Seventh IEEE International Conference on Data Mining

Mining Frequent Itemsets in a Stream

Toon Calders
Eindhoven University of Technology

Nele Dexters
University of Antwerp

Bart Goethals
University of Antwerp

Abstract

We study the problem of finding frequent itemsets in a continuous stream of transactions. The current frequency of an itemset in a stream is defined as its maximal frequency over all possible windows in the stream from any point in the past until the current state that satisfy a minimal length constraint. Properties of this new measure are studied and an incremental algorithm that allows, at any time, to immediately produce the current frequencies of all frequent itemsets is proposed. Experimental and theoretical analysis show that the space requirements for the algorithm are extremely small for many realistic data distributions.

1. Introduction

Mining frequent sets over streams of itemsets presents interesting new challenges over traditional mining in static databases. Due to the speed of new arriving data, it is assumed that the history of the stream can not be revisited, unless it is stored. Storing large parts of a stream, however, is impossible as the amount of data is typically huge.

Most previous work on mining frequently occurring itemsets over data streams either focusses on (1) the sliding window model, (2) the time-fading model, or (3) the landmark model. Each of these models requires a fixed window length or decay factor, given by the user. In many applications, however, choosing such parameters that are most appropriate for every itemset at every timepoint in an evolving stream is almost impossible. For example, consider a large retail chain of which sales can be considered as a stream. Then, in order to find frequent sets to do market basket analysis, it is very difficult to choose in which period of the collected data you are interested. For many products, the amount of them sold depends highly on the period of the year. In summer time, e.g., sales of ice cream increase and during the soccer world cup, sales of beer increase. Such seasonal behavior of a specific item or combination of items can only be discovered when choosing the correct window size for that item(set). This size, however, can hide a similar behavior of other item(set)s in another window.

Therefore, we propose to consider for each itemset the window in which it has the highest frequency. More specifically, we define the current frequency of an itemset as the maximum over all windows from the past until the current state that satisfy a minimal size constraint. Notice that this is an extension of the max-frequency measure defined before for items [1]. Hence, when the stream evolves, the length of the window containing the highest frequency for a given itemset can change continuously. This new stream measure turns out to be very suitable to early detect sudden bursts of occurrences of itemsets, while still taking into account the history of the itemset. This behavior might be particularly useful in applications where hot topics, or popular combinations of topics need to be tracked. Examples of such applications include, e.g., identifying stocks with a strong growth or tracking popular search terms on the internet. In these applications it is of vital importance to identify sudden bursts quickly, while still taking into account the history.

Concretely, our contributions are the following. First, (1) the max-frequency measure [1] is extended to itemsets and minimal window length, and (2) a detailed study of its behavior is performed, taking into account minimal window length and minimal frequency thresholds, resulting in several important properties. (3) An efficient algorithm for computing the exact frequencies for all frequent itemsets at any time is proposed; this in contrast to the often only approximate algorithms for other methods. Finally, (4) a theoretical and empirical evaluation of our proposed method is given.

The organization of the paper is as follows. In Section 2, the new measure is defined and the central problem statement is formally introduced. Section 3 gives several properties of the max-frequency and states the main theorem, on which the incremental algorithm in Section 4 is based. In Section 5, a theoretical analysis for the worst case is done. Experimental results in Section 6 show that the memory requirements for the algorithm are extremely small for many real-life data distributions. In Section 7, the relation between our measure and existing related work is explored, and Section 8 concludes the paper.

1550-4786/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDM.2007.66

2. Problem Statement

2.1. Streams and Max-Frequency

A *stream* $\langle I_1 \ I_2 \ \ldots \ I_n \rangle$ is a sequence of itemsets, denoted $\mathbb{S}$, where $n = |\mathbb{S}|$ is the *length* of the stream. I_1 is considered the first and oldest itemset in the stream, and I_n the latest and most recent. We assume that the items in the stream come from a finite set of items $\mathcal{I}$.

The number of sets in a stream $\mathbb{S}$ that contain itemset I is denoted $count(I, \mathbb{S})$. For example, $count(a, \langle ab \ c \ adf \rangle) = 2$ and $count(af, \langle ab \ c \ adf \rangle) = 1$. The *frequency of I in* $\mathbb{S}$ is defined as

$$freq(I, \mathbb{S}) := \frac{count(I, \mathbb{S})}{|\mathbb{S}|} \ .$$

For example, $freq(a, \langle ab \ c \ adf \rangle) = 2/3$ and $freq(af, \langle ab \ c \ adf \rangle) = 1/3$.

Let $\mathbb{S}_1$ be $\langle I_1^1 \ldots I_{n_1}^1 \rangle$, $\mathbb{S}_2$ be $\langle I_1^2 \ldots I_{n_2}^2 \rangle$, $\ldots$ and $\mathbb{S}_m$ be $\langle I_1^m \ldots I_{n_m}^m \rangle$. The *concatenation* of the streams $\mathbb{S}_1, \ldots, \mathbb{S}_m$, denoted $\mathbb{S}_1 \cdot \mathbb{S}_2 \cdot \ldots \cdot \mathbb{S}_m$, is

$$\langle \ I_1^1 \ \ldots \ I_{n_1}^1 \ I_1^2 \ \ldots \ I_{n_2}^2 \ldots \ I_1^m \ \ldots \ I_{n_m}^m \ \rangle \ .$$

Let $\mathbb{S} = \langle I_1 \ I_2 \ \ldots \ I_n \rangle$. Then, $\mathbb{S}[s, t]$ denotes the *substream* or *window* $\langle I_s \ I_{s+1} \ \ldots \ I_t \rangle$. The sub-stream of $\mathbb{S}$ consisting of the last k items of $\mathbb{S}$, denoted $last(k, \mathbb{S})$, is

$$last(k, \mathbb{S}) := \mathbb{S}\big[|\mathbb{S}| - k + 1, |\mathbb{S}|\big] \ .$$

We are now ready to define our new frequency measure:

Definition 1 *Given a minimal window size mwl, the* max-frequency $mfreq^{mwl}(I, \mathbb{S})$ *of itemset I in a stream $\mathbb{S}$ is defined as the maximum of the frequencies of I over all windows, of size at least mwl, extending from the end of the stream; that is:*

$$mfreq^{mwl}(I, \mathbb{S}) \ := \max_{k = mwl, \ldots, |\mathbb{S}|} \left(freq(I, last(k, \mathbb{S})) \right) \ .$$

If the length of the stream is less than mwl, the max-frequency is defined to be 0.

