Discovering Frequent Behaviors: Time is an Essential Element of the Context

Bashar Saleh, Florent Masseglia

Under consideration for publication in Knowledge and Information Systems

Discovering Frequent Behaviors: Time is an Essential Element of the Context

Bashar Saleh and Florent Masseglia

INRIA
AxIS Project-Team
2004 route des lucioles - BP 93
06902 Sophia Antipolis

Abstract. One of the most popular problems in usage mining is the discovery of frequent behaviors. It relies on the extraction of frequent itemsets from usage databases. However, those databases are usually considered as a whole and therefore, itemsets are extracted over the entire set of records. Our claim is that possible subsets, hidden within the structure of the data and containing relevant itemsets, may exist. These subsets, as well as the itemsets they contain, depend on the context. Time is an essential element of the context. The users' intents will differ from one period to another. Behaviors over Christmas will be different from those extracted during the summer. Unfortunately, these periods might be lost because of arbitrary divisions 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. We introduce the definition of solid itemsets, which represent coherent and compact behaviors over specific periods, and we propose SIM, an algorithm for their extraction.

Keywords:

1. Introduction

We do not expect to see a correlation between the sales of guacamole, chips, and antacids if you consider the activity of shops over the whole year. However, there is a correlation between those sales, if you look at a very specific period: the Super Bowl. During Super Bowl XXXV, 8 Million Pounds of guacamole and 14,500 tons of chips were consumed and antacid sales increased by 20 percent the Monday after (Crepeau, 2010; Duncan, 2010).

Received Feb 22, 2010
Revised Jul 27, 2010
Accepted Oct 09, 2010

The problem of association rule mining, intended to extract this kind of correlation, has been defined in (Agrawal, Imielinski and Swami, 1993). The goal is to obtain the frequent associations between the items of the database from a very large set of records. This problem has many applications in marketing, business management or decision analysis. 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 customers. If this number is above a certain threshold (given by the end user) then this itemset is considered to be frequent.

Many algorithms have been proposed for efficiently extracting frequent itemsets (Pasquier, Bastide, Taouil and Lakhal, 1999; Han, Pei and Yin, 2000; Wang, Han and Pei, 2003; Jr., 1998; Toivonen, 1996). 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). Unfortunately, for many real world applications, this definition of frequent itemsets is not appropriate. Possible interesting itemsets might remain undiscovered despite their very specific characteristics. In fact, interesting itemsets are often related to the context in which they occur. The moment during which they can be observed is an essential component of this context. We may consider, for instance, the behaviors of the customers during the Super Bowl. Another example would be the users of an on-line store after a special discount on recordable DVDs and CDs, advertised on TV. We could also consider the adverse drug reports related to a specific drug that appeared after an alert was publicized on that drug. Similarly, the web site of a conference will observe that a 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 with a time-stamp. This condition has already been proposed, for instance in (Ale and Rossi, 2000; Lee, Lin and Chen, 2001). In (Ale and Rossi, 2000), the authors propose 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 (Ale and Rossi, 2000) 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 occurrences of each item.

In this paper, we propose an extension of (Saleh and Masseglia, 2008). Our goal is to extract itemsets that are frequent in a temporally contiguous subset of the database. For instance, navigations on web sites of recordable CDs and DVDs occur randomly and are not correlated 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 decision-makers since they will want to discover this behavior and its specific window of frequency, and finally link it to the context and the TV spot.

This problem could seem similar to the problem of mining itemsets and bursty events in data streams (Chong, Yu, Lu, Zhang and Zhou, 2005; Zhu and Shasha, 2003; Michail Vlachos and Yu, 2005; Gao and Wang, 2009). However, we will prove that our

method provides features that have not been offered 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. In Section 5, we give an overview of existing methods for the temporal aspects of itemset mining. 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 6 gives a synthesis of our experiments leading to the conclusion of Section 7 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 (Agrawal et al., 1993), and numerous algorithms have been proposed in literature to solve it (Lucchese, Orlando and Perego, 2006; Burdick, Calimlim, Flannick, Gehrke and Yiu, 2005; Pasquier et al., 1999; Han et al., 2000; Wang et al., 2003; Jr., 1998; Toivonen, 1996; Palshikar, Kale and Apte, 2007; Lian, Cheung and Yiu, 2007). Definition 1 states the characteristics of frequent itemsets. It is different from the initial or traditional definitions in (Agrawal et al., 1993) since we consider that each item in the database is associated with 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 $X = \{i_1, i_2, ..., i_k\}$ where $k \leq n$ and $\forall j \in [1..k]\ i_j \in \mathcal{I}$. X is called an **itemset** (or a $k-$**itemset**). Let $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ be a set of time units, over which a linear order $<_U$ is defined, where $u_i <_U u_j$ means that u_i occurs before u_j. A **transaction** T is a couple $T = (tid, X)$ where $tid \in \mathcal{U}$ is the transaction identifier and X is the associated itemset. For simplicity of writing, we consider that each transaction identifier is the timestamp of that transaction and no more than one transaction is recorded for each time unit.
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 identifiers of transactions 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 itemsets in D with respect to γ is denoted by $F(D, \gamma) = \{X \in \mathcal{I} : frequency(X, D) \geq \gamma\}$.

