Information Systems 39 (2014) 233–255

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Mining frequent itemsets in a stream

Toon Calders [a], Nele Dexters [b], Joris J.M. Gillis [c,*,1], Bart Goethals [b]

[a] Eindhoven University of Technology, The Netherlands
[b] University of Antwerp, Belgium
[c] Hasselt University, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium

ARTICLE INFO

Available online 30 January 2012

Keywords:
Frequent itemset mining
Datastream
Theory
Algorithm
Experiments

ABSTRACT

Mining frequent itemsets in a datastream proves to be a difficult problem, as itemsets arrive in rapid succession and storing parts of the stream is typically impossible. Nonetheless, it has many useful applications; e.g., opinion and sentiment analysis from social networks. Current stream mining algorithms are based on approximations. In earlier work, mining frequent items in a stream under the max-frequency measure proved to be effective for items. In this paper, we extended our work from items to itemsets. Firstly, an optimized incremental algorithm for mining frequent itemsets in a stream is presented. The algorithm maintains a very compact summary of the stream for selected itemsets. Secondly, we show that further compacting the summary is non-trivial. Thirdly, we establish a connection between the size of a summary and results from number theory. Fourthly, we report results of extensive experimentation, both of synthetic and real-world datasets, showing the efficiency of the algorithm both in terms of time and space.

1. Introduction

Mining frequent itemsets in large static so-called *transaction* databases has been the topic of numerous studies for over the past 15 years. Many efficient algorithms and optimizations have been discovered, which have already made it into several commercial database and data mining products. When the given database is a dynamically and fast evolving stream of data, for which also a continuously up-to-date analysis needs to be provided, these known techniques are suddenly no longer applicable. Mining frequent itemsets over such streams of itemsets presents interesting new challenges. The speed of new arriving itemsets excludes revisiting the history, unless it is stored. But storing large parts of a stream is typically impossible, however, as huge volumes of data pass. Yet, mining streams has numerous interesting applications, for example, opinion and sentiment mining from social networks, network traffic analysis, sensor network analysis, time-dependent market basket analysis, stock price analysis, and much more.

Mining frequent itemsets over streams is an extension of mining frequent items over streams. For a comprehensive overview of the current state of the art, see Cormode and Hadjieleftheriou [9]. Although many of the techniques developed for frequent items can be reused; see, e.g., the hMiner algorithm of Wang and Chen [21] based on hCount and Lossy Counting, there are specific challenges associated with mining frequent itemsets, including the combinatorial explosion of the number of patterns, and that any transaction arriving over the stream can support an exponential (in its length) number of patterns. Some interesting approaches have been taken to tackle these problems, most of them focusing either on (1) a sliding window model [11–14,17–19] where only the frequent itemsets in the w most recent transactions is required for a given w; a slight deviation is the time-sensitive sliding

[*] Corresponding author. Tel.: +32 11 26 82 07.
E-mail address: joris.gillis@uhasselt.be (J.J.M. Gillis).
[1] PhD Fellow of the Research Foundation Flanders—FWO.

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.is.2012.01.005

window, where not the length of the window is fixed, but the time span, (2) the time-fading model [16], or (3) the landmark model [10,13,14,21,23]. In most of these works, a datastructure is maintained for the current window that either directly contains the frequent patterns, or is a convenient representation of the transactions in the window, such as a bit-sequence or a trie, that allows for quickly determining the frequent itemsets. In order to reduce the memory overhead, sometimes the number of patterns to be monitored is reduced to a condensed set; e.g., the closed itemsets [5,7], or approximate solutions are computed [10,15,21]. Next to mining the frequent itemsets in one stream, further extension to, for instance, distributed streams [22] have been proposed, or even to mining graph patterns over streams [1]. For a survey on many of the techniques for mining frequent itemsets over streams, see [6].

The sliding window or decay models typically require some user-specified parameters, such as a fixed window length or decay factor. Choosing one value of such a parameter adequately for every itemset is practically impossible [3,4]. For example, consider a large retail chain of which sales can be considered as a stream. Then, in order to find frequent itemsets to do market basket analysis, it is very difficult to choose in which period of the collected data you are particularly interested. For many products, the amount sold depends highly on the period of the year. In summer time, sales of ice cream increase and during the world cup, sales of beer increase, while during the new-year festivities, the sales of champagne and many other typical gifts increase. Such seasonal behavior of a specific item can only be discovered when choosing the correct window size for that item. This size, however, can hide a similar behavior of other items in another window. Therefore, we introduced the *max-frequency* measure and a new mining algorithm for solving this problem [3,4]. Using this new measure, the window length can vary for each itemset separately. More specifically, for every itemset, the optimal window length, for which the itemset's frequency is the highest, is being considered.

