Time Aware Mining of Itemsets

Bashar Saleh, Florent Masseglia

Time Aware Mining of Itemsets

Bashar Saleh
INRIA
2004 route des lucioles - BP 93
bsaleh@sophia.inria.fr

Florent Masseglia
INRIA
2004 route des lucioles - BP 93
fmassegl@sophia.inria.fr

Abstract

Frequent behavioural pattern mining is a very important topic of knowledge discovery, intended to extract correlations between items recorded in large databases or Web acces logs. However, those databases are usually considered as a whole and hence, itemsets are extracted over the entire set of records. Our claim is that possible periods, hidden within the structure of the data and containing compact itemsets, may exist. These periods, as well as the itemsets they contain, might not be found by traditional data mining methods due to their very weak support. Furthermore, these periods might be lost depending on an arbitrary division of the data. The goal of our work is to find itemsets that are frequent over a specific period but would not be extracted by traditional methods since their support is very low over the whole dataset. In this paper, we introduce the definition of solid itemsets, which represent a coherent and compact behavior over a specific period, and we propose SIM*, an algorithm for their extraction. This work may find many applications in sensitive domains such as fraud or intrusion detection.*

1 Introduction

The problem of association rule mining has been defined in [1]. The goal is to obtain, among a very large set of records, the frequent correlations between the items of the database. This problem has many application in marketing, business management or decision analysis, for instance. The core of this problem lies in the extraction of frequent itemsets. In market basket analysis, for instance, frequent itemset mining aims to discover sets of items that correspond to a large number of customer. If this number is above a certain threshold (given by the user) then this itemset is considered frequent. However, in the initial definition of frequent itemset mining, the search is performed over the whole database (*i.e.* given min_{supp}, the user's minimum support, the extracted itemsets appear in at least $|D| \times min_{supp}$ transactions of database D). However, for many real world applications, this definition of frequent itemsets is not well adapted. Possible interesting itemsets might remain undiscovered despite their very specific characteristics. In fact, interesting itemsets are often related to the moment during which they can be observed. We may consider, for instance, the behaviors of the users on the web site of an on-line store after a special discount on recordable DVDs and CDs, advertised on TV. Another example could be the adverse drug reports related to a specific drug that appeared after an alert was publicized on that precise drug. Similarly, the web site of a conference will observe that frequent behavior related to the submission procedure mainly occurs within a window of a few hours before the deadline. A necessary condition in order to discover this kind of knowledge is that each transaction is associated to a time-stamp. This condition has already been proposed, for instance in [2] and the authors proposed the notion of temporal association rules. Their idea consists of extracting itemsets that are frequent over a specific period that is shorter than the whole database. However, the periods proposed in [2] are defined by the lifetime of each item. Therefore, a data mining process for extracting the periods is not necessary since they only depend on the first and last occurrence of each item.

In this paper, we propose to find itemsets that are frequent over a contiguous subset of the database. For instance, navigations on the web page of recordable CDs and DVDs occur randomly all year, but the correlation between both items is not frequent if we consider the whole year. However, the frequency of this behavior will certainly be higher within the few hours (or days) that follow the TV spot. Therefore, the challenge is to find the time window that will optimize the support of this behavior. In other words, we want to find B, a contiguous subset of D where the support of the behavior on B is above the minimum support and the size of B is optimal. Let us consider that the TV spot was on March 3 and it has influenced the customers for two days. Our goal is to find the following kind of knowledge: "25% of the users, between March 3 and

March 5, have requested the page about recordable CDs, the page about recordable DVDs and the page about special discounts." The support of this behavior would certainly be too low for its extraction over the whole year, but this knowledge (*i.e.* the behavior along with its associated period of frequency) may be very important for deciders since they will want to discover this behavior and its specific window of frequency, and finally link it to the TV spot.

This problem could seem similar to the problem of mining bursty events in data streams [9, 4]. However, we will show that our method is able to combine several requirements that have not yet been met together in the fields of burst mining or data stream mining (*i.e.* we are able to extract itemsets with no fixed window size and to obtain the exact and exhaustive set of periods of optimal frequency for these itemsets).

The remainder of this paper is organized as follows. Section 2 gives the necessary definitions of itemset discovery and our new definitions for mining solid itemsets (with a comparison to temporal aspects of itemset mining in the literature). Section 3 summarizes the complexity of the problem exposed in this paper and Section 4 presents our algorithm for the extraction of solid itemsets. Finally, Section 5 gives a synthesis of our experiments leading to the conclusion of Section 6 with future avenues.