Given a set of 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. Each transaction Id is associated with the set of items in this transaction. In order to simplify the illustration, we assume that the transactions of D are sorted by order of date (*i.e.* T_1 occurred before T_2, etc.). Let us consider a minimum frequency $\gamma = \frac{1}{2}$ given by the user. With such a support, the

Date	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 q	(b)
6	**b** k l	(c)
7	**a b c**	(a c)
8	m n o	
9	**a b c**	
10	**a b c**	

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

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 $\frac{1}{2}$.

Our problem is based on the timestamps associated with the records in D and aims to provide itemsets that are frequent on particular periods of time in D. The main issue is to discover these periods and the frequent itemsets they contain. 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 $Tr(P) = \{T : T \subseteq D, P_s \leq T.tid \leq P_e\}$. We define the set of all potential periods over D as PR.

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 with 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) $\forall p_2 \in PR$ such that $x_p \subseteq p_2$ we have either a) or b) or both:
a) frequency$(x_i, p_2) < \gamma$
b) cover(x_i, p_2) =cover(x_i, x_p)
3) $\forall p_2 \in PR$ such that $p_2 \subset x_p$, 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. 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 of this definition 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 with size 3: $x = (\{a\ b\ c\}, [7..10], \frac{3}{4}$ corresponding to the itemset $(a\ b\ c)$ 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 with 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. Therefore, x respects the third condition of definition 5.

Meanwhile, let us consider the following temporal itemset: $z = ((a\ b\ c), [9..10], 100\%)$. We can observe that z_i has the minimum support over z_p. However, there exists a period $p_2 = [7..10]$ where $(a\ b\ c)$ is frequent with a larger cover than the cover of z_i on z_p. Hence, z is not a solid itemset since condition 2 of definition 5 is not respected (z is not maximal).

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.

Definition 6 gives the formal definition of a maximal solid itemset.

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

In other words, if x_i is included in y_i and if the period x_p is included or equal to y_p, then x is not a maximal solid itemset.

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.

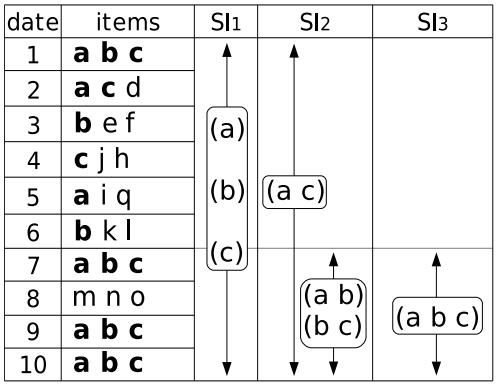


Fig. 2. Solid itemsets in D where $\gamma = \frac{1}{2}$

3. Solid Itemsets and Minimum Threshold: a Discussion

As illustrated in example 2, our problem aims to find itemsets that:

1. do not correspond to the minimum threshold when the entire database is considered,
2. satisfy this threshold over particular periods in the database.

This 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 major 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 the main memory. Even if this set is able to fit in the memory, the response time will be prohibitive. Finally, even if the extracted itemsets fit in the memory and the user is patient enough, the number of rules might be too high.
2. Even with a lower support, if the itemsets are extracted despite their number, they will not be associated with their period of frequency (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). Even if the itemset is frequent over a specific period, the user will not be aware of that, since traditional data mining algorithms are not designed for exhibiting the periods of frequency of the extracted itemsets.

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:

1. 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).
2. Undiscovered behaviors will remain (embedded in the undiscovered periods).
3. 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?).

Generally speaking, tuning the minimum support is a difficult task for users (Zhang, Wu, Zhang and Lu, 2008) and there is a need to go beyond that notion. To conclude this section about the motivation of this work, let us note that the total amount of combinations for enumerating the possible solid itemsets is $(2^n \times k!)$ with n being 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 SIM (Solid Itemset Miner), our algorithm 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. Let us consider y to be a temporal itemset that is not a solid itemset (*i.e.* $y_\sigma < \gamma$). Any superset $z = (z_x, z_p, z_\sigma)$ such that $y_x \subseteq z_x \wedge y_p \subseteq z_p$ of y cannot be a solid itemset (*i.e.* $z_\sigma < \gamma$). 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 itemsets of size k have a common prefix but do not share a common period, they are not considered for generating a new candidate).

