, it should be noticed that increasing the degree of the hierarchies does not significantly affect the performance.

Fig. 11(b) shows that, although the number of frequent atomic sequences increases with the degree of hierarchies, the number of maf-sequences decreases. This again shows that considering maf-sequences is an important feature of our approach regarding efficiency.

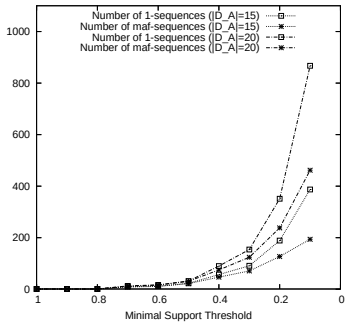
Moreover, as shown in Fig. 11(c), the memory usage is still acceptable (*i.e.*, close to $1,000$ Mb) according to the degree of the hierarchies, and again, is almost independent from the minimum support threshold value.



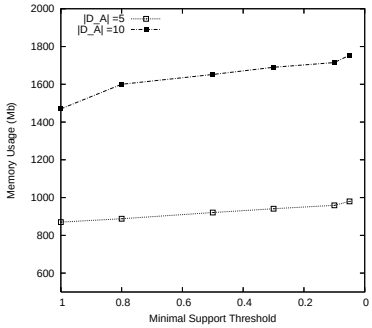
(a) Runtime over the minimum support threshold



(b) Runtime over the minimum support threshold



(c) Number of frequent atomic sequences and maf-sequences over minimum support threshold



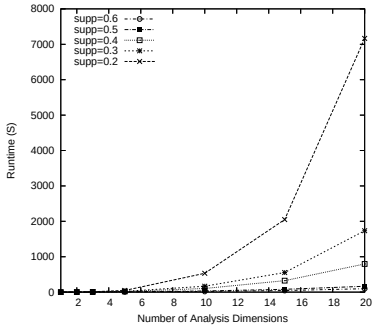
(d) Memory usage over the minimum support threshold

Fig. 9. Experiments for the minimum support threshold

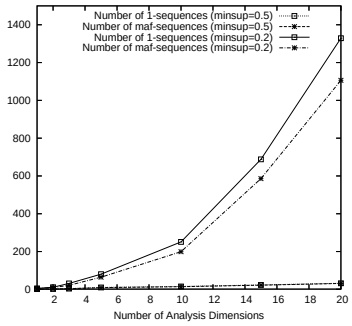
Fig. 12 describes the behavior of M^3SP according to the average height of the hierarchies associated to the analysis dimensions.

Fig. 12(a) shows the runtime over the average height of the hierarchies. We note that, although increasing significantly, the runtime is less than $1,000$ seconds for hierarchies of height at most 7, even when considering a low minimum support threshold value (*i.e.*, 30%). However, the slope of the plot increases significantly when the minimum support threshold is lower than 50%, because, in this case, M^3SP goes deeper in the search space. As an example, the search space for 5 analysis dimensions and an average depth of 10 is modelled as a lattice with 50 levels.

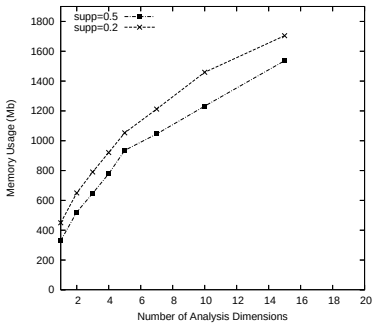
As shown in Fig. 12(b), the number of frequent atomic sequences increases when the average depth decreases, but we stress that such is not the case when considering maf-sequences. This again shows that considering maf-sequences is an important issue of our



(a) Runtime over the number of analysis dimensions



(b) Number of frequent atomic sequences and maf-sequences over the number of analysis dimensions



(c) Memory usage over the number of analysis dimensions

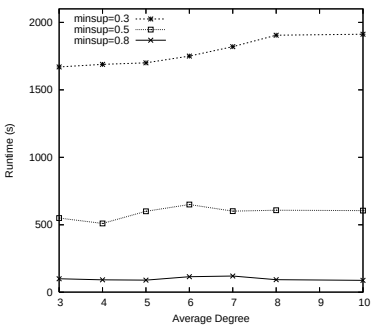
Fig. 10. Experiments for the number of analysis dimensions

approach regarding efficiency.