2 Definitions

The problem of association rule mining is based on the extraction of frequent itemsets. This problem has been proposed in [1], and numerous algorithms have been proposed in the literature to solve it. Definition 1 states the characteristics of frequent itemsets. It is different from the initial or traditional definitions in [1] since we consider that each item in the database is associated to a time-stamp. Therefore a transaction may cover a range of several timestamps.

Definition 1 *Let $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ be a set of items. Let $X = \{i_1, i_2, ..., i_k\}/k \leq n$ and $\forall j \in [1..k]\ i_j \in \mathcal{I}$. X is called a* **itemset** *(or a $k-$***itemset***). Let $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ be a set of times, over which a linear order $<_T$ is defined, where $t_i <_T t_j$ means t_i occurs before t_j. A* **transaction** *T is a couple $T = (tid, X)$ where tid is the transaction identifier and X is the associated itemset. Associated to each item i in X we have a time-stamp t_i which represents the valid time of occurrence of i in T.*
A transaction $T = (tid, I)$ is said to support an itemset $X \in \mathcal{I}$ if $X \subseteq I$. A **transaction database** *D is a set of transactions. The* **cover** *of an itemset X in D is the set of transaction identifiers of transaction in D that support X: cover$(X, D) = \{tid/(tid, I) \in D, X \in I\}$. The* **support** *of an itemset X in D is the number of transactions in the cover of X in D: support$(X, D) = |cover(X, D)|$. The* **frequency** *of an itemset X in D is the fraction of transactions in D that support X: frequency$(X, D) = \frac{support(X,D)}{|D|}$. Given a user's minimum threshold $\gamma \in]0..1]$, an itemset X is said to be* **frequent** *if frequency$(X, D) \geq \gamma$.*

Definition 2 *The set F of frequent itemset in D with respect to γ is denoted by $F(D, \gamma) = \{X \in \mathcal{I}/frequency(X, D) \geq \gamma\}$.*

Given a set if items $\mathcal{I}$, a transaction database D and a minimal threshold γ, the problem of **frequent itemset mining** aims to find $F(D, \gamma)$ and the actual support of the itemsets in F. Example 1 gives an illustration of the notions presented above.

Example 1 *Figure 1 shows the example database D. To each transaction Id is associated the set of items in the transaction. In order to simplify the illustration, we assume that the transactions of D are recorded by order of date (i.e. T_1 occurred before T_2, etc.) and a unique time-stamp is associated to all the items of a transaction (whereas in our definition, each separate item has a time-stamp). Let us consider a minimum frequency $\gamma = \frac{1}{2}$ given by the user. With such a support, the frequent items (highlighted in the transactions of figure 1) are a, b and c. The frequent itemsets of D, with $\gamma = \frac{1}{2}$, are (a), (b), (c), with a threshold of $\frac{6}{10}$, and (a, c), with a threshold of $\frac{1}{2}$.*

Tid	items	F(D,1/2)
1	**a b c**	
2	**a c** d	
3	**b** e f	
4	**c** j h	(a)
5	**a** i j	(b)
6	**b** k l	(c)
7	**a b c**	(a c)
8	m n o	
9	**a b c**	
10	**a b c**	

Figure 1. Frequent itemsets on D where $\gamma = \frac{1}{2}$

Our problem is based on the timestamps associated to the records in D and aims to provide itemsets that are frequent on particular periods of times in D. In the following definitions, we introduce the notions of temporal itemset and solid itemset, that are the core of this paper.

Definition 3 *A* **period** *$P = (P_s, P_e)$ is defined by a start time P_s and an end time P_e. The set of transactions that belong to period P is defined as $Tr(P) = \{T/T \subseteq D, \forall i \in T, P_s \leq P_i \leq P_e\}$ with P_i the time-stamp associated to i in transaction T. We define as PR the set of all potential periods over D.*

In other words, the set of transactions that belong to a period P is defined as the set of transactions having all their items associated to a time-stamp in the time range of P. The frequency of x over $Tr(P)$ the transactions of a period P is denoted by frequency(x, P) whenever it is clear from the context (as well as cover($x, Tr(P)$) which is denoted by cover(x, P) and support($x, Tr(P)$) which is denoted by support(x, P)).

Definition 4 *A* **Temporal Itemset** x *is a triple* (x_i, x_p, x_σ) *where* x_i *is an itemset,* x_p *is a period associated to* x_i *and* x_σ *is the threshold of* x_i *over* x_p*. Let* k *be the size of* x_i*, then* x *is called a* $k-$*temporal itemset.*

Let us consider the temporal itemset $y = (\{a, b, c\}, [7..10], \frac{3}{4})$ in D from figure 1. The itemset of y (*i.e.* y_i) is $\{a, b, c\}$. The period of y (*i.e.* y_p) is $[7..10]$ and the threshold of y over y_p (*i.e.* y_σ) is $\frac{3}{4}$ (y_i is supported by transactions 7, 9 and 10 in period y_p on D).