However, the counting step (or "pruning" in apriori) is not straightforward in our case. It is not correct to just count the number of occurrences of a candidate over a period. Let us consider $c = ((a\ b), [1..10], c_\sigma)$, a 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. This principle is illustrated in Figure 6 and details are given in Definition 7.

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) requires to discover the kernels "on-the-fly". Intuitively, the kernels of an itemset are the longest periods such that:

1. The first and last records support the itemset.
2. The support of the itemset is always higher than the minimum support when it is counted record after record in the period, starting from the lower timestamp.

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		

Fig. 3. Kernels and period of itemset (b)

Definition 7 gives the formal properties of a kernel and our algorithm, given in Section 4.2 allows for their effective discovery.

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 $x \subseteq Tr(k_s)$ and $Tr(k_s)$ 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 the loss of 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 $m \in N$ such that $\forall n \in N, n > m$ (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 (given in Figure 4 and illustrated in Figure 6) makes it possible to find the solid itemsets of x over P with respect to γ.

Lemma 1. Let K be the set of kernels of x on P with respect to γ. Algorithm MERGEKERNELS makes it possible to find 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$(x, k) > \gamma$. According to Definition 7, x is frequent on each kernel. Furthermore if k is the result of a merging process, then Algorithm MERGEKERNEL checks the frequency of x on the resulting period.
2) $\forall q$ such that $k \subseteq q$, we have one of the following cases:

Algorithm MERGEKERNELS
mergeable ← true;
While (mergeable)
mergeable ← false;
  Foreach ($q \in K$)
    Foreach ($r \in K$ such that $r \neq q \wedge \frac{cover(x,q)|+|cover(x,r)|}{|q \cup r|} \geq \gamma$)
      $K \leftarrow K + q \cup r$;
      $cover(x, q \cup r) = cover(x, q) \cup cover(x, r)$
      mergeable ← true;
      toRemove ← toRemove $+q + r$;
    endFor
  endFor
  Foreach ($k \in$ toRemove) $K \leftarrow K - k$;
  toRemove ← $\emptyset$
End while
End Algorithm MERGEKERNELS

Fig. 4. Algorithm mergeKernels

– $x \in k - q$, 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 makes it possible to know 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,

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

Fig. 5. Algorithm Update

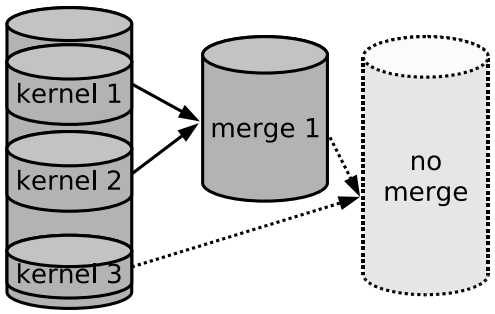


Fig. 6. Merging kernels

the goal of Algorithm UPDATE (Figure 5) 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 SIM (given in Figure 7) aims to generate candidates from size 1 to k.

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
$k \leftarrow 0$;
For each ($i \in \mathcal{I}$)
// 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
$k++$;
$SI_k \leftarrow \emptyset$;
For each ($d \in D$); // scan the database
For each ($c \in C_k$ where $d_{time} \in c.p$)
// The timestamp of d corresponds to the period of c
UPDATE(c,d,γ);
End for
End for
For each ($c \in C_k$)
MERGEKERNELS(c);
For each ($p \in c.kernels$)
$SI_k \leftarrow SI_k + (c_i, p, frequency(c_i, p))$;
End for
$C_{k+1} \leftarrow$GENERATECANDIDATES(SI_k)
While ($C_{k+1} \neq \emptyset$)
And algorithm SIM

Fig. 7. Algorithm Sim

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. 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 $\forall i \subset x_i$ such that $|i| = |x_i - 1|, \exists q$ such that $x_p \subseteq q \wedge frequency(i, q) \geq \gamma$.

The proof is straightforward and based on the monotonicity property. 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 (Figure 8) is based on the properties of Lemmas 1 and 2

Another special case is not detailed in this algorithm: solid itemsets which have 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. In order to avoid the enumeration of all such itemsets, we add a filter on the minimum cover that has to be respected

Algorithm GENERATECANDIDATES
In: SI_k the set of solid itemsets having length k
Out: C_{k+1} the set of candidates having length $k+1$
$C_{k+1} \leftarrow \emptyset$
For each $x, y \in SI_k$ 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

Fig. 8. Algorithm generateCandidates

for a solid itemset before it is added to SI_k, the set of solid itemsets of size k in SIM.

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

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 extend each itemset of the solid itemsets in SI_{k-1} with all possible items, and limit their period of possible frequency to 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 the detection of the kernels of C_k and the generated $k-$candidates on the corresponding intersection, and the merging of the obtained kernels, leads to the discovery of the $k-$solid itemsets □

5. Related Work

Our problem can be compared to two main fields of data mining: mining burst events from data streams and mining temporal itemsets. In this section, we provide an overview of existing methods and problems as well as a comparison with our study.