The longest window in which the maximum frequency is reached is called the maximal window for I in $\mathbb{S}$*, and its starting point is denoted $startmax^{mwl}(I, \mathbb{S})$. That is, $startmax^{mwl}(I, \mathbb{S})$ is the smallest index such that*

$$mfreq^{mwl}(I, \mathbb{S}) \ = \ freq(I, \mathbb{S}\big[startmax^{mwl}(I, \mathbb{S}), |\mathbb{S}|\big]) \ .$$

mwl wil be omitted when clear from the context.

Example 1 *Let* $mwl = 3$.

$$\begin{aligned} mfreq^{mwl}(a, \langle a \ b \ a \ a \ a \ b \rangle) &= 3/4 \ . \\ mfreq^{mwl}(a, \langle b \ c \ d \ a \ b \ c \ d \ a \rangle) &= 2/5 \ . \end{aligned}$$

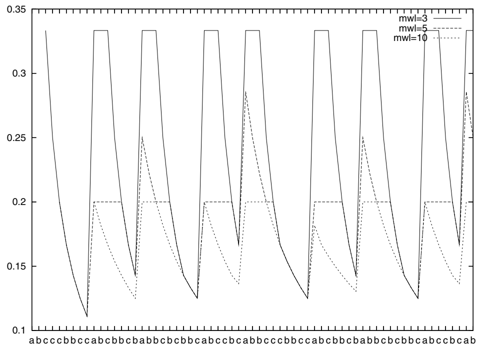


Figure 1. Max-frequency for minimal window lengths 1, 3, and 10.

In the definition of the max-frequency, an explicit lower bound is given on the size of the windows in which the frequencies are considered. This lower bound is given to relieve the undesirable effect of having a frequency of 100% in a window of length 1, every time the target item arrives in the stream. The effect of the minimal window length mwl is illustrated in Figure 1. It is clear that for longer minimal window lengths, there are still jumps in the frequency, but they are less pronounced. Hence, setting an appropriate minimal window length effectively resolves the instability of the max-frequency measure.

2.2. Evolving Streams

A stream was defined as a statical object. In reality, however, a stream is an evolving object that is essentially unbounded. When processing a stream, it is to be assumed that only a small part of it can be kept in memory.

$\mathbb{S}_t$ will denote the stream $\mathbb{S}$ up to timestamp t; that is, the part of the stream that already passed at time t, $\mathbb{S}_t = \mathbb{S}[1, t]$. For simplicity, we assume that the first itemset arrives at timestamp 1, and since then, at every timestamp a new itemset is inserted into the stream.

The main problem we study in this paper is the following: *Given a minimal frequency threshold and a minimal window length, for an evolving stream $\mathbb{S}$, maintain a small summary of the stream in time, such that, at any timepoint t, all current frequent itemsets can be produced instantly from this summary.* More formally, we will introduce a concise summary, $summary(\mathbb{S}_t)$, and efficient procedures $Update$, and Get_mfreq, such that $Update(summary(\mathbb{S}_t), I)$ equals $summary(\mathbb{S}_t \cdot \langle I \rangle)$, and $Get\_mfreq(summary(\mathbb{S}_{t+1}))$ equals $mfreq^{mwl}(A, \mathbb{S}_{t+1})$.

Because $Update$ has to be executed every time a new itemset arrives, it has to be extremely efficient in order to be finished before the next itemset arrives. Similarly, because the stream continuously grows, the summary must be independent of the number of items seen so far, or, at least grow very slowly as the stream evolves. The method we develop will indeed meet these criteria, as the theoretical analysis in Section 5, and the experiments in Section 6 show.

For ease of presentation, we present our solution in a modular way; first we present how a summary can be maintained that allows for *one* itemset A, to produce its max-frequency at any point in time, for the case *no* minimal window length has been set. Notice that no minimal window length actually corresponds to having a minimal window length of 1. We denote the max-frequency of A in $\mathbb{S}$ without minimal window length simply as $mfreq^{\mathbf{1}}(A, \mathbb{S})$. Then, we extend the method to work with minimal window length and minimal frequency, but still for only one target itemset A. Finally, we show how to combine everything into one solution for mining all frequent itemsets at once, without having to maintain a separate summary for every itemset.

3. Properties of Max-Frequency

In this section, we show some properties of max-frequency for one itemset A without a minimal window length constraint. These properties will be crucial for the incremental algorithm that maintains the summary of the stream for A.

Obviously, checking all possible windows to find the maximal one is infeasible algorithmically, given the constraints of stream problems. Fortunately, not every point in the stream needs to be checked. The theoretical results from this section show exactly which points need to be inspected. These points will be called the *borders* in the stream. The summary of the stream will consist exactly of the recording of these borders, and the corresponding frequency of the target itemset.

Definition 2 *Timestamp q is called a* border for set A in $\mathbb{S}$ *if there exists a stream $\mathbb{B}$ such that $q = startmax(A, \mathbb{S} \cdot \mathbb{B})$.*

Thus, a border is a point in the stream that can still become the starting point of the maximal window. Based on the next theorem, it is possible to give an exact syntactic characterization of the borders.

Theorem 1 *Let $\mathbb{S}$ be a stream of length L, and let $\mathbb{S}[q, L]$ be the maximal window for the itemset A. Then, for any p, r with $p < q \leq r$: $freq(A, \mathbb{S}[p, q-1]) < freq(A, \mathbb{S}[q, r])$.*

Proof 1 *Let $\mathbb{B}_1$ denote $\mathbb{S}[p, q-1]$, $\mathbb{B}_2$ denote $\mathbb{S}[q, r]$, and $\mathbb{B}_3$ denote $\mathbb{S}[r+1, L]$. Because $\mathbb{B}_2 \cdot \mathbb{B}_3$ is the maximal window for A in $\mathbb{S}$, it holds that the frequency of A in $\mathbb{B}_2 \cdot \mathbb{B}_3$ is strictly higher than in $\mathbb{B}_1 \cdot \mathbb{B}_2 \cdot \mathbb{B}_3$ and it is at least as high as in $\mathbb{B}_3$ (remember that in the case of multiple windows with maximal frequency the largest one is selected). Now, let $l_1 = |\mathbb{B}_1|$, $l_2 = |\mathbb{B}_2|$, and $l_3 = |\mathbb{B}_3|$, and let $a_1 = count(A, \mathbb{B}_1)$, $a_2 = count(A, \mathbb{B}_2)$, and $a_3 = count(A, \mathbb{B}_3)$, as depicted in:*

$$\boxed{} \;\; \overbrace{\boxed{a_1/l_1}}^{\mathbb{B}_1} \;\Big|\; \overbrace{\boxed{a_2/l_2}}^{\mathbb{B}_2} \;\; \overbrace{\boxed{a_3/l_3}}^{\mathbb{B}_3} \quad .$$