Given γ, a user's minimum threshold, we introduce the characteristics of solid itemsets in Definition 5.

Definition 5 *Let* x *be a temporal itemset.* x *is called a* **Solid Itemset (SI)** *iff the following conditions hold:*
1) $x_\sigma \geq \gamma$
2) $\forall p_2 \in PR/x_p \subseteq p_2$ *we have either a) or b) or both:*
a) support$(x_i, p_2) < \gamma$
b) cover(x_i, p_2) =cover(x_i, x_p)
3) $\forall p_2 \in PR/p_2 \subseteq x_p$, cover$(x_i, p_2)$ <cover(x_i, x_p)
Let k *be the size of* x_i*, then* x *is a* **k-solid itemset**. *Finally,* $\mathbf{SI_k}$ *is the set of all* k*-solid itemsets.*

The first condition of definition 5 ensures that x represents an itemset that is frequent over its associated period. The second condition ensures that the size of x_p is maximal. Actually, if a larger period exists, then, on this period, x_i is not frequent or the cover of x_i is the same (*i.e.* it is not worth extending the period from x_p to p_2, since the extension will not contribute to the support of x_i). Finally, the third condition ensures that the size of x_p is minimal. In fact, x_i is supported by the first and last transaction in x_p, so if a smaller period exists where x_i is frequent, the cover will be lower anyway (*i.e.* relevant transactions supporting x_i would have been dropped from the period and should be kept). An illustration is given in example 2.

Example 2 *Figure 2 shows the example database* D *of figure 1 and the extracted* k*-solid itemsets. We can observe that the solid itemsets of size 1 are* (a)*,* (b) *and* (c)*, and their period corresponds to the entire database with a threshold of* $\frac{6}{10}$*. Then, we have three solid itemsets of size 2:*

- $(a\ c)$*, with a threshold of* $\frac{5}{10}$ *and a period that corresponds to the entire database.*
- $(a\ b)$ *and* $(b\ c)$*, on the period* $[7..10]$ *with a threshold of* $\frac{3}{4}$*.*

Finally, there is one solid itemset having size 3: $(a\ b\ c)$ *which occurs during the period* $[7..10]$ *with a threshold of* $\frac{3}{4}$*. We can observe that, thanks to the definition of solid itemsets, a new kind of knowledge has been extracted. This knowledge concerns punctual behaviors of the users. In* D *it is illustrated by, for instance, a compact itemset of size 3 (i.e.* $(a\ b\ c)$*) occurring on a very specific period (i.e.* $[7..10]$*). This itemset, associated to this period, is optimal (as stated in definition 5) since:*

- *This itemset is frequent over this period.*
- *No longer period allows this itemset to have the minimum threshold (condition 2 in definition 5 is respected for all periods larger than* $[7..10]$*).*
- *no shorter period allows this itemset to have the minimum threshold without diminishing the cover.*

On the other hand, let us consider the following temporal itemsets: $y = ((a\ b\ c), [9..10], 100\%)$ *and* $z = ((a\ b\ c), [6..10], \frac{3}{5})$*. We can observe that* y_i *and* z_i *have the minimum support over their respective periods. However, there exists a period* $p_2 = [7..10]$ *where* $(a\ b\ c)$ *is frequent and the cover is larger than the cover of* y_i *on* y_p*. Hence,* y *is not a solid itemset since condition 2 of definition 5 is not respected. Finally,* z_i *is frequent on* p_2 *and its cover is the same on* p_2 *and* z_p*, so condition 3 of definition 5 is not respected and* z *is not a solid itemset.*

Let us note that itemsets $(a\ b)$*,* $(b\ c)$ *and* $(a\ b\ c)$ *were not frequent over the whole database in example 1 with* $\gamma = \frac{1}{2}$*, since their threshold on* D *is* $\frac{4}{10}$*. However, thanks to the definition of solid itemsets, they can be discovered along with their associated periods of frequency.*

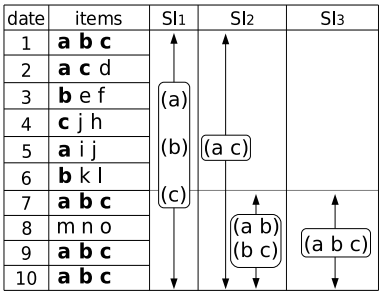

Figure 2. Solid itemsets in D where $\gamma = \frac{1}{2}$

Definition 6 *The set of* **Maximal Solid Itemsets (MSI)** *is defined as follows: let* x *be a SI,* x *is a* MSI *if the following condition holds:*
$\forall y \in SI/x \neq y$ *if* $x_i \subseteq x_i$ *then* $x_p \neq y_p$*.*

The goal of this paper is to propose an optimized algorithm in order to extract the exact and entire set of maximal solid itemsets, as stated in definition 6.