In this paper, we significantly extend our previous results. First, we revisit the definitions for *max-frequency* and its properties, and we explain the basic algorithm for mining a single max-frequent itemset in a stream [4]. Essentially, we have to maintain a summary of the stream in order to provide the max-frequency at any given time, which needs to be kept up to date continuously with the stream.

Several optimizations for speeding up the algorithm and a worst case analysis, based on well-known results from number theory, are given.

Then, we extend the work on items to itemsets and we investigate whether we need to maintain a summary for every itemset occurring in the stream and dig into compressing the memory size needed. Unfortunately, as we will show, attempting to exploit the subset relation between itemsets in order to compact the summaries is futile. Finally, we report results of extensive experimentation with our software, both on synthetic and real-world datasets. These experiments show that our algorithm is efficient both in terms of time and space.

2. Max-frequency revisited

In this section we review the definition of the max-frequency measure and the problem definition [2–4]. Throughout the paper, we assume that a finite set of items $\mathcal{I}$ has been given. An itemset is any subset of $\mathcal{I}$.

Definition 2.1 (*Calders et al. [3]*). A *stream* $\mathbb{S}$ is a sequence of itemsets $\langle s_1\ s_2\ \cdots\ s_n\rangle$. $n=|\mathbb{S}|$ is the *length* of the stream. s_1 is considered the first and oldest itemset in the stream, and s_n the latest and youngest. We will use s_t, $1\le t\le n$, to denote the t-th itemset in the stream $\mathbb{S}$. We call s_t the *itemset at timestamp t in stream* $\mathbb{S}$.

The *concatenation* of two streams $\mathbb{S}_1$ and $\mathbb{S}_2$ will be denoted $\mathbb{S}_1\cdot\mathbb{S}_2$.

Let $1\le i\le j\le|\mathbb{S}|$. $\mathbb{S}[i,j]$ denotes the stream $\langle s_i\ s_{i+1}\ \cdots\ s_j\rangle$. $\mathbb{S}[i,]$ denotes $\mathbb{S}[i,|\mathbb{S}|]$, and $\mathbb{S}[,j]$ denotes $\mathbb{S}[1,j]$.

The suffix of $\mathbb{S}$ consisting of the last k itemsets of $\mathbb{S}$, denoted $last(k,\mathbb{S})$ is

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

Throughout the paper we will be illustrating our algorithms by a running example based upon the following stream (the numbers at the top denote the timestamps of the itemsets in the stream):

$$\begin{array}{ccccccccccccc} & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \mathbb{S} & = & a & b & \emptyset & ab & \emptyset & b & ab & a & b & \emptyset & ab \end{array}$$

We use the shorthand a, b, ab as shorthand for the sets of items $\{a\}$, $\{b\}$, and $\{a,b\}$, respectively. For this example stream, $|\mathbb{S}|=11$, $s_1=a$, $s_7=ab$, $\mathbb{S}[5,9]=\langle\emptyset\ b\ ab\ a\ b\rangle$, $\mathbb{S}[,3]=\langle a\ b\ \emptyset\rangle$, and $last(3,\mathbb{S})=\langle b\ \emptyset\ ab\rangle$.

A stream is here defined as a static 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. When analyzing algorithms and in examples we will therefore assume a stream $\mathbb{S}$ of sufficient length. $\mathbb{S}_t$ will denote the stream $\mathbb{S}$ up to timestamp t; that is $\mathbb{S}[1,t]$, the part of the stream that already passed at time t. At time t only $\mathbb{S}_t$ has been observed so far.

2.1. Counts, frequencies and max-frequency

In [2,4], the following new frequency measure for items in streams was introduced. This measure was trivially extended to itemsets in [3].

Definition 2.2 (*Calders et al. [2]*). $count(A,\mathbb{S})$ denotes the number of times itemset A *occurs* in stream $\mathbb{S}$ and is defined as

$$count(A,\mathbb{S}) := |\{1\le t\le|\mathbb{S}|\,|A\subseteq s_t\}|.$$

The *frequency of A in* $\mathbb{S}$ is defined as

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

Finally, the *max-frequency* $mfreq(A,\mathbb{S})$ of itemset A in a stream $\mathbb{S}$ is defined as the maximum of the frequencies of

Running Example: *Frequencies* **reference chart**

	1	2	3	4	5	6	7	8	9	10	11
$\mathbb{S} =$	a	b	$\emptyset$	ab	$\emptyset$	b	ab	a	b	$\emptyset$	ab

M_a (*Frequencies of a*)

1	100										
2	50	0									
3	33	0	0								
4	50	33	50	100							
5	40	25	33	50	0						
6	33	20	25	33	0	0					
7	42	33	40	50	33	50	100				
8	50	42	50	60	50	66	100	100			
9	44	37	42	50	40	50	66	50	0		
10	40	33	37	42	33	40	50	33	0	0	
11	45	40	44	50	42	50	60	50	33	50	100