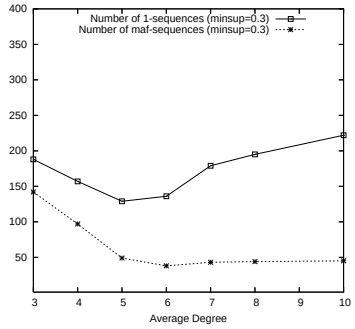
Regarding memory usage, we note from Fig. 12(c) that, on the one hand, it is quite stable when the average height of hierarchies increases, and on the other hand, it does not significantly depend on the minimum support threshold value.

6.3 Experiments on Real Datasets

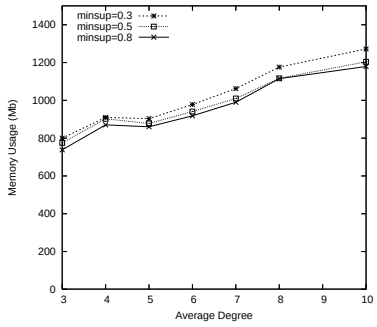
We also carried out experiments on real data from the Marketing Department of EDF (Electricité de France), in the context of a research collaboration between EDF Research and Development and LIRMM laboratory, aiming at studying OLAP Mining for discovering



(a) Runtime over the average degree of the hierarchies



(b) Number of frequent atomic sequences and maf-sequences over the average degree of the hierarchies



(c) Memory usage over the average height of the hierarchies

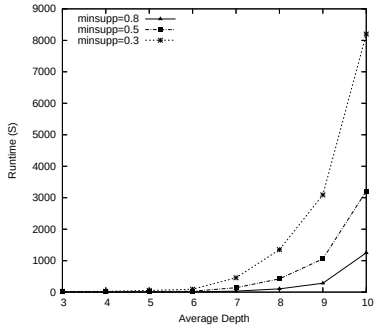
Fig. 11. Experiments for the average degree of the hierarchies

non typical temporal evolutions in data cubes[1].

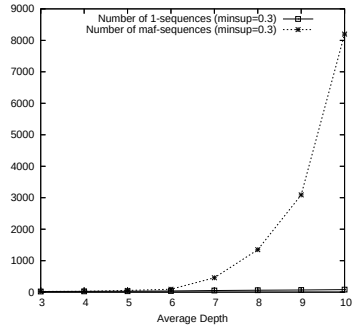
The two considered datasets, referred to as EDF1 and EDF2, describe the marketing activity, based on a very large customer database (about 30.10^6 customers). The dataset EDF1 (Fig. 13(a) and Fig. 13(b)) contains 8 blocks and 6 analysis dimensions, and the dataset EDF2 (Fig. 13(c) and Fig. 13(d)) contains 85 blocks and 5 analysis dimensions. In both cases, the average height of the hierarchies is about 2.

Fig. 13(a) and Fig. 13(c) describe the behavior of the runtime of M^3SP according to the minimum support threshold for the two datasets. We note that runtimes in the case of the dataset EDF1 are of the same order as those in the case of synthetic data. However, in the case of the dataset EDF2, the runtime exceeds $1,500$ seconds for low support values,

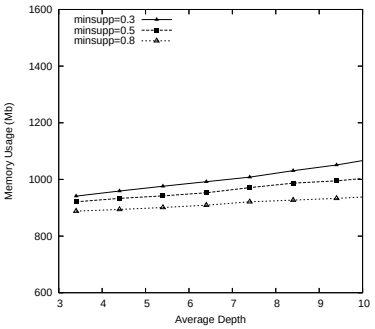
[1]The authors would like to thank Françoise Guisnel, Sabine Goutier and Marie-Luce Picard from EDF R&D for providing real data to assess our approach.



(a) Runtime over the average height of the hierarchies



(b) Number of frequent atomic sequences and maf-sequences over the average height of the hierarchies



(c) Memory usage over the average height of the hierarchies

Fig. 12. Experiments for the average height of the hierarchies

which is twice as much as in the worst case of synthetic datasets. This is due to the fact that the number of maf-sequences is very important in this case.

Fig. 13(b) and Fig. 13(d) describe the behavior of the numbers of frequent atomic and maf-sequences according to the minimum support threshold for the two datasets. Interestingly enough, it can be seen in both cases that the number of maf-sequences is much lower than that of frequent atomic ones. The ratio is about 10%, which is much lower than that observed in the case of synthetic data. This clearly confirms the interest of mining maf-sequences first, as this incures a drastic reduction of the number of candidate sequences in the subsequent steps of the computation.