Then, the conditions on the frequency translate into:

$$\frac{a_2 + a_3}{l_2 + l_3} > \frac{a_1 + a_2 + a_3}{l_1 + l_2 + l_3} \quad \textit{and} \quad \frac{a_2 + a_3}{l_2 + l_3} \geq \frac{a_3}{l_3}.$$

From these conditions, it can be derived that

$$freq(A, \mathbb{B}_1) = \frac{a_1}{l_1} < \frac{a_2}{l_2} = freq(A, \mathbb{B}_2) \ .$$

Corollary 1 *Let $\mathbb{S}$ be a stream of length L, and let $1 \leq q \leq L$. Position q is a border for target itemset A in $\mathbb{S}$ if and only if for all indices j, k with $1 \leq j < q$ and $q \leq k \leq L$, it holds that $freq(A, \mathbb{S}[j, q-1]) < freq(A, \mathbb{S}[q, k])$.*

Proof 2 ***Only if:*** *Follows directly from Theorem 1.*

If: *We need to show that there exists a continuation $\mathbb{S}'$ of stream $\mathbb{S}$ (resulting in stream $\mathbb{S} \cdot \mathbb{S}'$) in which q is the starting point of the maximal window. We consider two cases: either q is the rightmost border in $\mathbb{S}$, or not. If q is the rightmost border, then q is the maximal border in $\mathbb{S}$, because for any other border $p < q$, $freq(A, \mathbb{S}[p, q-1]) < freq(A, \mathbb{S}[q, L])$ which implies $freq(A, \mathbb{S}[p, L]) < freq(A, \mathbb{S}[q, L])$, and hence the Corollary holds.*

In the other case, we will show that it is always possible to continue $\mathbb{S}$ in such a way that the rightmost border disappears, while all other borders remain and no new borders are introduced. By consecutively applying this procedure, any border will eventually become the rightmost border at one point, and hence become the starting point of the maximal window.

Let $q < q'$ be the two largest borders in $\mathbb{S}$. Since, because of the only-if part of this theorem,

$$\begin{aligned} freq(A, \mathbb{S}[q, q'-1]) &\leq freq(A, \mathbb{S}[q', L]) \\ &= \frac{count(A, \mathbb{S}[q', L])}{L - q' + 1} \end{aligned} \ ,$$

we can always find positive integers $x \leq y$ such that:

$$freq(A, \mathbb{S}[q, q'-1]) = \frac{count(A, \mathbb{S}[q', L]) + x}{L - q' + 1 + y} \ .$$

Then, the following continuation of $\mathbb{S}$ has exactly the same borders as $\mathbb{S}$, except from q', which is no longer a border:

$$\mathbb{S} \cdot \langle \overbrace{A\, A\, \cdots\, A}^{x \times} \overbrace{\emptyset\, \emptyset\, \cdots\, \emptyset}^{y - x \times} \rangle \ .$$

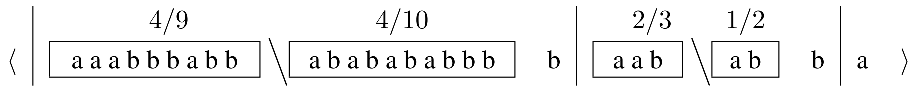


Figure 2. Example of dropping borders.

Example 2 *Assume we have the stream $\mathbb{S}_{27}$, given in Figure 2 and we focus on target $\{a\}$. In this stream, two positions have been marked with a backslash. Both these points do* not *meet the criteria to be a border given in Corollary 1. Indeed, for both positions, a block before and after it is indicated such that the frequency in the before-block is higher than in the after-block. The only positions that do meet the requirement are indicated by vertical bars.*

4. Algorithm

Based on the results of Section 3, we present an incremental algorithm to efficiently maintain the summary for one itemset A allowing us to produce the current max-frequency (without minimal window length constraint) of an itemset instantly at any time.

4.1. The Summary

Let $p_1 < p_2 < \ldots < p_r$ be the border positions for itemset A in the stream $\mathbb{S}_t$, ordered from oldest to most recent. Let $a_i = \mathit{count}(A, \mathbb{S}_t[p_i, p_{i+1} - 1])$ be the number of occurrences of the target itemset A in between two subsequent border positions p_i and p_{i+1} (for $i = 1, \ldots, r-1$). $a_r = \mathit{count}(A, \mathbb{S}_t[p_r, t])$ denotes the number of occurrences of A since the last border. The summary S_t of $\mathbb{S}_t$ is defined as the array

$$S_t = \begin{array}{|c|c|c|} \hline p_1 & \cdots & p_r \\ \hline a_1 & \cdots & a_r \\ \hline \end{array} \ .$$

We can easily compute the frequencies of itemset A for any of the border positions from this summary:

$$\mathit{freq}(A, \mathbb{S}_t[p_i, t]) \ = \ \frac{\sum_{j=i}^{r} a_j}{t - p_i + 1} \ .$$

Example 3 *The summary S_{17} for target a in stream*

$$\mathbb{S}_{17} = \langle b \ a \ a \ a \ b \ a \ a \ b \ a \ b \ b \ a \ a \ a \ a \ b \ a \rangle :$$

$$S_{17} = \begin{array}{|c|c|c|} \hline 2 & 12 & 17 \\ \hline 6 & 4 & 1 \\ \hline \end{array} \ .$$

We can find the frequencies of itemset $\{a\}$ since any of the border positions:

$$\begin{aligned} \mathit{freq}(a, \mathbb{S}_t[2, 17]) &= 11/16 \ , \\ \mathit{freq}(a, \mathbb{S}_t[12, 17]) &= 5/6 \ , \\ \mathit{freq}(a, \mathbb{S}_t[17, 17]) &= 1/1 \ . \end{aligned}$$

We now give some properties of the summary that will be used by the algorithm. First of all, we show that the fractions in the blocks in between two subsequent border positions are increasing, and as a consequence, among all borders p_i, we have that $\mathit{freq}(A, \mathbb{S}_t[p_i, t])$ is maximal for i equal to r.

Property 1 *Let $\mathbb{S}_t$ be a stream and $\mathit{summary}(\mathbb{S}_t) = [(p_1, a_1), \ldots, (p_r, a_r)]$. Then,*

$$\frac{a_1}{p_2 - p_1} < \frac{a_2}{p_3 - p_2} < \ldots < \frac{a_{r-1}}{p_r - p_{r-1}} < \frac{a_r}{t - p_r + 1}$$

and

$$\mathit{freq}(A, \mathbb{S}_t[p_1, t]) < \mathit{freq}(A, \mathbb{S}_t[p_2, t]) < \ldots < \mathit{freq}(A, \mathbb{S}_t[p_r, t]) \ .$$

(The proof is a direct consequence of Corollary 1).