5.1. Data Streams, Bursts and Sliding Windows

In recent years, burst mining in data streams has gained growing attention. An event is considered bursty if it occurs with strong support in a certain time window. The definitions of bursts may vary in literature, but the idea is generally to find the items that correspond to this time window and a significant threshold. 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 (Cheong Fung and Lu, 2005) with events made of correlations between multiple items).

Let us mention that mining in data streams requires a compromise between the time response and the quality of the result. As a consequence, approximation is a key in data stream mining methods, whereas in our framework, we want to extract the exact set of solid itemsets without compromising on the quality of the result. In fact, when dealing with security issues, such as fraud or intrusion detection, such exhaustive methods can be helpful. For these reasons, our problem cannot be reduced to a data stream mining problem such as frequent itemsets in sliding windows (Gao and Wang, 2009; Chi, Wang, Yu and Muntz, 2006; Teng, Chen and Yu, 2003) or batches (Giannella, Han, Pei, Yan and Yu, 2003). In methods based on sliding windows, the goal is usually to extract the frequent patterns of the current window in reduced time (at least, before the arrival of the next records). In this case, the threshold is computed over the window and corresponds to this specific size. In our definition of the problem, we want to extract the frequent patterns as well as the window size that will allow this frequency. Therefore, the main difference is on the window size which is not pre-defined in our case, which makes the problem challenging.

In (Chong et al., 2005; Michail Vlachos and Yu, 2005; Zhu and Shasha, 2003), we can find methods for mining bursty events in the form of frequent items. In (Zhu and Shasha, 2003) the data stream is a unique sequence of items (or a 2D image) and the authors propose to exploit a wavelet-based method in order to find bursty items. In this case, the items can be photons.

The authors of(Chong et al., 2005) also propose to consider a stream consisting of a single series of items. Their goal is to extract false-negative items with an algorithm (Loss-Negative) which handles bursting.

Fast burst correlation of financial data is considered in (Michail Vlachos and Yu, 2005) and the authors propose to find bursty events (*i.e.* items) in a data stream which consists of m time-series. The authors then propose burst identification in the form of burst intervals and the discovery of overlapping bursts with a query Q.

More similar to our problem, a definition of bursty event detection in text stream can be found in (Cheong Fung and Lu, 2005). In this case the stream consists of text documents. However, the authors propose to use the time information to determine a set of bursty features which may occur in different time windows and to detect bursty events based on the feature distributions with the algorithm HB-Event. The bursty events are made of correlations between features but the main difference with our problem is that overlapping sets of features (for instance $E_1 = \{a, b\},\ E_2 = \{b, c\}$) cannot be found using HB-Event. Furthermore, this method is based on fixed window sizes, whereas our goal is to extract the exact window size that optimizes the threshold of the itemsets.

Let us mention that (Zhu and Shasha, 2003) makes it possible to find the exact size of the time period for a bursty event but in this case, the events consist of items and not itemsets. In (Calders, Dexters and Goethals, 2007), the authors propose to discover the optimal size of a window regarding the frequency of a pattern. This problem is closely related to ours but deals with data streams where the end of the window is fixed (and corresponds to the current point in the stream). Their goal is to find the latest record up to which the window can be extended while optimizing the threshold of the itemset. This is very different from our problem, where we have random access to all the records and can choose the record where the window ends.

Finally, mining itemsets in data streams is frequently linked to a time decaying model. For instance, in (Giannella et al., 2003; Chang and Lee, 2003) the authors pro-

pose to diminish the effect of old transactions on the mining result by decaying the old occurrences of each itemset as time goes by. This can be considered as closely related to our problem since frequent itemsets can be associated with their periods of frequency. However, in time decaying models, if the user wants a minimum support of $\frac{1}{4}$, with a batch or a window of size $10,000$, the minimum number of occurrences should be $2,500$ for an itemset to be extracted. Our claim is that an itemset with a cover of, say, size 3, on a window of size 12 should be extracted. It will not be the case with decaying models or models based on batches having a fixed size (*e.g.* $10,000$), because the division of data is arbitrary and not guided by the frequency of patterns (like we want to do).

To conclude the subject of data streams, bursts and sliding windows, at present and to the best of our knowledge, we can affirm that despite some similarities, there is no method which allows mining frequent itemsets at optimal time granularities in an exhaustive manner (like we propose to do with solid itemsets).