Moreover, these experiments show that, using M^3SP, relevant rules involving several levels of hierarchies could be found. As an example, we have considered one reference

dimension decribing the type of heating system of customers places along with the following three analysis dimensions, the description of which is simplified for confidentiality reasons:

(1) the type of contract for which a two-level hierarchy is defined (contract type generalized into contract category).
(2) the location for which a three-level hierarchy is defined (namely, town generalized into district and district generalized into area), and
(3) the type of place where the customers stay (e.g., appartment, individual house,...).

In this setting, for $minsup = 50\%$, the following multi-dimensional sequence was found to be frequent:

$$\langle\{(Opt1, PACA, Appt), (Opt1, Bordeaux, House)\}, \{(Opt2, North, Appt)\}\rangle$$

meaning that, for at least half of the types of heating systems,

(1) customers living in the *PACA* district and staying in an appartment subscribed a contract of category $Opt1$ at the same time as customers living in the town *Bordeaux* and staying in an invidual house, also subscribed a contract of category $Opt1$, and then
(2) customers living in the *North* area and staying in an appartment subscribed a contract of category $Opt2$.

The users showed interest in this frequent pattern, since it revealed an unexpected subscription sequence concerning customers staying in locations of the south of France and those staying in the north area of France.

On the other hand, it should be clear that such a pattern, involving several analysis dimensions with non trivial hierarchies, could not have been discovered using standard approaches that cannot deal with multi-dimensional and multi-level sequences.

6.4 Discussion

The experiments on both synthetic and real datasets reported in this section show that our approach allows to *efficiently* mine frequent multi-dimensional and multi-level sequences from huge datasets. Compared to the preliminary version of this work in [Plantevit et al. 2006], the algorithms have been significantly improved. The major improvements are the following:

—In Step 1, maf-sequences are mined according to a BUC-based strategy, instead of an Apriori-like strategy, as done in [Plantevit et al. 2006].
—The transformation of the underlying database presented at the end of Section 5 allows for a better memory usage than handling the dataset itself, as done in [Plantevit et al. 2006].
—In Step 2, based on the transformation mentioned above, any standard algorithm for mining frequent sequences can be used. Having compared the efficiencies of these algorithms, SPADE has shown the best performance. Thus, this algorithm was chosen, instead of PSP, as done in [Plantevit et al. 2006].

We note that our approach may lead to the study of the relationship between the moving up and down in the hierarchies and the tuning of the minimum support threshold. To

see this, given a multi-dimensional item $a = (d_1, \ldots, d_m)$ let $level(a) = \sum_{i=1}^{i=m} l_i$ where, for $i = 1, \ldots, m$, l_i is the level of d_i in the hierarchy H_i.

Then, given a minimum support threshold *minsup*, the average level of all levels of the frequent maf-sequences can provide information on how the value of *minsup* relates to the average level of the items occurring in frequent sequences.

One could even think of a pre-processing phase to compute maf-sequences for some given values of *minsup*. In this way, one could not only get efficiently the frequent multi-dimensional sequences given a minimum support threshold (as the output of the pre-processing phase could be used to optimize the computation), but also, get the appropriate minimum support threshold value, given a desired average level of detail in the frequent multi-dimensional sequences to be computed.

Although this issue has not been investigated in the present work, it can be considered as a valuable extension of our approach.

7. CONCLUSION

In this paper, we have proposed a novel approach for mining multi-dimensional and multi-level sequential patterns, according to which, contrary to the work from [Pinto et al. 2001; de Amo et al. 2004; Yu and Chen 2005], several analysis dimensions at several detailed levels can be taken into account.

We have provided formal definitions and properties from which algorithms have been designed and implemented. Experiments on synthetic and real datasets have been reported and show the interest and the scalability of our approach.

This work offers several research perspectives. First, as mentioned above, the study of the relationship between the moving up and down in the hierarchies and the tuning of the minimum support threshold can be an interesting extension of the present work.

On the other hand, the efficiency of the extraction could be enhanced by considering condensed representations of the mined knowledge, based on the notions of closed, free, or non-derivable patterns [Mannila and Toivonen 1996; Pasquier et al. 1999a; 1999b; Pei et al. 2000; Burdick et al. 2001; Zaki and Hsiao 2002; Calders and Goethals 2002; Boulicaut et al. 2003; Zaki 2004; Bonchi and Lucchese 2006; Calders et al. 2006].