Our problem can be compared to two main fields of data mining: mining burst events from data streams and mining temporal itemsets. An event is considered bursty if it occurs with strong support in a certain time window. The definitions of bursts may vary in the literature, but the idea is generally to find the items that correspond to this time window and a significant threshold [5, 7, 9, 4]. The notion of burst is thus close to our definition of solid itemsets. However, at this time and to the best of our knowledge, there is no method for mining bursty itemsets since the existing methods propose to detect events of one item (except [4] with events made of correlations between multiple items, but with fixed window sizes and disjoined itemsets). Let us mention that mining in data streams implies to find a compromise between the time response and the quality of the result. Hence, approximation is a key in data stream mining methods, whereas in our framework, we want to extract the exact set of solid itemsets without compromise on the quality of the result.

Interesting studies have been proposed for the temporal aspects related to association rule mining. We propose to divide these studies into three main categories:**1) A specific period is given** and the goal is to find the frequent itemsets within this period [2], **2) A specific pattern is given** and the goal is to find the corresponding periods [3] and **3) Mining periodic (repetitive) patterns** and the timestamps are analyzed in order to find repetitive patterns [6]. Eventually, we note that an instructive survey on temporal knowledge extraction can be found in [8].

3 Motivation

As illustrated in example 2, our problem could be seen as a mere lowering of the minimum threshold (the itemset $(a\,b\,c)$ in example 1 has a threshold of $\frac{4}{10}$ over D) in order to find the itemsets corresponding to our solid itemsets. However this point of view has two main drawbacks, compared to our problem definition:

1. Lowering the support is a well known source of failure for existing data mining algorithms. Generally, the number of candidates, or the number of frequent items, will not fit in main memory. Even if this set is able to fit in memory, the response time will be prohibitive.
2. Even with a lower support, if the itemsets are extracted despite their number, they will not be associated to their period of frequency (actually they would be extracted because they are frequent on a period corresponding to the whole database, which is not really instructive from the localization point of view).

Another naive method would consist of dividing the database into multiple subsets corresponding to periods of fixed size. For instance, the web access log file of a shop for one year could be divided into 365 subsets corresponding to each day of this year. In this case, we have to keep in mind that undiscovered periods will remain (for instance a period of two consecutive days or a period of one hour embedded in one of the considered days) and the method would be based on an arbitrary division of the data (why working on each day and not on each hour or week or half day?).

Eventually, let us note that the total amount of combinations for enumerating the possible solid itemsets is $(2^n \times k!)$ with n the number of itemsets and $k = |D|$. So, 2^n is the number of potential itemsets on D and $k!$ is the number of possible contiguous subsets (windows) of D. Fortunately, the monotonicity property of frequent itemsets allows avoiding the enumeration of 2^n possible itemsets. Based on this property our goal is to show that avoiding the enumeration of the $k!$ potential periods is also possible, and we provide in section 4 an exhaustive and optimized algorithm for mining solid itemsets.

4 General Principle & Algorithm

This section is devoted to the presentation of "Solid Itemset Miner" SIM designed for the extraction of solid itemsets in databases. The notion of kernels, introduced in this section, will allow extracting the solid itemsets efficiently. First, we give an overview of the principle and main idea for this extraction in Section 4.1 and the details of the algorithm are given in Section 4.2.

4.1 General Principle

SIM introduces a new paradigm for the counting step of the generated candidates. Actually, let us consider t a temporal itemset that is not a solid itemset (*i.e.* $t_\sigma < gamma$). Any superset $u = (u_x, u_p, u_\sigma)/u_x \subseteq t_x \wedge u_p \subseteq t_p$ of t cannot be a solid itemset (*i.e.* $u_\sigma < gamma$). SIM thus extends the Generating-Pruning principle of apriori in order to generate candidate solid itemsets and count their support. The generating principle is provided with a filter on the possible intersection of the candidates (*i.e.* if two solid itemset of size k have a common prefix but do not share a common period, then they are not considered for generating a new candidate).

However, the counting step (or "pruning" in apriori) is not straightforward in our case. Let us consider c, a candidate. A possible solution would be to count the occurrences of c over its lifetime within c_p. This is not a good solution.