M_b (*Frequencies of b*)

1	0										
2	50	100									
3	33	50	0								
4	50	66	50	100							
5	40	50	33	50	0						
6	50	60	50	66	50	100					
7	57	66	60	75	66	100	100				
8	50	57	50	60	50	66	50	0			
9	55	62	57	66	60	75	66	50	100		
10	50	55	50	57	50	60	50	33	50	0	
11	54	60	55	62	57	66	60	50	66	50	100

M_{ab} (*Frequencies of ab*)

1	0										
2	0	0									
3	0	0	0								
4	25	33	50	100							
5	20	25	33	50	0						
6	16	20	25	33	0	0					
7	28	33	40	50	33	50	100				
8	25	28	33	40	25	33	50	0			
9	22	25	28	33	20	25	33	0	0		
10	20	22	25	28	16	20	25	0	0	0	
11	27	30	33	37	28	33	40	25	33	50	100

Fig. 1. Running example: reference chart; entry t,i of the matrix M_A of itemset A contains $freq(A,\mathbb{S}[i,t])$ in percent; i.e., the frequency of A in the suffix starting at position i of $\mathbb{S}_t$. $mfreq(A,\mathbb{S}_t)$ is hence $\max_{i=1,\dots,t}(M_A[t,i])$.

A over all suffixes of the stream; that is

$$mfreq(A,\mathbb{S}) := \max_{k=1,\dots,|\mathbb{S}|}(freq(A,last(k,\mathbb{S}))).$$

Example 1. Consider Fig. 1 which reports the frequencies for the itemsets a, b, and ab in our running example stream. The stream $\mathbb{S}$ has length 11. For timestamps $t=1,\dots,11$, the frequencies of the itemsets in suffixes of the stream $\mathbb{S}_t$ are given. Consider, e.g., the first row in the first matrix: 100. This number indicates that the frequency of the itemset a in the suffix starting at position 1 of $\mathbb{S}_1 = \langle a \rangle$ is 100%. Consider now the stream at timestamp 5; i.e., $\mathbb{S}_5 = \langle a\ b\ \emptyset\ ab\ \emptyset \rangle$. The frequency of a in the suffixes of that stream is 40% (suffix $\mathbb{S}_5[1,]$), 25% (suffix $\mathbb{S}_5[2,]$), 33% (suffix $\mathbb{S}_5[3,]$), 50% (suffix $\mathbb{S}_5[4,]$), 0% (suffix $\mathbb{S}_5[5,]$), as indicated by the fifth row in the matrix M_a. The matrices are lower diagonal matrices as a stream of length n has exactly n suffixes.

The max-frequency at timestamp t of itemset a (b, ab respectively) is now, by definition, equal to the maximum value in the t-th row in the matrix M_a (M_b, M_{ab} respectively). For example, $mfreq(a,\mathbb{S}_6) = \max\{33\%, 20\%, 25\%, 33\%, 0\%, 0\%\} = 33\%$. Other examples are $mfreq(b,\mathbb{S}_9) = 100\%$ (maximum value in the 9th row of the second matrix), and $mfreq(ab,\mathbb{S}_{10}) = 28\%$ (maximum value in the second to last row of the third matrix).

2.2. Minimal window length

One of the problems with the max-frequency measure that is clearly illustrated in Fig. 1 is that every time a transaction arrives in the stream, the frequency of all itemsets, that are contained in that transaction, suddenly peaks and becomes 100%. In Fig. 1 we can observe this, e.g., at timestamp 4 when itemset ab arrives. For itemsets a, b, and ab the frequency peaks to 100%, as these sets have a frequency of 100% in the suffix $\langle ab \rangle$. The solution for this problem, however, is simple: we can disallow too short windows by setting a minimal window length *mwl*.

Definition 2.3 (*Calders et al.* [3]). Given a minimal window size *mwl*, the *max-frequency with minimal window length* $mfreq^{mwl}(A,\mathbb{S})$ of itemset A in a stream $\mathbb{S}$ is defined as the maximum of the frequencies of A over all suffixes of $\mathbb{S}$ with a length of at least *mwl*; that is

$$mfreq^{mwl}(A,\mathbb{S}) := \max_{k=mwl,\dots,|\mathbb{S}|}(freq(A,last(k,\mathbb{S}))).$$

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

Example 2. Consider once more Fig. 1. Setting a minimal window-length *mwl* actually comes down to ignoring the last $mwl-1$ entries of all rows in the matrices, as they correspond to frequencies in suffixes of the streams that are too short. Let $mwl=3$; hence, we ignore the last two entries in all rows. We get $mfreq^{mwl}(a,\mathbb{S}_6) = \max\{33\%, 20\%, 25\%, 33\%\} = 33\%$, $mfreq^{mwl}(b,\mathbb{S}_9) = 75\%$ (the last two entries 50, 100 are ignored), and $mfreq^{mwl}(ab,\mathbb{S}_4) = 33\%$ (the last two entries 50, 100 are ignored).