5.2. Temporal itemsets

Interesting studies have been proposed on 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. In (Ale and Rossi, 2000), the authors propose the notion of temporal association rules. Their idea consists of extracting itemsets that are frequent over a specific period that will be shorter than the whole database. However, the periods proposed in (Ale and Rossi, 2000) 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. Furthermore, let us consider the itemset $(a\ b\ c)$ in example 2, this itemset would not be discovered with the definition of periods given in (Ale and Rossi, 2000), since the first occurrence of the involved items is 1 and their last occurrence is 10 (and we know that the itemset $(a\ b\ c)$ has a threshold of $\frac{4}{10}$ on this period). A similar idea is proposed in (Lee et al., 2001), where the authors propose to extract itemsets in a publication database. Their goal is to extract rules in the form $(X \Rightarrow Y)^{t,n}$, where t is the first occurrence of X and Y in the same transaction and n is the end of the whole database.

Both previous studies have opened the door to exploring the temporal aspects of association rules. However, if a data mining paradigm is necessary for extracting the itemsets, this is not the case for the periods they propose since they correspond to a straightforward definition. *In our work, we want to avoid the usage of such a specific time granularity. The different window sizes will be discovered by the algorithm we propose and they will optimize the frequency of the corresponding itemsets.*

2) A specific pattern is given and the goal is to find the corresponding periods. In (Chen and Petrounias, 1999), the authors propose to identify the valid period and periodicity of patterns. In other words, given a specific association rule specified by the user, their goal is to find the valid interval for this rule. In (Yoo, Zhang and Shekhar, 2005), the authors propose to extract time-profiled associations in order to discover interacting relationships consistent with a query prevalence sequence over time.

3) Mining periodic (repetitive) patterns. In this category, the timestamps are analyzed in order to find repetitive patterns, *i.e.* patterns that occur regularly and with a pre-

cise periodicity in the records. In (Li, Ning, Wang and Jajodia, 2003), a repetition model or pattern is given by the user in the form of a calendar. For instance, $< 2000, *, 16 >$ corresponds to the 16^{th} day of a month in 2000. They propose the extraction of association rules during this time interval with two methods: one precise and the other fuzzy. In (Ozden, Ramaswamy and Silberschatz, 1998), the problem of mining cyclic association rules is proposed. A rule is considered cyclic if it has minimal confidence and support at regular time intervals and two algorithms are proposed to solve the problem of their extraction.

We note that an instructive survey on temporal knowledge extraction can be found in (Roddick and Spiliopoulou, 2002).

We have defined a fourth kind of study: given a minimum support, extract all the optimal periods and the corresponding patterns, as stated in definition 6. In (Masseglia, Poncelet, Teisseire and Marascu, 2008) we have proposed a problem related to sequential patterns and period mining in a Web access log file. However, the definition of periods in (Masseglia et al., 2008) is straightforward (a period lasts as long as no user logs in or out). Furthermore, the proposed solution was based on a heuristic whereas in this paper, we propose an exhaustive and exact extraction of such periods and patterns.

5.3. Context-aware division of the data

How to divide data and where to search for interesting knowledge is a subject of growing interest (Palma, Bogorny, Kuijpers and Alvares, 2008; Xiong, Steinbach, Ruslim and Kumar, 2009). Usually, when the context is involved in the division, it enables the extraction of patterns that are relevant and useful.

In (Palma et al., 2008) the context is made of locations on a map that are used to perform a better clustering in a set of trajectories. The authors propose to use a set of geographic objects given by the user to find interesting places (though we are searching for interesting periods, the goal is to obtain a better division of the data). The goal of (Xiong et al., 2009) is to divide the data using a pattern preserving method. The authors claim that interesting patterns might be ignored if the data that support them is divided into separate clusters. They propose clustering solutions that do not split the patterns between different clusters.

6. Experiments

The goal of this section is to show the points of interest of our approach from two main points of view. First, the extracted patterns associated with their periods of frequency are the core of a new kind of relevant knowledge. Second, they would not be extracted with a traditional method of itemset mining. In Subsection 6.1 we will provide some examples of extracted itemsets, and for each itemset we provide its possible interpretation. In Subsection 6.2 we provide some studies on the threshold of the patterns extracted by our method. Our dataset comes from the Web access log of Inria Sophia Antipolis from March 2004 to June 2007 and its size is 253 Gb. 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 2 Gb of main memory.

6.1. Behaviors

The goal of this section is to show some of the most relevant behaviors that we have extracted. For each behavior in this section, we have investigated and found a convincing explanation of the frequency of the pattern and its associated period. That explanation is always related to the context. Let us mention that a behavior with a cover of, say, 15 navigations within one day may be considered as highly frequent. This is due to the fact that proxies generally hide most navigations from the Web server (the pages are stored in caches of the proxy and requests are often handled by the proxy rather than the server itself). Meanwhile, 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 impracticable number of behavior patterns (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 makes it possible to decrease the number of patterns and consume less CPU time.

In this section, 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 percentage of requests hidden by the caches of the proxies) and, according to our investigation on that point, the explanation lies in the following information:

1. This Web page is dedicated to Joan Miro, a famous artist.
2. Joan Miro was born on April 20 1893.
3. Christophe's page is ranked fifth on Google with the keywords "Joan Miro" (at the time of writing this paper).