From this property, it follows directly that the last entry of a summary always represents the max-frequency.

On every timestamp a new itemset arrives and the summary needs to be updated. Algorithm 1 presents the pseudo-code of the algorithm. First the summary is initialized after the first target itemset entered the stream. Then, we consider the following cases.

1. A superset of the target itemset arrives in the stream (lines 6–10):

(a) (lines 7–8) If the frequency of the last block is 1, and hence, the previous itemset in the stream also contained the target itemset, then we need to increment its number of occurrences in the last entry of the summary. Otherwise, (lines 9–10) a new border $(t+1, 1)$ needs to be added as the frequency in this last window of size 1 is 1, and hence, it is larger than the previous max-frequency.

(b) None of the existing borders can be removed from the summary.

2. An itemset not containing the target itemset arrives in the stream (lines 11–20):

(a) No new borders need to be added to the summary.

(b) This is the only case in which borders can actually be removed from the summary. Therefore, according to Corollary 1, we have to compare the frequencies of every two blocks adjacent to a border. That

Algorithm 1 *Update*(S_t, I) for target itemset A on time $t+1$

```
Require: $S_t = summary(\mathbb{S}_t) = [(p_1, a_1), \cdots, (p_r, a_r)]$
Ensure: $S_{t+1} = summary(\mathbb{S}_{t+1}) = summary(\mathbb{S}_t \cdot \langle I \rangle)$
 1: Set $S_{t+1}$ := [ ]
 2: if ($S_t$ is empty) then
 3:     if (target itemset $A \subseteq I$) then
 4:         $S_{t+1} := [(t+1, 1)]$
 5: else
 6:     if (target itemset $A \subseteq I$) then
 7:         if $a_r = t - p_r + 1$ then
 8:             $S_{t+1} := [(p_1, a_1), \cdots, (p_r, a_r + 1)]$
 9:         else
10:             $S_{t+1} := [(p_1, a_1), \cdots, (p_r, a_r), (t+1, 1)]$
11:     else
12:         $S_{t+1} := S_t$
13:         $i := r$
14:         while $i > 1$ do
15:             if $\frac{a_i}{t-p_i+1} \leq \frac{a_i + a_{i-1}}{t-p_{i-1}+1}$ then
16:                 $a_{i-1} := a_{i-1} + a_i$
17:                 remove $(p_i, a_i)$ from $S_{t+1}$
18:                 $i := i - 1$
19:             else
20:                 $i := 1$
```

is, to drop border p, we have to find a before-block and an after-block such that the before-block has a higher frequency than the after-block. Obviously, the before-block with the highest frequency is exactly the block represented by the border before p. Indeed, at timestamp p, that border represented the maximum window according to Property 1. Then, we only have to compare this frequency with the frequency from p until the current timestamp t. Indeed, any other after-block with a lower frequency would have caused the border to have been removed earlier. Furthermore, we do not have to consider every border for removal independently, but, as stated in the following property, only the most recent borders need to be considered for removal. In other words, if a border can not be removed, then all earlier borders can not be removed either, and hence, we must only consider the removal of borders from right to left, until one can not be removed (lines 12–15).

Example 4 *The working of the algorithm is explained in detail for the following stream*

$$\mathbb{S}_{17} = \langle b\ a\ a\ a\ b\ a\ a\ b\ a\ b\ b\ a\ a\ a\ a\ b\ a \rangle$$

and target itemset $\{a\}$. *In Figure 3, a sample run of the algorithm is illustrated for each timepoint.*

In this example, some interesting things happen. First of all, the stream starts with an itemset, $\{b\}$, *that does not contain the target itemset* $\{a\}$. *Therefore,* $Update(S_0, \{b\}) = Update([\ ], \{b\})$ *at timestamp* 1 *remains empty, i.e.,* $S_1 = [\ ]$. *At timestamp* 2, $Update([\ ], \{a\})$ *results in* $S_2 = [(2, 1)]$, *corresponding to the stream* $\langle b\ |a \rangle$ *with a border at position* 2 *and the corresponding frequency* $1/(2-2+1) = 1/1$. *At timestamp* 8, *something interesting happens.* $S_7 = [(2, 3), (6, 2)]$, *corresponding with stream* $\langle b\ |a\ a\ a\ b\ |a\ a \rangle$. $Update(S_7, \{b\})$ *will yield* $S_8 = [(2, 5)]$, *and not* $[(2, 3), (6, 2)]$. *Because the corresponding frequencies decrease from the border at position* 2 *to the border at position* 6, *namely* $(3+2)/[(6-2)+(8-6+1)] = 5/7 > 2/(8-6+1) = 2/3$, *we can conclude that position* 6 *is no longer a border. This is reflected in summary* $S_8 = [(2, 3+2)]$ *and can be visualised by* $\langle b\ |a\ a\ a\ b\ a\ a\ b \rangle$.

4.2. Minimal Frequency

Until now, we assumed that for the target itemset we need to be able to report its frequency exactly. We will now relax this requirement by setting a minimal frequency threshold *minfreq*. That is, for the target itemset, we should be able, at any timepoint, to produce its exact frequency only if it is above the frequency threshold. This relaxation allows us to decrease the size of the summary.

Let $\mathbb{S}_t$ be a stream with $S_t = [(p_1, a_1), \ldots, (p_r, a_r)]$, and suppose that

$$freq(a, \mathbb{S}_t[p_1, t]) = \frac{a_1 + \ldots + a_r}{t - p_1 + 1} < minfreq\ .$$

Then we can safely remove (p_1, a_1) from the left-side of the summary; even though it is possible that p_1 can still become the starting point of a maximal window in the future, it can be proven that it can never be the starting point of a maximal window *in which the target item is above the threshold*. Indeed; suppose that $freq(A, (\mathbb{S}_t \cdot \mathbb{B})[p_1, t + |\mathbb{B}|])$ exceeds the minimal frequency threshold, then it is easy to show that $freq(A, \mathbb{B})$ must be even larger, and hence p_1 is not the maximal border. In order to be able to perform this pruning efficiently, we store and maintain for the summaries also the count $total = a_1 + a_2 + \ldots + a_r$. When the left-most border is pruned, $total$ is decreased by a_1 to reflect the new total.