By setting a minimal window length we can reduce the problem of sudden peaks in frequency for sparse items. Instead of peaking to 100%, the items will peak to a frequency of only $1/mwl$. This behavior can even be further suppressed in the case we only report itemsets that reach a certain minimal frequency *minfreq*, as will be the case in the general problem statement as introduced in the next subsection. In that case, if we carefully chose the minimal window length such that $1/mwl < minfreq$, a single occurrence of a sparse item will no longer result in any peak in frequency whatsoever.

2.3. The max-frequent itemset mining problem

The main problem we study in this paper is the following:

Problem 2.4 (*Calders et al.* [3]). Given a *minimal frequency threshold* and a *minimal window length*, for an evolving stream $\mathbb{S}$, maintain a small summary of the stream, such that, at any timepoint t, all currently max-frequent itemsets can be produced instantly from this summary.

Example 3. For minimal frequency $minfreq=0.4$, and $mwl=3$, the frequent itemsets with their respective frequencies will be as follows for our running example (cfr. Fig. 1):

Time	Itemsets		
	a (%)	*b* (%)	*ab* (%)
1	–	–	–
2	–	–	–
3	33	33	–
4	50	66	–
5	40	50	–
6	–	66	–
7	50	75	50
8	66	66	40
9	66	75	–
10	50	60	–
11	60	66	40

A dash denotes that the set is not part of the output; only if it is present in the output, its frequency is reported in the table.

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 $Get\_mfreq(A,summary(\mathbb{S}_t))$ equals $mfreq(A,\mathbb{S}_t)$.

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.

3. Mining a single itemset

In this section we introduce the algorithms for mining the max-frequency of an itemset with and without minimal window length and/or minimal (max-)frequency threshold. Before we go into the details of the algorithms, first some important notions are revisited.

3.1. Maximal windows, borders, and summaries

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

$$mfreq(A,\mathbb{S}) = freq(A,\mathbb{S}[startmax(A,\mathbb{S}),|\mathbb{S}|]).$$

If the minimum window length is set, the starting point of the maximal window, denoted by $startmax^{mwl}(A,\mathbb{S})$, is the smallest index smaller than or equal to $|S| - mwl + 1$ satisfying

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

Obviously, checking all possible windows to find the maximal one is infeasible algorithmically, given the constraints of the stream problems. Fortunately, not every point in the stream needs to be checked.

Definition 3.1. Timestamp q is called a *border for itemset 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 and corollary, it is possible to give an exact syntactic characterization of the borders.

Theorem 3.2 (Calders et al. [4]). *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 $p < q \le r$: $freq(A,\mathbb{S}[p,q-1]) < freq(A,\mathbb{S}[q,r])$.*

Corollary 3.3 (Calders et al. [4]). *Let $\mathbb{S}$ be a stream, and let $1 \le q \le |\mathbb{S}|$. Position q is a border for target itemset A in $\mathbb{S}$ if and only if for all indices j,k with $1 \le j < q$ and $q \le k \le |\mathbb{S}|$, it holds that $freq(A,\mathbb{S}[j,q-1]) < freq(A,\mathbb{S}[q,k])$.*

Theorem 3.2 and Corollary 3.3 characterize exactly which positions in a stream are borders, i.e., potential starting points of the maximal window. They state that a position q in a stream $\mathbb{S}$ *cannot* be a border if there is a block before q (i.e., $\mathbb{S}[p,q-1]$) with an equal or higher frequency than a block after q (i.e., $S[q,r]$). Thus, if a before- and after-block satisfying the conditions can be found, the position can be pruned from the summary as a border. This is because the before-block will boost the frequency of $\mathbb{S}[p,]$ up to or over the frequency of $\mathbb{S}[q,]$.

Example 4. Consider the stream of our running example. The border positions of itemsets a, b, and ab have been indicated by vertical bars with the set as subscript:

$$\begin{array}{lccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \mathbb{S} = & |_a a & |_b b & \emptyset & |_{ab} ab & \emptyset & |_b b & |_a ab & a & b & \emptyset & |_{a,b,ab} ab \end{array}$$

As can be seen, not all occurrences of an itemset in the stream result in a border for that item; e.g., for a, there is no border at position 8. Similarly for ab there is no border at 7, since, for example, the frequency of ab in the block $[4,6]$ is at least as large as the frequency of ab in the block $[7,10]$. Therefore, 7 can never become a maximal position for ab anywhere in the future, as the position 4 will always represent a window in which the frequency of ab is larger.