Our conclusion is that on April 20, (*i.e.* Miro's birthday) people have searched for information about the artist and found Christophe's page. This behavior is also true for 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 users have requested the following 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 this behavior.

MedINRIA: our last behavior is related to a software developed by Asclepios and made available to 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.

6.2. Characteristics of the itemsets

The following experiments have been conducted on a dataset that represent only one month: March 2006. This month is representative of our whole dataset and makes it possible to understand the characteristics of the results obtained with SIM. The dataset for March 2006 contains 1,205,754 navigations and 99,262 items (URLs).

In order to reduce the number of extracted solid itemsets, we proposed to add a parameter to SIM: the minimum cover. In our dataset the number of solid itemsets having a cover of 2 is very high (approximately 80,000 for March 2006). Furthermore, the corresponding solid itemsets are not very informative. This is why we decided to provide SIM with a filter on the minimum cover and therefore make it possible to lower the number of irrelevant itemsets. In order to study the impact of the minimum cover on the number of solid itemsets we have reported in Figure 9 the number of solid itemsets corresponding to each possible cover that has a size greater than three for March 2006. The information is provided for five different minimum thresholds of solid itemsets: 5%, 4%, 3%, 2% and 1.5%. The results of Figure 9 show, for instance, that we have extracted ten solid itemsets with a cover of six and a minimum support of 3%. The possible covers range from 4 to 18 (however, in order for the graphic to be readable,

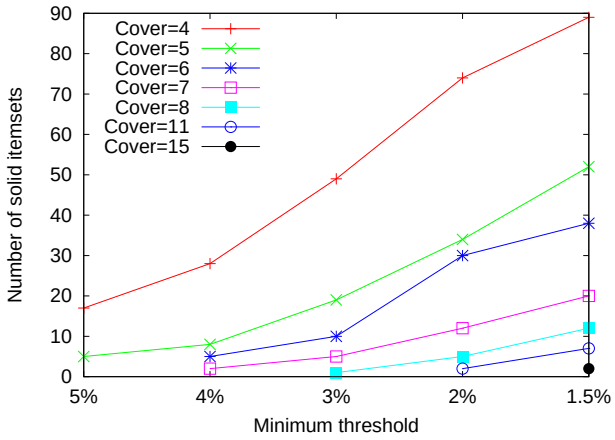


Fig. 9. Number of solid itemsets for each possible cover.

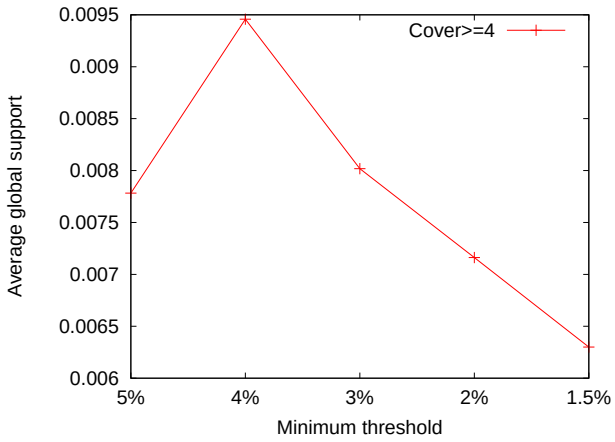


Fig. 10. Average global supports for a minimum cover of 4.

we only report the number of SI for a selected set of covers). The minimum number of solid itemsets with a specific cover is 1 (*i.e.* cover 8 and support 3%) and the maximum number of solid itemsets having a specific cover is 89 (cover 4 and support $1,5\%$).

It is difficult to compare SIM with traditional algorithms since our itemsets have very particular thresholds. Hence, we propose to compute the average threshold of the solid itemsets over the whole dataset. Let us consider, for instance, a solid itemset x. Our goal is to scan the dataset and find the exact support of x_i over the entire dataset. This operation is performed for all the solid itemsets and the average global support is computed. This global support is reported in Figure 10. We can observe, for instance, that the average global support of the solid itemsets extracted with $\gamma = 2\%$ is 0.07%. The average global supports are very low and this is a well known issue for traditional data mining algorithms. However, even in the case of a possible extraction by means of a traditional data mining method with such supports, the following is true:

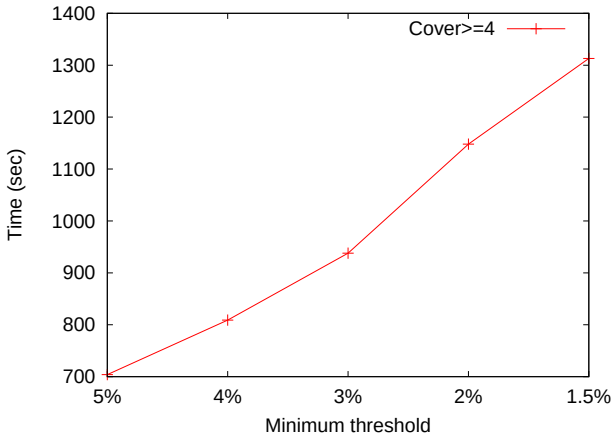


Fig. 11. Execution times for a minimum cover of 4.