Finally, we are aware that considering maf-sequences to generate the frequent sequences prevents us from extracting all most specific sequences. This is so because such sequences may contain items that do not occur in maf-sequences, and thus, cannot be mined by our approach. Coping with this issue is another interesting extension of the present work that we plan to investigate in the near future.

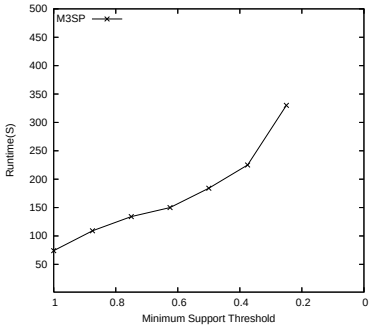
ACKNOWLEDGMENT

The authors wish to thank the referees whose valuable comments helped improving significantly a preliminar version of the paper.

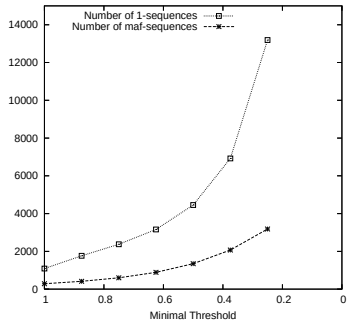
REFERENCES

AGRAWAL, R. AND SRIKANT, R. 1995. Mining sequential patterns. In *International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 3–14.

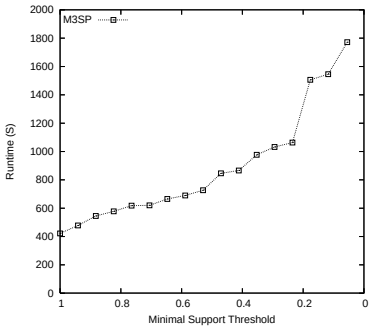
AYRES, J., FLANNICK, J., GEHRKE, J., AND YIU, T. 2002. Sequential pattern mining using a bitmap representation. In *International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press, 429–435.



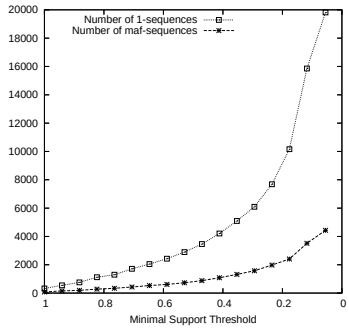
(a) Dataset EDF1: Runtime over minimum support threshold



(b) Dataset EDF1: Number of frequent atomic sequences and maf-sequences over minimum support threshold



(c) Dataset EDF2: Runtime over minimum support threshold



(d) Dataset EDF2: Number of frequent atomic sequences and maf-sequences over minimum support threshold

Fig. 13. Experiments on real data

BEYER, K. AND RAMAKRISHNAN, R. 1999. Bottom-up computation of sparse and iceberg cube. In *International Conference on Management of Data (SIGMOD)*. ACM Press, 359–370.

BONCHI, F. AND LUCCHESE, C. 2006. On condensed representations of constrained frequent patterns. *Knowledge and Information Systems 9*, 2, 180–201.

BOULICAUT, J.-F., BYKOWSKI, A., AND RIGOTTI, C. 2003. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery 7*, 1, 5–22.

BURDICK, D., CALIMLIM, M., AND GEHRKE, J. 2001. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 443–452.

CALDERS, T. AND GOETHALS, B. 2002. Mining all non-derivable frequent itemsets. In *Principles and Practice of Knowledge Discovery in Databases (PKDD)*. LNCS, vol. 2431. Springer Verlag, 74–85.

CALDERS, T., RIGOTTI, C., AND BOULICAUT, J.-F. 2006. A survey on condensed representations for frequent

sets. In *Constraint-Based Mining and Inductive Databases: European Workshop on Inductive Databases and Constraint Based Mining*. LNCS, vol. 3848. Springer Verlag, 64–80.

CHIU, D.-Y., WU, Y.-H., AND CHEN, A. L. P. 2004. An efficient algorithm for mining frequent sequences by a new strategy without support counting. In *International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 375–386.

DE AMO, S., FURTADO, D. A., GIACOMETTI, A., AND LAURENT, D. 2004. An apriori-based approach for first-order temporal pattern mining. In *Simpósio Brasileiro de Bancos de Dados*. 48–62.

DIETTERICH, T. AND MICHALSKI, R. 1985. Discovering patterns in sequences of events. *Artificial Intelligence 25*, 2, 187–232.

HAN, J. AND FU, Y. 1999. Mining multiple-level association rules in large databases. *IEEE Transactions on Knowledge and Data Engineering 11*, 5, 798–804.