Now let us consider the candidate $c = ((a\ b), [1..10], c_\sigma)$ the candidate temporal itemset that has been generated thanks to the solid itemsets of size 1: $x = ((a), [1..10], \frac{6}{10})$ and $y = ((b), [1..10], \frac{6}{10})$. c is not a solid itemset since $c_\sigma = \frac{4}{10}$. However c_p contains a solid itemset $c' = ((a\ b), [7..10], \frac{3}{4})$. Based on this observation, our goal, during the counting step, is to build "kernels" of the candidate temporal itemsets over their period of possible frequency. Then, the kernels will be merged in order to find the corresponding solid itemsets. Details are given in Definition 7.

date	b	kernels	merge
1	1	Kernel 1: [1..3] threshold=2/3	Itemset: (b) period: [1..10] threshold: 6/10
2	0		
3	1		
4	0		
5	0		
6	1	Kernel 2: [6..10] threshold=4/5	
7	1		
8	0		
9	1		
10	1		

Figure 3. Kernels and period of itemset (b)

The following definition is based on the fact that we perform successive scans over the data in order to find the periods that correspond to solid itemsets. The way a scan is performed (*i.e.* reading the transaction from the first to the last one) implies discovering the kernels "on-the-fly".

Definition 7 *A* **kernel** *is a period. Let $K(x, P, \gamma)$ be the set of kernels for the item x over the period P with respect to the minimum threshold γ. $K(x, P, \gamma)$ is defined as follows: Let $k \subseteq P$ be a period such that $x \subseteq Tr(k_s) \wedge Tr(k_s)$ is the first occurrence of x in P. If k does not exist then $K = \emptyset$. If k exists, then let N be the set of timestamps such that $\forall n \in N, n \in P \wedge n > k_s \wedge frequency(x, [k_s..n]) < \gamma$ (in other words, N is the set of timestamps in P such that extending the period k up to any of those timestamps leads to lose the frequency for x). If N is empty then k_e is defined as the last occurrence of x in P, and $K(x, P, \gamma) = \{k\}$. Otherwise (i.e. $N \neq \emptyset$), let $m \in N / \forall n \in N, n > m$ (m is the first time-stamp such that frequency of x is lost on $[k_s..m]$). Then, k_e is defined as the last occurrence of x in $[k_s..m]$ and $K(x, P, \gamma) = \{k\} \cup K(x, P - [k_s..k_e], \gamma)$*

Example 3 *Let us consider the candidate temporal itemset of size 1 $c = ((b), [1..10], c_\sigma)$. Figure 3 gives the boolean table of occurrences for the item b. There are two kernels of (b) over c_p (i.e. $[1..3]$ and $[6..10]$). Those kernels can be merged (the frequency of the itemset on the resulting period is above the minimum threshold) in order to obtain the resulting solid itemset $((b), [1..10], \frac{6}{10})$.*

Let us consider that we are provided with an itemset x and K the kernels of x over a period P with respect to γ. Based on lemma 1 we show that merging the kernels with algorithm MERGEKERNELS allows finding the solid itemsets of x over P with respect to γ.

Algorithm MERGEKERNELS
In: x an itemset, K a set of kernels for x
and γ a minimum support.
Out: The modifications of $c.kernels$, containing the optimal periods for x with γ
mergeable $\leftarrow$ true;
While (mergeable)
mergeable $\leftarrow$ false;
Foreach ($q \in K$)
Foreach ($r \in K / r \neq q \wedge \frac{cover(x,q)|+|cover(x,r)|}{|q \cup r|} \geq \gamma$)
$K \leftarrow K + q \cup r$;
$cover(x, q \cup r) = cover(x, q) \cup cover(x, r)$
mergeable $\leftarrow$ true;
toRemove $\leftarrow$ toRemove $+q + r$;
endFor
endFor
Foreach ($k \in$ toRemove) $K \leftarrow K - k$;
toRemove $\leftarrow \emptyset$
End while
End Algorithm MERGEKERNELS

Lemma 1 *Let K be the set of kernels of x on P with respect to γ. Algorithm* MERGEKERNELS *allows finding all the solid itemsets $s = (x, x_p, \sigma)$ on P with respect to γ.*

Proof Let $k \in K$, be a kernel of x after Algorithm MERGEKERNELS (*i.e.* k cannot be merged with any other kernel in K), then:
1) Support$(x, k) > \gamma$. Actually, according to Definition 7, x is frequent on each kernel. Furthermore if k is the result of a merging, then Algorithm MERGEKERNEL checks the frequency of x on the resulting period.
2) $\forall q / k \subseteq q$ we have one of the following cases:

- $x \in k - q$, then x is not frequent on q (otherwise, let us consider k' the kernel to which belongs the occurrence of x in q, then k and k' would have been merged).
- $x \notin k - q$, then cover(x, q)=cover(x, k) (in this case, x may remain frequent on q or not, depending on the size of q).