This theorem and corollary form the basis of an incremental algorithm to efficiently update the summary for one itemset A. The summary at timestamp t of the itemset A is denoted S_t. From the summary the current max-frequency of A can be produced instantly. The summary of itemset A at time-stamp t will consist of the list of all borders of A in $\mathbb{S}_t$, together with counters that store the number of occurrences of A between the borders; e.g., the summary

$$[(p_1,a_1),\ldots,(p_r,a_r)]$$

denotes that A has borders at positions p_1, p_2, …, p_r. The count a_i equals $count(A,\mathbb{S}[p_i,p_{i+1}-1])$ for $i = 1,\ldots,r-1$, and $a_r = count(A,\mathbb{S}[p_r,])$.

The algorithm for maintaining the summary is now based upon the following observations (for a detailed explanation and proofs, see [4]):

- The frequency of A in the suffixes starting at the borders is increasing from left to right; i.e.,

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

- As such, the maximal border is always the last one; i.e.,

$$mfreq(A,\mathbb{S}_t) = Get\_mfreq(A,summary(\mathbb{S}_t)) = \frac{a_r}{t-p_r+1}.$$

- Borders disappear always from right to left; i.e., if $p_1,\ldots,p_r$ are the borders in $\mathbb{S}_t$, and p_i is no longer a border in $\mathbb{S}_{t+1}$, then neither are $p_{i+1},p_{i+2},\ldots,p_r$. Thus we start from (p_r,a_r) and work towards (p_1,a_1).
- Borders only disappear if non-target items arrive in the stream; i.e., if $A \subset I$, and p is a border in $\mathbb{S}_t$, then p is also a border in $\mathbb{S}_t \cdot \langle I \rangle$.
- If a target item arrives in the stream, a new border is added to the stream, unless the max-frequency of A was already 100%; i.e., if $A \subset I$, then the summary of $\mathbb{S}_t \cdot \langle I \rangle$ will be $[(p_1,a_1),\ldots,(p_r,a_r),(t+1,1)]$ if $a_r < t-p_r+1$, and $[(p_1,a_1),\ldots,(p_r,a_r+1)]$ otherwise.

In Algorithm 1 it is shown how the summary at timestamp $t+1$, denoted S_{t+1}, is derived from the summary at the previous timestamp, S_t, and I, the itemset at $\mathbb{S}[t+1]$. The algorithm is illustrated on the running example in Fig. 2.

The combination of *Update* and the max-frequency deduction is called Max-Freq-Miner. Next, we discuss three extensions of the basic Max-Freq-Miner algorithm to tackle some issues of Max-Freq-Miner. This results in three algorithms derived from Max-Freq-Miner: Max-Freq-Miner^{mwl}, Max-Freq-Miner$_\sigma$ and Max-Freq-Miner$_\sigma^{mwl}$. These three algorithms consider only one itemset at a time and are potentially very inefficient when tracking all itemsets satisfying the constraints. Therefore, an optimized algorithm for mining all itemsets, Max-Freq-Miner-All$_\sigma^{mwl}$ is given as well.

Algorithm 1. *Update*(S_t,I) for one target itemset A on time $t+1$, *without* minimal frequency threshold, and *without* minimal window length.

```
Require: $S_t = summary(\mathbb{S}_t) = [(p_1,a_1),\ldots,(p_r,a_r)]$
Ensure: $S_{t+1} = summary(\mathbb{S}_{t+1}) = summary(\mathbb{S}_t \cdot \langle I \rangle)$
Set $S_{t+1}$ := [ ]
if($S_t$ is empty) then
    if (target itemset $A \subseteq I$) then
        $S_{t+1}$ := [$(t+1,1)$]
else
    if (target itemset $A \subseteq I$) then
        if $a_r = t-p_r+1$ then
            $S_{t+1}$ :=[$(p_1,a_1),\ldots,(p_r,a_r+1)$]
        else
            $S_{t+1}$ :=[$(p_1,a_1),\ldots,(p_r,a_r),(t+1,1)$]
    else
        $S_{t+1}$ := $S_t$
        $i$ := $r$
        while $i>1$ do
            if $\frac{a_i}{t-p_i+1} \le \frac{a_i+a_{i-1}}{t-p_{i-1}+1}$ then
                $a_{i-1}$ := $a_{i-1}+a_i$
                remove $(p_i,a_i)$ from $S_{t+1}$
                $i$ := $i-1$
            else
                $i$ := 1
```

3.2. Minimal window length

As noted in the previous section, a minimal window length resolves the frequency peaks of rare itemsets.