4.3. Minimal Window Length

In the algorithm without minimal window length, as given in Algorithm 1, we use the fact that a border q in stream $\mathbb{S}$ can be pruned if we can find two blocks $\mathbb{B}_1 = \mathbb{S}[p, q-1]$ and $\mathbb{B}_2 = \mathbb{S}[q, r]$ such that the frequency of the target in $\mathbb{B}_1$ is higher than in $\mathbb{B}_2$. The intuition behind the proof of this theorem is that in such a situation, q can never become a border again, because either the window starting

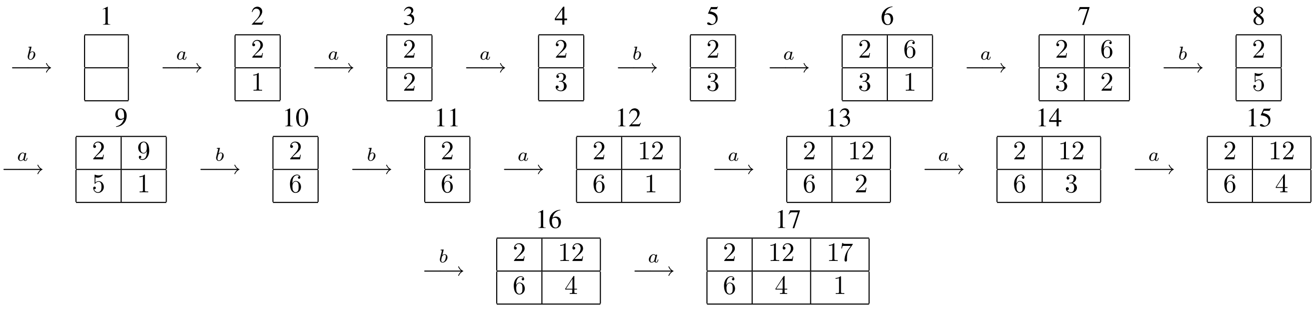


Figure 3. Example for stream $\langle\ b\ a\ a\ a\ b\ a\ a\ b\ a\ b\ b\ a\ a\ a\ a\ b\ a\rangle$**.**

at p will have higher frequency, or the window starting at $r + 1$ has. When we are working with a minimal window length, however, this observation does no longer imply that q can be pruned! Indeed; it could be the case that the suffix of the stream starting at $r + 1$ does not meet the minimal window length requirement. In that case, even though the window starting at q has lower frequency than the window starting at $r+1$, it can still have the highest frequency of all windows *that meet the minimal window requirement*! The next example illustrates this situation.

Example 5 *Consider stream* $\mathbb{S} = \langle|a\ a\ a\ b\ |a\ a\rangle$ *in which the borders* 1 *and* 5 *are marked with a vertical bar. When itemset* $\{b\}$ *arrives in the stream, resulting in* $\langle|a\ a\ a\ b\ a\ a\ b\rangle$*, then position* 5 *is no longer a border, as the block* $a\ a\ a\ b$ *before position* 5 *has a higher frequency of the target item than the block* $a\ a\ b$ *after position* 5*. Therefore, in the algorithm without minimal window length, the border at position* 5 *is pruned, because no matter how the stream evolves, position* 5 *will never be a border again.*

However, consider now the case where we do have a minimal window length of 3*. Then, position* 5 *can still become a border again! Indeed, suppose two more target itemsets are added to the stream, resulting in:* $\langle|a\ a\ a\ b\ .a\ a\ b\ |a\ a\rangle$*. In this stream, the window starting at position* 5 *has the highest frequency of the target items* among the windows satisfying the minimal window length.

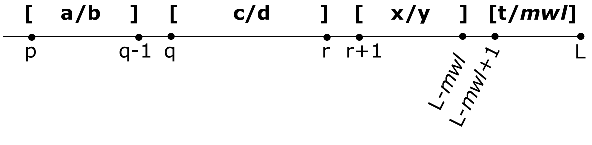
Similarly, the minimal window length also has an influence on the pruning of summary entries based on the minimal frequency. In the case we have to check if we have to remove the last entry of a summary $[(p, a)]$ of a stream $\mathbb{S}_t$, we test whether $a/(t - p + 1) < minfreq$, and the reasoning is that if this is the case, every extension $\mathbb{B}$ that would turn p into a maximal and frequent border, would be even more frequent itself, so p can be removed as a border. With minimal window length, this is no longer true, though, as this $\mathbb{B}$ could not meet the minimal window length. In this case, p might be the starting point of the maximal window *of length at least* mwl.

Fortunately, as the next theorem states, this problem can easily be resolved as follows:

Theorem 2 *Let* $\mathbb{S}$ *be a stream of length* L*, and let* mwl *be the minimal window length. Let* $\mathbb{S}^{-mwl}$ *denote* $\mathbb{S}[1, L - mwl]$*. If* $q = startmax^{mwl}(A, \mathbb{S})$*, then,*

- *either,* $q = L - mwl + 1$*,*
- *or,* q *is a border in* $\mathbb{S}^{-mwl}$*.*

Proof 3 *Notice that, because the length of the maximal window is at least* mwl*, we have* $q \leq L - mwl + 1$*. Hence, either* $q = L - mwl + 1$*, or* $q < L - mwl + 1$*. In the latter case, we have to show that* q *is a border in* $\mathbb{S}[1, L - mwl]$*. Because of Theorem 1 it now suffices to show that for any positions* p *and* r *in* $\mathbb{S}[1, L - mwl]$*, such that* $1 \leq p < q < r \leq L - mwl$*, the frequency of* A *in* $\mathbb{S}[p, q - 1]$ *is less than the frequency of* A *in* $\mathbb{S}[q, r]$*. We denote the number of occurrences of target* A *in* $\mathbb{S}[p, q - 1]$ *by* a *and* $q - p$ *by* b*, and we denote the number of occurrences of the target* A *in* $\mathbb{S}[q, r]$ *by* c *and* $r - q + 1$ *by* d*. The number of occurrences of the target* A *in* $\mathbb{S}[r + 1, L - mwl]$ *is denoted* x *and* y *is* $L - mwl - r$ *and the number of occurrences of the target* A *in* $last(mwl, \mathbb{S})$ *by* t*. These notations are summarized in the next picture:*



Using basic arithmetic operations, it is easy to show that in this setting, $freq(A, \mathbb{S}[p, q - 1]) < freq(A, \mathbb{S}[q, r])$*, i.e.* $a/b < c/d$*.*

Hence, in order to know the maximal frequency with a minimal window length mwl, it suffices to apply the method *without any minimal window length* to keep track of the borders for the stream $\mathbb{S}[1, L - mwl]$. Then, when we need the max-frequency, we check the borders of $\mathbb{S}[1, L - mwl]$ in the complete stream $\mathbb{S}$, and the minimal window itself, $last(mwl, \mathbb{S})$.