3) According to Definition 7, x is supported by the first and the last transaction in k. Then, x will have a lower cover on any sub-period of k.
Based on the three observations above, let $T_x =$

$\{(x, k, \sigma) \forall k \in K)\}$ be the set of temporal itemsets corresponding to all the merged kernels of x on P with respect to γ, then T_x is the set of all solid itemsets $s = (x, x_p, \sigma)$ on P with respect to γ □

4.2 SIM Algorithm

Our algorithm is based on the candidate generating principle. Our goal is to start with solid itemsets of size 1 and explore the support of larger solid itemsets with a limited number of scans over the database. To this end, we need to find the periods of frequency for a candidate solid itemset in only one scan. Let $c \in C_k$ be a candidate of size k in the set of candidates (C_k). Then, in our data structure, c is associated to $c.i$, the itemset, $c.p$, the period of possible frequency (*i.e.* the limits within c has to be compared to a transaction) and $c.kernels$, the set of kernels of $c.i$ over $c.p$ with respect to γ (one of our goals is to extract $c.kernels$ for all the candidates in C_k during one single scan). Furthermore, a boolean value allows knowing the status of the current kernel ("kernel_closed" means that definition 7 was not respected "on-the-fly" during the scan). For each kernel $c.kernel_i$, of a candidate c, we have $c.kernel_i.s$ (the starting time-stamp of the kernel), $c.kernel_i.e$ (end of the kernel), $c.kernel_i.last$ (the last occurrence of $c.i$ in the current kernel), $c.kernel_i.freq$ (the frequency of $c.i$ over $[c.kernel_i.s..c.kernel_i.e]$) and $c.kernel_i.cov$ (the size of the cover of $c.i$ over $[c.kernel_i.s..c.kernel_i.e]$). Finally, $c.current$ refers to the current kernel of c (the last opened kernel).

Let us consider that we are provided with C_k, a set of candidates. During the scan, the goal of Algorithm UPDATE is to update the information about the kernels of a candidate having a period of scan that includes the time-stamp of the current transaction. At the end of the scan performed by Algorithm SIM we are provided with all the kernels for each candidate.

Algorithm UPDATE
In: c, the candidate; d, the transaction; γ, the threshold
Out: update of the kernel(s) of c
If ($c.kernel\_closed$)
  If ($c.i \subseteq d$) // Start a new kernel
    $c.current \leftarrow new\_kernel$;
    $c.kernel\_closed \leftarrow False$;
    $c.current.s \leftarrow d.timestamp$;
    $c.current.e \leftarrow d.timestamp$;
    $c.current.last \leftarrow d.timestamp$;
  End if
Else if ($c.i \subseteq d$)// Continue the current kernel
  $c.current.e \leftarrow d.timestamp$;
  $c.current.last \leftarrow d.timestamp$;
  $c.current.cov++$;
  $c.current.freq \leftarrow \frac{c.current.cov}{|[c.current.s..c.current.e]|}$;
Else// Check validity of current kernel
// *i.e.* ($c.i \not\subseteq d$) $\Rightarrow$ must current kernel be closed?
  $c.current.e \leftarrow d.timestamp$;
  $c.current.freq \leftarrow \frac{c.current.cov}{|[c.current.s..c.current.e]|}$;
  If ($c.current.freq < \gamma$)
    $c.current.e \leftarrow c.current.last$
    $c.kernel\_closed \leftarrow True$;
  End if
End if
End algorithm UPDATE

Algorithm SIM aims to generate candidates from size 1 to k. At each step, the set of candidates is compared to the database thanks to Algorithm UPDATE. At the end of the scan, the kernels obtained for each candidate temporal itemset are merged in order to obtain the solid itemsets.

Algorithm SIM
In: γ, the minimum threshold; D the database;
  $\mathcal{I}$ the set of all items
Out: SI the set of solid itemsets corresponding to γ on D
$k \leftarrow 0$;
Foreach ($i \in \mathcal{I}$)
  // Build one candidate for each item and associate
  // this candidate to an empty set of kernels
  $C_1 \leftarrow C_1 + (i, [D_s..D_e], \emptyset)$
End for
Do// Successive scans of the database
  $k++$;
  $SI_k \leftarrow \emptyset$;
  Foreach ($d \in D$); // scan the database
    Foreach ($c \in C_k / d_{time} \in c.p$)
    // The timestamp of d corresponds to the period of c
    UPDATE(c,d,γ);
    End for
  End for
  Foreach ($c \in C_k$)
    MERGEKERNELS(c_i, $c.kernels$, γ);
    Foreach ($p \in c.kernels$)
      $SI_k \leftarrow SI_k + (c_i, p, frequency(c_i, p))$;
  End for
  $C_{k+1} \leftarrow$GENERATECANDIDATES(SI_k)
While ($C_{k+1} \neq \emptyset$) // Candidate solid itemsets generated
And algorithm SIM

The generating principle of SIM is based on the following lemma.