1. The periods of interest would not be exhibited.
2. Itemsets with this support would be extracted over the whole dataset, even if they don't correspond to a period of interest (in other words, the number of itemsets would be very large and would not have the interesting characteristics of the solid itemsets).

6.3. Execution Times and Scalability

Our last experiment is dedicated to studying the response times of SIM. In Figure 11 we have reported the response times of SIM with different thresholds and a minimum cover of four, on the log file corresponding to March 2006. The first observation is that the execution time of SIM still needs to be improved. Presently, however, this kind of knowledge can only be extracted using SIM. Second, we demonstrate the scalability of our algorithm in Figure 12. We can see that, for three different values with minimum support, our algorithm has linear response times regarding the data size. For this experiment, we built a synthetic data set by selecting subsets of the original file. The size of these subsets grows from 300,000 navigations up to 1,200,000 navigations (which corresponds to the whole file of March 2006).

7. Conclusion

In this paper, we have proposed a new definition of itemsets that present a high frequency over a specific period without specifying a time granularity or a particular period. The periods of frequency and the corresponding itemsets have to be determined 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

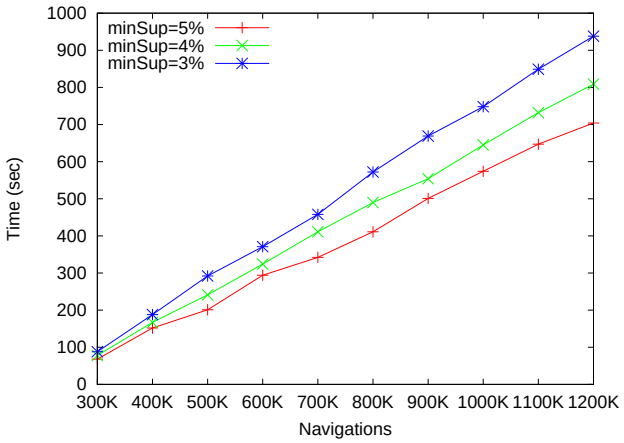


Fig. 12. Execution times with an increasing number of navigations.

to extract the solid itemsets from very large datasets and provides useful and readable results.

Since our method is intended to discover periods of frequency, we have observed that this extraction might be divided into subprocesses. Our goal is now to study possible divisions of the data that will ensure a safe extraction of solid itemsets. In other words, instead of an arbitrary division that might touch a period of frequency, we will study a division driven by the data mining algorithm on the basis of a first scan. That would be a first step towards a parallel version of this algorithm.

References

Agrawal, R., Imielinski, T. and Swami, A. N. (1993), Mining association rules between sets of items in large databases, *in* 'SIGMOD', Washington, D.C., USA, pp. 207–216.

Ale, J. M. and Rossi, G. H. (2000), An approach to discovering temporal association rules, *in* 'SAC '00: Proceedings of the 2000 ACM symposium on Applied computing', pp. 294–300.

Burdick, D., Calimlim, M., Flannick, J., Gehrke, J. and Yiu, T. (2005), 'Mafia: A maximal frequent itemset algorithm', *IEEE Transactions on Knowledge and Data Engineering* **17**(11).

Calders, T., Dexters, N. and Goethals, B. (2007), Mining frequent itemsets in a stream, *in* 'ICDM', pp. 83–92.

Chang, J. H. and Lee, W. S. (2003), Finding recent frequent itemsets adaptively over online data streams, *in* 'KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining', pp. 487–492.

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

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

Chi, Y., Wang, H., Yu, P. S. and Muntz, R. R. (2006), 'Catch the moment: maintaining closed frequent itemsets over a data stream sliding window', *Knowl. Inf. Syst.* **10**(3), 265–294.

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

Crepeau, R. C. (2010), 'The economics of super bowl xliii'. [Online], http://www.poppolitics.com/archives/2009/01/the-economics-of-super-bowl-xliii.

Duncan, A. (2010), 'Super bowl xxxv fun facts'. [Online], http://advertising.about.com/od/superbowlcoverage/a/xxxvfunfacts.htm.

Gao, C. and Wang, J. (2009), Efficient itemset generator discovery over a stream sliding window, *in* 'CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management', ACM, New York, NY, USA, pp. 355–364.

Giannella, C., Han, J., Pei, J., Yan, X. and Yu, P. (2003), *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*, In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining, AAAI/MIT.