4.4. Mining All Itemsets

Until now, we merely focused on mining a single frequent itemset. Of course, in reality, the goal is to find *all* frequent itemsets in the stream. A straightforward way to do this is to apply Algorithm 1, together with Theorem 2 for all itemsets at the same time. That is, for every itemset, we maintain a summary for the stream minus the last mwl transactions. Of course, this is impossible to do for all itemsets. Fortunately, this can be resolved using the following observation (without proof due to space limitations).

Theorem 3 *Let $\mathbb{S}$ be a stream of length L. $\mathbb{S}^{-mwl}$ denotes $\mathbb{S}[1, L - mwl]$. Suppose that $mfreq^{mwl}(A, \mathbb{S}) \geq minfreq$. If $q = startmax^{mwl}(A, \mathbb{S})$, then,*

- *either $L - 2 \cdot mwl + 2 \leq q \leq L - mwl + 1$*
- *or, the following conditions are all fulfilled:*
  - *$freq(A, \mathbb{S}[q, q + mwl - 1]) \geq minfreq$,*
  - *$mfreq^{\mathbf{1}}(A, \mathbb{S}^{-mwl}) \geq minfreq$, and*
  - *q is a border in $\mathbb{S}^{-mwl}$.*

Proof 4 *First of all, because the length of the maximal window is at least mwl, we have that $q \leq L - mwl + 1$. We now can have that $q > L - 2mwl + 1$ or $q \leq L - 2mwl + 1$. The first case, $q > L - 2mwl + 1$, leads to the situation $L - 2mwl + 2 \leq q \leq L - mwl + 1$. In the case that $q \leq L - 2mwl + 1$, we have to prove the above three statements.*

This can easily be proven using similar techniques as in the proof of Theorem 2.

Hence, we do not need to maintain the summaries of all itemsets, but only of those that were once frequent in the minimal window, and that are, at the same time, frequent now within the part of the stream $\mathbb{S}[1, L - mwl]$. Furthermore, we need to find the frequent itemsets in the mwl windows $\mathbb{S}[L - 2mwl + 1, L], \ldots, \mathbb{S}[L - mwl, L]$.

Hence, the algorithm to update the summary when a new transaction T arrives is as follows: for every itemset A for which we are maintaining a summary, update the summary with the transaction that leaves the minimal window. Check if max-frequency in the part of the stream without the minimal window is still frequent. If not, remove the summary. Then, for all itemsets that are frequent in the minimal window and for which we are not yet maintaining a summary, start a summary. In this way, we guarantee that we are able to capture all maximal windows with $q \leq L - 2 \cdot mwl$. Furthermore, we always keep the last $2 \cdot mwl$ transactions. When the frequent itemsets are required, we need to generate all frequent itemsets from the summaries *plus all itemsets frequent in one of the windows* $\mathbb{S}[L - 2mwl + 1, L], \ldots, \mathbb{S}[L - mwl, L]$. This can be done efficiently with a small adaptation to efficient incremental algorithms that have already been proposed in literature [11].

5. Worst Case Analysis

In this section we study how *large* the summary can be in worst case. For a specific streamlength l, we will identify a stream of this length that maximizes the number of borders. Farey sequences play an important role in this analysis.

5.1. Farey Streams

Consider a stream of length l in which we have N borders, and the blocks separated by these borders have lengths $l_1, \ldots, l_N$, and contain respectively $a_1, \ldots, a_N$ times the target:

$$\Big|\ \boxed{a_1/l_1}\ \Big|\ \boxed{a_2/l_2}\ \Big|\ \cdots\ \Big|\ \boxed{a_N/l_N}\ .$$

From Theorem 1, we know that the frequencies of the target itemset in the blocks must be increasing:

$$\frac{a_1}{l_1} < \frac{a_2}{l_2} < \cdots < \frac{a_N}{l_N}.$$

Thus, with every stream with N borders corresponds such an increasing sequence of N fractions. We call this sequence of fractions the *block frequency sequence* of the stream. The length of the stream is the sum of the denominators $l_1 + \ldots + l_N$. The other direction is also true: for every increasing sequence of numbers

$$0 < \frac{a'_1}{l'_1} < \frac{a'_2}{l'_2} < \cdots < \frac{a'_N}{l'_N} \leq 1\ ,$$

we can find a stream of length $l'_1 + \ldots + l'_N$ with N borders, namely:

$$|\overbrace{a\ldots a}^{a'_1\times}\ \overbrace{b\ldots b}^{l'_1-a'_1\times}|\overbrace{a\ldots a}^{a'_2\times}\ \overbrace{b\ldots b}^{l'_2-a'_2\times}|\ldots|\overbrace{a\ldots a}^{a'_N\times}\ \overbrace{b\ldots b}^{l'_N-a'_N\times}$$

We will call this stream the *canonical stream associated with the sequence $a'_1/l'_1 < a'_2/l'_2 < \ldots < a'_N/l'_N$* . Therefore, finding the maximal number of borders for a stream length l corresponds to finding the largest number of different fractions between 0 and 1, of which the sum of the denominators adds up to l. In this context, the notion of *Farey sets* and *Farey sequences* will be very useful.

Definition 3 *The Farey set of order k, denoted F_k is the following set of completely reduced fractions:*

$$F_k \ := \ \left\{ \frac{a}{b} \;\middle|\; gcd(a, b) = 1,\ \ 0 < a \leq b \leq k \right\}\ .$$

The Farey Sequence [2] of order k, is the list where the elements of F_k are ordered in increasing order.

Just like any other increasing sequence of fractions, also the Farey sequence F_k can be associated with its canonical stream $\mathbb{F}_k$, which has $|F_k|$ borders, and a length that equals the sum of the denominators of the elements in F_k. For example, consider the Farey sequence of the fifth order:

$$F_5 = \frac{1}{5} < \frac{1}{4} < \frac{1}{3} < \frac{2}{5} < \frac{1}{2} < \frac{3}{5} < \frac{2}{3} < \frac{3}{4} < \frac{4}{5} < \frac{1}{1} .$$

The corresponding *Farey stream* of the fifth order, $\mathbb{F}_5$, is given in Figure 4. This stream has $|F_5| = 10$ borders and a total length of $5+4+3+5+2+5+3+4+5+1 = 37$.

We will now show that the Farey streams have the *maximal* number of borders; that is, for every stream $\mathbb{S}$ of length equal to the length of $\mathbb{F}_k$, the number of borders in $\mathbb{S}$ is less than or equal to the number of borders in $\mathbb{F}_k = |F_k|$. This result is based on the following straightforward observation. Let $dsum(\{a_1/l_1, \ldots, a_N/l_N\}) = \sum_{i=1}^{N} l_i$, i.e., $dsum(S)$ is the sum of the denominators of the elements in S.