3.2.1. Pruning

In Max-Freq-Miner 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 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

Running Example: *MaxFreq*

	1	2	3	4	5	6	7	8	9	10	11
$\mathbb{S} =$	a	b	$\emptyset$	ab	$\emptyset$	b	ab	a	b	$\emptyset$	ab

t	s_t	*Summaries*			*Frequencies*		
		a	b	ab	a	b	ab
1	a	[(1, 1)]	[]	[]	100	0	0
2	b	[(1, 1)]	[(2, 1)]	[]	50	100	0
3	$\emptyset$	[(1, 1)]	[(2, 1)]	[]	33	50	0
4	ab	[(1, 1), (4, 1)]	[(2, 1), (4, 1)]	[(4, 1)]	100	100	100
5	$\emptyset$	[(1, 1), (4, 1)]	[(2, 2)]	[(4, 1)]	50	50	50
6	b	[(1, 2)]	[(2, 2), (6, 1)]	[(4, 1)]	33	100	33
7	ab	[(1, 2), (7, 1)]	[(2, 2), (6, 2)]	[(4, 1), (7, 1)]	100	100	100
8	a	[(1, 2), (7, 2)]	[(2, 2), (6, 2)]	[(4, 1), (7, 1)]	100	66	50
9	b	[(1, 2), (7, 2)]	[(2, 2), (6, 2), (9, 1)]	[(4, 1), (7, 1)]	66	100	33
10	$\emptyset$	[(1, 2), (7, 2)]	[(2, 2), (6, 3)]	[(4, 2)]	50	60	28
11	ab	[(1, 2), (7, 2), (11, 1)]	[(2, 2), (6, 3), (11, 1)]	[(4, 2), (11, 1)]	100	100	100

Fig. 2. Running example: max-frequency (no mwl, no minfreq).

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 $\langle a\ a\ a\ b\rangle$ before position 5 has a higher frequency of the target item than the block $\langle a\ a\ b\rangle$ 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.

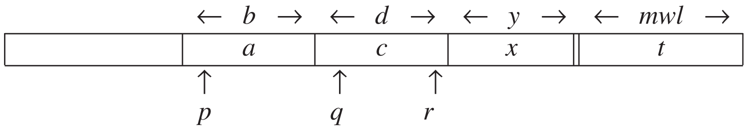
However, consider now the case where we do have a minimal window length of 3. Then, position 5 can still become a border again! Indeed, suppose two more target itemsets are added to the stream, resulting in $\langle |a\ a\ a\ b\ .a\ a\ b\ |a\ a\rangle$. In this stream, the window starting at position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

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

Theorem 3.4 (*Calders et al.* [3]). *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 (*Only a short version of the proof was included in* [3]). First of all, because the length of the maximal window is at least *mwl*, we have that $q \le L-mwl+1$. We now can have that $q = L-mwl+1$ or $q < L-mwl+1$. In the latter case, we have to show that q is a border in $\mathbb{S}[1,L-mwl]$. According to Corollary 3.3, we have to prove that the frequency of target A in every block before position q is strictly less than the frequency in every block following q. More concretely, we pick a certain $1 \le p < q$ and we denote the occurrences of target A in $\mathbb{S}[p,q-1]$ by a and we use the shorthand notation b for the length of this substream, $q-p$. We also pick a certain $q \le r \le L-mwl$ and we denote the occurrences of the target A in $\mathbb{S}[q,r]$ by c and we introduce the shorthand notation d for the length $r-q+1$. Finally, the occurrences of the target A in $\mathbb{S}[r+1,L-mwl]$ are denoted by x and y is used to express the length of this substream, $L-mwl-r$. Remark that in the case of $q = L-mwl$ or $q < L-mwl$ but $r = L-mwl$, x=0 and y=0. We also denote the amount of occurrences of the target A in $last(mwl,\mathbb{S})$ by t, and the length of this last substream equals *mwl*. This situation is depicted in the following visual:



It is sufficient to show that $freq(A,\mathbb{S}[p,q-1]) < freq(A,\mathbb{S}[q,r])$, i.e., $a/b < c/d$. Because q is the starting point of the maximal window of length at least *mwl* in $\mathbb{S}$, we know that

$$\frac{c+x+t}{d+y+mwl} > \frac{a+c+x+t}{b+d+y+mwl} \quad \text{and} \quad \frac{c+x+t}{d+y+mwl} \ge \frac{x+t}{y+mwl}.$$

Notice that the first inclusion is strict, since $startmax^{mwl}(A,\mathbb{S})$ is the smallest index on which the max-frequency is reached in case of a tie. These inequalities are equivalent to

$$\frac{c+x+t}{d+y+mwl} > \frac{a}{b} \tag{1}$$

and

$$\frac{c}{d} \ge \frac{x+t}{y+mwl}. \tag{2}$$

Assume now for the sake of contradiction that $a/b \ge c/d$. We then have, according to (1), that

$$\frac{c+x+t}{d+y+mwl} > \frac{a}{b} \ge \frac{c}{d} \Rightarrow \frac{x+t}{y+mwl} > \frac{c}{d},$$

which is not possible, as shown in (2). Therefore, the assumption is wrong, illustrating that $a/b < c/d$. □