Han, J., Pei, J. and Yin, Y. (2000), Mining frequent patterns without candidate generation, *in* 'SIGMOD', pp. 1–12.

Jr., R. J. B. (1998), Efficiently mining long patterns from databases., *in* 'SIGMOD, June 2-4, Seattle, Washington, USA.', pp. 85–93.

Lee, C.-H., Lin, C.-R. and Chen, M.-S. (2001), On mining general temporal association rules in a publication database., *in* 'ICDM 29 November - 2 December 2001, San Jose, California, USA', pp. 337–344.

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

Lian, W., Cheung, D. W. and Yiu, S. M. (2007), Maintenance of maximal frequent itemsets in large databases, *in* 'SAC', pp. 388–392.

Lucchese, C., Orlando, S. and Perego, R. (2006), 'Fast and memory efficient mining of frequent closed itemsets', *IEEE Transactions on Knowledge and Data Engineering* **18**(1), 21–36.

Masseglia, F., Poncelet, P., Teisseire, M. and Marascu, A. (2008), 'Web usage mining: extracting unexpected periods from web logs', *Data Min. Knowl. Discov.* **16**(1), 39–65.

Michail Vlachos, Kun-Lung Wu, S.-K. C. and Yu, P. S. (2005), Fast Burst Correlation of Financial Data, *in* 'Knowledge Discovery in Databases: PKDD 2005', pp. 422–434.

Ozden, B., Ramaswamy, S. and Silberschatz, A. (1998), Cyclic association rules, *in* 'ICDE, Orlando, Florida, USA', pp. 412–421.

Palma, A. T., Bogorny, V., Kuijpers, B. and Alvares, L. O. (2008), A clustering-based approach for discovering interesting places in trajectories, *in* 'SAC', pp. 863–868.

Palshikar, G. K., Kale, M. S. and Apte, M. M. (2007), 'Association rules mining using heavy itemsets', *Data Knowl. Eng.* **61**(1), 93–113.

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (1999), Discovering frequent closed itemsets for association rules., *in* 'ICDT', pp. 398–416.

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

Saleh, B. and Masseglia, F. (2008), Time aware mining of itemsets, *in* 'TIME', pp. 93–97.

Teng, W.-G., Chen, M.-S. and Yu, P. S. (2003), A Regression-Based Temporal Pattern Mining Scheme for Data Streams, *in* 'VLDB', pp. 93–104.

Toivonen, H. (1996), Sampling large databases for association rules., *in* 'VLDB, September 3-6, Mumbai (Bombay)', pp. 134–145.

Wang, J., Han, J. and Pei, J. (2003), Closet+: searching for the best strategies for mining frequent closed itemsets., *in* 'KDD, Washington, DC, USA, August 24 - 27', pp. 236–245.

Xiong, H., Steinbach, M., Ruslim, A. and Kumar, V. (2009), 'Characterizing pattern preserving clustering', *Knowl. Inf. Syst.* **19**(3), 311–336.

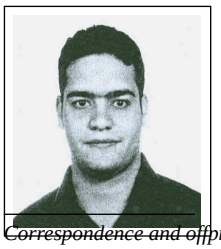
Yoo, J. S., Zhang, P. and Shekhar, S. (2005), Mining time-profiled associations: An extended abstract., *in* 'PAKDD, Hanoi, Vietnam, May 18-20', pp. 136–142.

Zhang, S., Wu, X., Zhang, C. and Lu, J. (2008), 'Computing the minimum-support for mining frequent patterns', *Knowl. Inf. Syst.* **15**(2), 233–257.

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

Author Biographies

Florent Masseglia is currently a researcher for INRIA (Sophia Antipolis, France). He did research work in the Data Mining Group at the LIRMM (Montpellier , France) from 1998 to 2002 and received a Ph.D. in computer science from Versailles University , France in 2002. His research interests include data mining (particularly sequential patterns and applications such as Web Usage Mining), data streams and databases. He has co-chaired the 6th and 7th editions of the international workshop on âĂIJMultimedia Data MiningâĂİ in conjunction with the KDD conference. He is one of the guest editors of two special issues on this topic in the IEEE TMM and the MTAP journal. Florent Masseglia is co-editor of two books on data mining ("Data Mining Patterns: New Methods and Applications" and "Successes and New Directions in Data Mining"). He is reviewer for a dozen of major international journals.



Bashar Saleh was born and raised in Syria. He graduated from Damascus University's in software engineering and information systems. He continued his education in France, obtained an MSc in computer science from the University of Nice. He worked like a research engineer in the domain of personalization and recommendation systems. After that he started a self enterprise specialized in web development. His actual interests spread over two domains; web development (frameworks, ecommerce, personalization) and information system's security.

Correspondence and offprint requests to: Florent Masseglia, INRIA, 2004 route des Lucioles - BP 93, 06902 Sophia Antipolis, FRANCE. Email: florent.masseglia@inria.fr