Lemma 2 *Let γ be the minimum threshold and x be a solid itemset then* $\forall i \subset x_i/|i| = |x_i - 1|, \exists q/x_p \subseteq q \wedge frequency(i, q) \geq \gamma$.

The proof is straightforward and based on the mono-

tonicity property. Actually, x is a solid itemset and x_i is frequent on x_p. Then, any subset of x_i is frequent on x_p.

The algorithm does not give details about the particular case of generating candidates of size 2. This case is similar to size $n > 2$, but the generated candidates come from the self join $SI_1 \times SI_1$ filtered by the intersection of the periods of each considered items (*i.e.* if two solid itemset of size 1 (a) and (b) do not share a common period, then $(a\ b)$ is not generated). The candidate generation of Algorithm GENERATECANDIDATES is based on the properties of Lemmas 1 and 2

Another special case is not detailed in this algorithm: solid itemsets having a cover of one transaction. In fact, any itemset supported by at least one transaction can be considered as a solid itemset according to definition 5. To avoid the enumeration of all such itemsets, we add a filter on the minimum cover that has to be respected for a solid itemset before it is added to SI_k, the set of solid itemsets of size k in SIM.

Algorithm GENERATECANDIDATES
In: SI_k the set of solid itemsets having length k
Out: C_{k+1} the set of candidates having length $k+1$
$C_{k+1} \leftarrow \emptyset$
Foreach $x, y \in SI_k$ such that:
$$(x_{i_1}, ..., x_{i_{k-1}}) = (y_{i_1}, ..., y_{i_{k-1}})$$
$$\wedge y_{i_k} > x_{i_k} \wedge |x_p \cap y_p| > 1$$
//the periods of x and y have an intersection and
//their prefixes catch the generation criteria
$z = (x_{i_1}, ..., x_{i_{k-1}}, y_k)$
$C_{k+1} \leftarrow C_{k+1} + (z, x_p \cap y_p, \emptyset)$
End for
End algorithm GENERATECANDIDATES

Theorem 1 *At each step of Algorithm* SIM, $SI_k \subseteq C_k$ *(i.e.* $\forall s \in SI_k, \exists c \in C_k / s_x = c_i \wedge s_p \subseteq c_p$*).*

Proof Based on lemma 2 we know that $\forall s \in SI_k, \exists u, v \in SI_{k-1}$ such that:

1. u_x and v_x are prefixes of size $k-1$ of s.
2. u_x and v_x are frequent on u_p and v_p with $s_p \subseteq u_p$ and $s_p \subseteq v_p$.
3. u_x is not frequent on $v_p - u_p$ (since u is a SI and is frequent only on its period u_p).
4. v_x is not frequent on $u_p - v_p$.

Therefore, if we extended each itemset of the solid itemsets in SI_{k-1} with all possible items, and limit their period of possible frequency at the intersections of the corresponding $(k-1)$ solid itemsets, we would be provided with a superset of SI_k. Clearly, Algorithm GENERATECANDIDATES builds candidates on this principle and limits their period of possible frequency to that intersection.

Finally, based on lemma 1 detecting the kernels of C_k, the generated $k-$candidates on the corresponding intersection and merging the obtained kernels, leads to the discovery of the $k-$solid itemsets □

5 Experiments

The goal of this section is to show the points of interest of our approach since the extracted patterns associated to their periods of frequency are the core of a new kind of relevant knowledge and they would not be extracted with a traditional method of itemset extraction. Our dataset comes from the Web access log of Inria Sophia Antipolis from March 2004 to June 2007. It represents 253 Go of rough data. The total number of navigations after the preprocessing is 36,710,616. SIM has been written in C++ on a PC (2.1Ghz) running Linux with 2Go of main memory.

Let us first mention that a behaviour with a cover of, say, 15 navigations within one day may be considered as highly frequent. This is due to the fact the proxies generally hide most navigations from the Web server (the pages are stored in caches of the proxy and requests are most of the time handled by the proxy rather than the server itself). On the other hand, given the characteristics of our data, a cover of 15 navigations would represent a threshold of 4.10^{-5} over three years of records. Our goal is not to extract "frequent" navigations with a minimum threshold $\gamma \approx 0\%$, because that would be of no interest and would lead to a unpracticable number of behaviors (and there is no data mining algorithm able to handle such supports). In fact, thanks to the characteristics of the solid itemsets, we are able to extract patterns that have such a low support while being highly frequent on "regions of interest". This allows decreasing the number of patterns and consuming less CPU. We propose to analyze some of the extracted solid itemsets on the web log of Inria Sophia from 2004 to 2007.