INMON, W. 2003. *Building the Data Warehouse*. John Wiley and Sons.

MANNILA, H. AND TOIVONEN, H. 1996. Multiple uses of frequent sets and condensed representations. In *International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press, 189–194.

MANNILA, H., TOIVONEN, H., AND VERKAMO, A. 1995. Discovering frequent episodes in sequences. In *International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press, 210–215.

MASSEGLIA, F., CATHALA, F., AND PONCELET, P. 1998. The PSP approach for mining sequential patterns. In *Principles and Practice of Knowledge Discovery in Databases (PKDD)*. LNCS, vol. 1510. Springer Verlag, 176–184.

PASQUIER, N., BASTIDE, Y., TAOUIL, R., AND LAKHAL, L. 1999a. Discovering frequent closed itemsets for association rules. In *International Conference on Database Theory (ICDT)*. LNCS, vol. 1540. Springer Verlag, 398–416.

PASQUIER, N., BASTIDE, Y., TAOUIL, R., AND LAKHAL, L. 1999b. Efficient mining of association rules using closed itemset lattices. *Information Systems 24*, 1, 25–46.

PEI, J., HAN, J., AND MAO, R. 2000. Closet: An efficient algorithm for mining frequent closed itemsets. In *SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. ACM Press, 21–30.

PEI, J., HAN, J., MORTAZAVI-ASL, B., WANG, J., PINTO, H., CHEN, Q., DAYAL, U., AND HSU, M.-C. 2004. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering 16*, 11, 1424–1440.

PINTO, H., HAN, J., PEI, J., WANG, K., CHEN, Q., AND DAYAL, U. 2001. Multi-dimensional sequential pattern mining. In *International Conference on Information and Knowledge Management (CIKM)*. ACM Press, 81–88.

PLANTEVIT, M., CHOONG, Y. W., LAURENT, A., LAURENT, D., AND TEISSEIRE, M. 2005. M2SP: Mining sequential patterns among several dimensions. In *Principles and Practice of Knowledge Discovery in Databases (PKDD)*. LNAI, vol. 3721. Springer Verlag, 205–216.

PLANTEVIT, M., LAURENT, A., AND TEISSEIRE, M. 2006. HYPE: Mining hierarchical sequential patterns. In *International Workshop on Data Warehousing and OLAP (DOLAP)*. ACM Press, 19–26.

RASHAD, S., KANTARDZIC, M. M., AND KUMAR, A. 2007. MSP-CACRR: Multidimensional Sequential Patterns Based Call Admission Control and Resource Reservation for Next-Generation Wireless Cellular Networks. In *Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE Computer Society, 552–559.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Extending Data Base Technology (EDBT)*. LNCS, vol. 1057. Springer Verlag, 3–17.

STEFANOWSKI, J. 2007. Algorithms for context based sequential pattern mining. *Fundamenta Informaticae 76*, 4, 495–510.

STEFANOWSKI, J. AND ZIEMBINSKI, R. 2005. Mining context based sequential patterns. In *Atlantic Web Intelligence Conference (AWIC)*. LNCS, vol. 3528. Springer Verlag, 401–407.

YANG, Z., KITSUREGAWA, M., AND WANG, Y. 2006. Paid: Mining sequential patterns by passed item deduction in large databases. In *International Database Engineering and Applications Symposium (IDEAS)*. IEEE Computer Society, 113–120.

YU, C.-C. AND CHEN, Y.-L. 2005. Mining sequential patterns from multidimensional sequence data. *IEEE Transactions on Knowledge and Data Engineering 17*, 1, 136–140.

ZAKI, M. J. 2001. SPADE: an efficient algorithm for mining frequent sequences. *Machine Learning Journal, Special issue on Unsupervised Learning 42*, 1/2, 31–60.

ZAKI, M. J. 2004. Mining non-redundant association rules. *Data Mining and Knowledge Discovery 9*, 223–248.

ZAKI, M. J. AND HSIAO, C.-J. 2002. CHARM: an efficient algorithm for closed itemset mining. In *SIAM International Conference on Data Mining (SDM)*. SIAM.

ZHANG, C., HU, K., CHEN, Z., CHEN, L., AND DONG, Y. 2007. ApproxMGMSP: A scalable method of mining approximate multidimensional sequential patterns on distributed system. In *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. Vol. 2. IEEE Computer Society, 730–734.

Received August 2008; Revised January 2009; Accepted May 2009