Lemma 1 *Let $S = \{a_1/l_1, \ldots, a_N/l_N\}$ be a set of N different fractions, with $0 < a_i < l_i$, for all $i = 1 \ldots N$. Let k be such that $|S| > |F_k|$, then*

$$dsum(S) > dsum(F_k) \ .$$

Theorem 4 *Let $\mathbb{S}$ be a stream with $L = |\mathbb{F}_k|$. Then, the number of borders in $\mathbb{S}$ is at most the number of borders in $\mathbb{F}_k$.*

Corollary 2 *Let $l = dsum(F_k)$, and $N = |F_k|$, for a fixed k. A stream of length l has maximally N borders.*

5.2. Bounds

For a Farey stream $\mathbb{F}_k$ the number of borders in it equals $|F_k|$ and the length equals $dsum(F_k)$. This representation does, however, not reveal the actual ratio between the size and the number of borders of a stream. Therefore, the asymptotic behavior of these quantities has been worked out, based on known results in number theory.

$$\begin{aligned} \sum_{i=1}^{k} \phi(i) &= \frac{3k^2}{\pi^2} + \mathcal{O}(k \log k) \ , \\ \sum_{i=1}^{k} i \cdot \phi(i) &= \frac{2k^3}{\pi^2} + \mathcal{O}(k^2 \log k) \ . \end{aligned}$$

This leads to the observation that, asymptotically, the number of borders N and the length of the stream L in worst case are related as follows:

$$N = \left(\frac{\pi^2 L}{2}\right)^{2/3} \frac{3}{\pi^2} \ .$$

Experiments for Farey streams up to length 10^7 has shown this approximation to be extremely accurate.

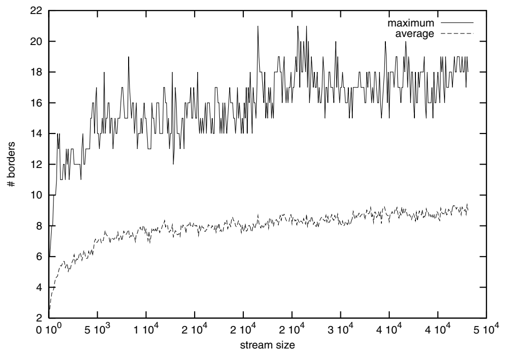


Figure 6. Size of the summaries for a real-life dataset

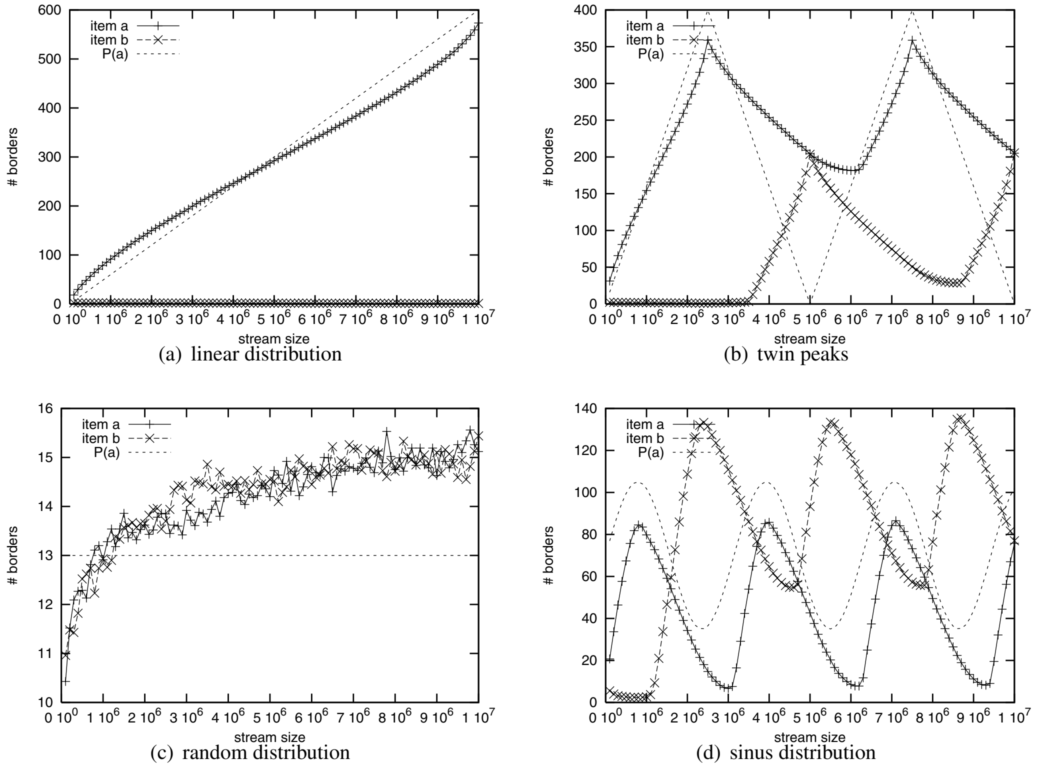
6. Experiments

From the description of the algorithm it is clear that the update procedure is very efficient, given that the summaries remain small. Producing the current support of the target itemset is obviously very efficient, as it amounts to simply a lookup of the most recent entry. Hence, the complete approach will be feasible if and only if the summaries remain small. Therefore, for different synthetical streams, we have recorded the size of the summary. The results are reported in Figure 5. For didactic reasons, we consider streams over two items a and b. Note that it does not matter whether we report for 2 itemsets or for 1 000 itemsets, as the itemsets do not influence the size of each others summary. The streams have a length of 10^7, and after every 10 000 items, the size of the summary for the items a and b are reported. The streams are randomly generated. The probability of having itemset $\{a\}$ in the stream is given by the line $P(a)$. Thus, in the random graph, the probability of having a is $1/2$ in the whole stream, independent of the moment. The probability of b is 1 minus the probability of a. The graphs report the average over 100 streams, generated with the indicated distributions. In general, we can conclude that the size of the summary is extremely small w.r.t. the size of the stream. If the probability of the target item increases also the size of the summary will increase, when the probability decreases the summary will shrink. This is easily explained by the entries in the summary that need to have increasing frequency.

In Figure 6, this experiment is repeated for a real-life dataset. This dataset was obtained by collecting one month of click-stream data of visitors of the website of the University of Antwerp. For every minute a transaction is generated, consisting of the set of all webpages visited in that minute. For every webpage the max-frequency is monitored with a minimal window length of 60. No minimal support threshold was used. At every timepoint, the maximal and the average number of borders is plotted for all web-pages. As can be seen in this real-life experiment, again the sizes

$$\langle\ |a\ b\ b\ b\ b\ |a\ b\ b\ b\ |a\ b\ b\ |a\ a\ b\ b\ b\ |a\ b\ |a\ a\ a\ b\ b\ |a\ a\ b\ |a\ a\ a\ b\ |a\ a\ a\ a\ b\ |a\rangle$$

Figure 4. Illustration of $\mathbb{F}_5$, the Farey Stream of fifth order.