3.2.2. Max-Freq-Miner^{mwl}

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 $\mathbb{S}_t[1,t-mwl] = \mathbb{S}_t^{-mwl}$. Then, when we need the max-frequency, we check the borders of $\mathbb{S}_t^{-mwl}$ in the complete stream $\mathbb{S}$, and the minimal window itself, $last(mwl,\mathbb{S})$. The summary of the stream $\mathbb{S}_t$ with minimal window length is a pair consisting of the normal summary of $\mathbb{S}_t^{-mwl}$ and the content of the minimal window itself; i.e., $\mathbb{MW}_t := last(mwl,\mathbb{S})$. We will call this summary, the *mwl*-summary *of* $\mathbb{S}_t$. Algorithm 2 explains how the *mwl*-summary $(S_t,\mathbb{MW}_t)$ of $\mathbb{S}_t$ is updated upon the arrival of a new itemset I in order to get the *mwl*-summary $(S_{t+1},\mathbb{MW}_{t+1})$ for $\mathbb{S}_{t+1} = \mathbb{S}_t \cdot \langle I\rangle$. Notice that for a new stream the first *mwl*-summary will be created at timepoint *mwl*, and will be $([],\mathbb{S})$. Notice also that *Update* refers to the summary updating method without minimal window length as described in Algorithm 1.

Algorithm 2. $Update^{mwl}((S_t,\mathbb{MW}_t),I)$ for *one target itemset* I on time $t+1$, *with minimal window length mwl*, and *without minimal frequency threshold*.

```
Require: $(S_t,\mathbb{MW}_t)$, the $mwl$-summary of $\mathbb{S}_t$
Ensure: $(S_{t+1},\mathbb{MW}_{t+1})$ the $mwl$-summary of $\mathbb{S}_t \cdot \langle I\rangle$
$S_{t+1} := Update(S_t,\mathbb{MW}_t[1])$
$\mathbb{MW}_{t+1} := \mathbb{MW}_t[2,mwl] \cdot \langle I\rangle$
```

From the *mwl*-summary $(S,\mathbb{MW})$ for itemset A in stream $\mathbb{S}$, we can no longer assume that the max-frequency is in the last entry of the summary S; the fact that this entry is the maximal one among all border positions for $\mathbb{S}_t^{-mwl}$, does not necessarily imply that this last entry is also the maximal one among the borders for the stream $\mathbb{S}$, as the following example illustrates:

Example 6. Let *mwl* be 5. Consider the following three streams (border positions in $\mathbb{S}_t^{-mwl}$ and the minimal window have been indicated by vertical bars and a rectangle respectively):

$$\mathbb{S}^1 = \langle\, |a\ b\ b\ b\ |a\ a\ b\ b\ \boxed{bbbbb}\rangle$$

$$\mathbb{S}^2 = \langle\, |a\ b\ b\ b\ |a\ a\ b\ b\ \boxed{bbbba}\rangle$$

$$\mathbb{S}^3 = \langle\, |a\ b\ b\ b\ |a\ a\ b\ b\ \boxed{bbaaa}\rangle$$

All three streams $\mathbb{S}^i$ have an *mwl*-summary $(S,\mathbb{MW})$ with $S=[(1,1),(5,2)]\}$. Nevertheless, the maximal border $startmax^{mwl}$ for these three streams is respectively 1, 5, and 9.

Therefore, the function Get_mfreq_{mwl} is defined as follows. t denotes the current time and A the target itemset. Let $S_t=([(p_1,a_1),\dots,p_r,a_r],\mathbb{MW}_t)$; l_i will denote p_i-p_{i-1} for $i=1,\dots,r-1$, and $l_r=t-p_r+1$. c_i denotes $\sum_{j=i}^{r} a_j$ for all $i=1,\dots,r$. Furthermore, let $a=count(A,\mathbb{MW}_t)$.

$$Get\_mfreq_{mwl}(A,S_t) := \max_{i=1,\dots,r} \frac{c_i+a}{l_i+mwl}.$$

3.2.3. Example

Fig. 3 shows the summaries and max-frequencies for the itemsets a, b and ab at the various timestamps.