Joan Miro: our first behavior involves a Web page created in 2002. This page has been written by Christophe Berthelot, a member of Omega team at Inria Sophia Antipolis. Here is the corresponding solid itemset:
start: Thu Apr 20 07:05:39 2006
end: Thu Apr 20 17:21:06 2006
frequency: 0.024565
cover: 120
itemset:
with the prefix "omega/personnel/Christophe.Berthelot/"
{ css/style.css,
Omega/JoanMiro/joanmiro.html }

The interpretation of this behavior is not straightforward. First, Christophe is not employed by Inria any more and we

have no contact with him. Second, the web page is from 2002. However, a cover of 120 navigations is exceptionally high (given the amount percentage of requests hidden by the caches of the proxies) and, according to our investigation on that point, the explanation lies in the following informations: 1) This Web page is dedicated to Joan Miro, a famous artist; 2) Joan Miro was born in 1893 April 20; and 3) Christophe's page is ranked fifth on Google with the keywords "Joan Miro" (at the time we write this paper). Our conclusion is that on April 20, (*i.e.* Miro's birthday) people have searched information on the artist and found Christophe's page. This behavior is also discovered in April 2004, 2005 and 2007.

MC2QMC2004 Conference: our second behavior is related to an international conference (MC2QMC2004) organized by Omega (a team at Inria Sophia). Here is the extracted solid itemset:
start: Mon May 17 09:22:41 2004
end: Mon May 17 12:49:26 2004
frequency: 0.0170512
cover: 19
itemset:
{ omega/MC2QMC2004,
omega/MC2QMC2004/monday.html,
omega/MC2QMC2004/tuesday.html }

This behavior may be interpreted as follows: "during the period which begins on Monday May 17 at 09:22:41 and ends on Monday May 17 at 12:49:26 (2004) 1,7% of the users have requested the pages : the index of MC2QMC2004 and the program of MC2QMC2004 for the first day (Monday) and the second day (Tuesday)." After a discussion with the organizers, it appears that a message was widely sent to the community of this conference in order to advertise the program and remind people to register. This was immediately followed by the type of behavior which corresponds to this exact situation.

MedINRIA: our last behavior is related to a software that Asclepios develops and makes available for download.
start: Thu Sep 28 01:53:45 2006
end: Thu Sep 28 04:27:25 2006
support: 0.0148305
cover: 12
itemset:
{ asclepios/style.css, asclepios/software/MedINRIA,
asclepios/software/MedINRIA/download,
asclepios/software/MedINRIA/doc }

Once again, this particular behavior is explained by the corresponding team. In fact, Asclepios has released a new version of MedINRIA in September 2006 and sent a message to the users on September 27 (in the evening). The resulting behavior is a frequent download of the software within the night.

6 Conclusion

In this paper, we have proposed a new definition of itemsets that correspond to a high frequency on a specific period without specifying a time granularity or a particular period. The periods of frequency and the corresponding itemsets have to be discovered by the algorithm based on the only notion of minimum support. However, discovering these itemsets is a true challenge since the periods of frequency and the corresponding itemsets have to be discovered at the same time. Furthermore the number of possible combinations is impracticable and has to be reduced. We provided the theoretical foundation of our approach and our algorithm is based on the discovery of 'kernels' of frequency and their possible aggregations. Our experiments showed that SIM is able to extract the solid itemsets from very large datasets and provide useful and readable results.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, Washington, D.C., USA, 26–28 1993.

[2] J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *SAC '00: Proceedings of the 2000 ACM symposium on Applied computing*, pages 294–300, 2000.

[3] X. Chen and I. Petrounias. Mining temporal features in association rules. In *PKDD '99: Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 295–300, 1999.

[4] P. S. Y. Cheong Fung, Jeffrey Xu Yu and H. Lu. Parameter free bursty events detection in text streams. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 181–192, 2005.

[5] Z. Chong, J. X. Yu, H. Lu, Z. Zhang, and A. Zhou. False-Negative Frequent Items Mining from Data Streams with Bursting. In *DASFAA'05: Database Systems for Advanced Applications*, pages 422–434, 2005.

[6] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering calendar-based temporal association rules. *DKE*, 44(2), 2003.

[7] S.-K. C. Michail Vlachos, Kun-Lung Wu and P. S. Yu. Fast Burst Correlation of Financial Data. In *Knowledge Discovery in Databases: PKDD 2005*, pages 422–434, 2005.

[8] J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE TKDE*, 14(4):750–767, 2002.

[9] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345, 2003.