(a) linear distribution

(b) twin peaks

(c) random distribution

(d) sinus distribution

Figure 5. Size of the summaries for synthetic data

of the summaries remain extremely small, leading to a very efficient algorithm.

7. Related Work

There are already many other measures defined for frequency of itemsets over streams. These measures can roughly be divided into three categories: the sliding window model [3, 6, 7, 8, 10, 12], the time fading model [9], and the landmark model [7, 8, 13]. The different frequency measures are illustrated in Figure 7. The bottom line reports the probability at that timepoint in the stream of the itemset for which the frequency is reported. Notice that this bottom line only gives the probability; the actual frequency can be slightly different. At every timestamp, the frequency of the target itemset has been plotted for the different measures.

In the *time-fading* model [9], the entire stream is taken into account to compute the frequency of an itemset, but more recent transactions contribute more to the frequency than older ones. This is achieved by introducing a decay factor $d < 1$. A transaction that is n timepoints in the past, is weighted d^n, thus the weight is exponentially decreasing. In general, the closer to 1 the decay, the more the history is taken into account. In Figure 7, the time-fading frequency has been given for two different decay factors, 0.99 and 0.999. Notice that although these two values are very similar, the evolution of the frequency is very different.

In the *sliding window* [3, 6, 7, 8, 10] model, at every time point, only the data in the most recent window of a predefined fixed length (measured either in duration or in number of transactions) is considered. In Figure 7, the sliding window frequency is plotted for window lengths 200 and 400. Notice that for window length 400, some of the frequency jumps go by unnoticed and others are significantly lowered, because of the smoothing implied by a large window length.

In the *landmark* model [7, 8, 13], particular timepoints, called landmarks, are fixed. The analysis of the stream is performed for only the part of the stream between the landmarks and the current time instance. Clearly, this method is less suitable for evaluating evolving and unbounded streams.

The tilted-time windows [5, 4] can be seen as a combina-

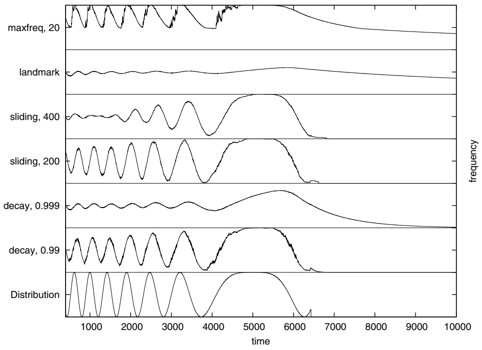


Figure 7. Comparison of different frequency measures

tion of the different models. For an itemset, frequencies are computed for the most recent windows of lengths w, $2w$, $4w$, $8w$, etc. So, the most recent part of the stream is covered more thoroughly. The combination of these frequencies allow for efficient query answering over the history of the stream.

Also the max-frequency is given (the top line) in Figure 7. As can be seen in the illustration, max-frequency takes into account the history without fading away sudden jumps in the frequency. As a direct result of this, the line for the max-frequency is less smooth than the other lines, because the actual frequency of the item only approximates the given distribution. Notice that these deviations also show in the time-fading model (e.g. with decay .99), although in a far less pronounced way. The other methods do not show the existing short deviations from the ideal distribution, as they are less sensitive to short-time changes. Max-frequency is much less dependent on finding the exact right parameter setting than other models, because it only determines a lower bound on the window size; e.g., for a stream of length 10 000, and extreme minimum window lengths of 10 and 1 000, still 9 010 of the 10 000 windows are treated in the same way for both parameter settings. In the other models, the parameters completely determine the weight of every point in the stream; for different parameter settings, all 10 000 points in the stream will be handled differently.

8. Conclusion

We presented a new frequency measure for itemsets in streams that does not rely on a fixed window length or a time-decaying factor. Based on the properties of the measure, an algorithm to compute it was shown. An experimental evaluation supported the claim that the new measure can be computed from a summary with extremely small memory requirements, that can be maintained and updated efficiently. The summary of the stream consists of the borders and their corresponding frequencies. For a specific type of streams, the so-called Farey streams, we theoretically showed an upper bound on the size of the summary, by giving an upper bound on the size of the borders.

References

[1] T. Calders, N. Dexters, and B. Goethals. Mining frequent items in a stream using flexible windows. *Intelligent Data Analysis*, 12(3), May 2008.

[2] J. H. Conway and R. K. Guy. Farey fractions and ford circles. In *The Book of Numbers*, pages 152–154. Springer-Verlag, 1996.

[3] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *ESA*, pages 348–360, 2002.

[4] C. Giannella, J. Han, J. Pei, X. Yan, and P. Yu. Mining frequent patterns in data streams at multiple time granularities. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, editors, *NSF Workshop on Next Generation Data Mining*, 2002.

[5] C. Giannella, J. Han, E. Robertson, and C. Liu. Mining frequent itemsets over arbitrary time intervals in data streams. Technical Report TR587, Indiana University, Bloomington, USA, November 2003.

[6] L. Golab, D. DeHaan, E. D. Demaine, A. López-Ortiz, and J. I. Munro. Identifying frequent items in sliding windows over on-line packet streams. In *Internet Measurement Comference*, pages 173–178, 2003.

[7] R. Jin and G. Agrawal. An algorithm for in-core frequent itemset mining on streaming data. In *ICDM*, pages 210–217, 2005.

[8] R. M. Karp, S. Shenker, and P. H. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28:51–55, 2003.

[9] D. Lee and W. Lee. Finding maximal frequent itemsets over online data streams adaptively. In *ICDM*, pages 266–273, 2005.

[10] C.-H. Lin, D.-Y. Chiu, Y.-H. Wu, and A. L. P. Chen. Mining frequent itemsets from data streams with a time-sensitive sliding window. In *SDM*, 2005.

[11] A. Veloso, W. Meira Jr., M. de Carvalho, B. Pôssas, S. Parthasarathy, and M. J. Zaki. Mining frequent itemsets in evolving databases. In *SDM*, 2002.

[12] R. Wong and A. Fu. Mining top-K frequent itemsets from data streams. *Data Mining and Knowledge Discovery*, 13(2):193–217, 2006.

[13] J. X. Yu, Z. Chong, H. Lu, and A. Zhou. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In *VLDB*, pages 204–215, 2004.