3.3. Minimal frequency threshold

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

Running Example: Max-Freq-Miner^{mwl}

$$\begin{array}{lllllllllllll} & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \mathbb{S} & = & a & b & \emptyset & ab & \emptyset & b & ab & a & b & \emptyset & ab \\ mwl & = & 3 \end{array}$$

t	s_{t-3}	$\mathbb{MW}_t$	*Summaries* $(\mathbb{S}_{t-3})$			*Frequencies*		
			a	b	ab	a	b	ab
3	–	$a,b,\emptyset$	[]	[]	[]	33	33	0
4	a	$b,\emptyset,ab$	[(1, 1)]	[]	[]	50	66	33
5	b	$\emptyset,ab,\emptyset$	[(1, 1)]	[(2, 1)]	[]	40	50	33
6	$\emptyset$	$ab,\emptyset,b$	[(1, 1)]	[(2, 1)]	[]	33	66	33
7	ab	$\emptyset,b,ab$	[(1, 1), (4, 1)]	[(2, 1), (4, 1)]	[(4, 1)]	50	75	50
8	$\emptyset$	b,ab,a	[(1, 1), (4, 1)]	[(2, 2)]	[(4, 1)]	66	66	40
9	b	ab,a,b	[(1, 2)]	[(2, 2), (6, 1)]	[(4, 1)]	66	75	33
10	ab	$a,b,\emptyset$	[(1, 2), (7, 1)]	[(2, 2), (6, 2)]	[(4, 1), (7, 1)]	50	60	28
11	a	$b,\emptyset,ab$	[(1, 2), (7, 2)]	[(2, 2), (6, 2)]	[(4, 1), (7, 1)]	60	66	40

Fig. 3. Running example: max-frequency with a minimal window length (no minimal frequency threshold).

3.3.1. Pruning

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

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

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 $freq(A,(\mathbb{S}_t\cdot\mathbb{B})[p_1,])$ 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.

3.3.2. Max-Freq-Miner$_\sigma$

In order to be able to perform this pruning efficiently, we store and maintain for the summary, the count $total=a_1+a_2+\dots+a_r$. When the left-most border is pruned, *total* is decreased by a_1 to reflect the new total. Algorithm 3 shows how the summary is updated in Max-Freq-Miner$_\sigma$.

Deriving the max-frequency is equivalent to deriving the max-frequency in the case of *no* minimal frequency threshold.

Algorithm 3. *Update*$_\sigma(S_t,I)$ for *one target itemset* A on time $t+1$, *with minimal frequency threshold* σ, and *without minimal window length*.

```
Require: $S_t=[(p_1,a_1),\dots,(p_r,a_r)]$ the summary of $\mathbb{S}_t$
Ensure: $S_{t+1}$ the summary of $\mathbb{S}_t\cdot\langle I\rangle$
total := $a_1+a_2+\dots+a_r$
$S_{t+1}$ := $Update(S_t,I)$
if $A\subseteq I$ then
    $total$ := $total+1$
else
    while $S_{t+1}$ not empty and changing do
        Let $(p,a)$ be the first entry of $S_{t+1}$
        if $\frac{total}{(t+1)-p+1}<\sigma$ then
            $total$ := $total-a$
            remove $(p,a)$ from $S_{t+1}$
```

3.3.3. Example

Fig. 4 shows the summaries and the max-frequencies for the itemsets a, b and ab at the various timestamps.

3.4. Minimal window length and minimal frequency threshold

The last variation of Max-Freq-Miner combines the minimal window length and minimal frequency threshold constraints into one algorithm. We start from Max-Freq-Miner^{mwl}, with the *mwl*-summary, and extend the update algorithm to incorporate the minimal frequency threshold.

3.4.1. Pruning

A border p_i in the summary S of $\mathbb{S}^{-mwl}$ of the *mwl*-summary $(S,\mathbb{MW})$ of $\mathbb{S}$ can be pruned if A is infrequent in $\mathbb{S}^{-mwl}[p_i,]$, but it cannot necessarily be pruned if A is infrequent in $\mathbb{S}[p_i,]$. The reason of this difference with

Running Example: Max-Freq-Miner$_\sigma$

$$\begin{array}{rcccccccccccc} & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \mathbb{S} & = & a & b & \emptyset & ab & \emptyset & b & ab & a & b & \emptyset & ab \\ minfreq & = & 0.4 \end{array}$$

t	s_t	*Summaries*			*Frequencies*		
		a	b	ab	a	b	ab
1	a	$[(1,1)]$	$[]$	$[]$	100	0	0
2	b	$[(1,1)]$	$[(2,1)]$	$[]$	50	100	0
3	$\emptyset$	$[]$	$[(2,1)]$	$[]$	–	50	0
4	ab	$[(4,1)]$	$[(2,1),(4,1)]$	$[(4,1)]$	100	100	100
5	$\emptyset$	$[(4,1)]$	$[(2,2)]$	$[(4,1)]$	50	50	50
6	b	$[]$	$[(2,2),(6,1)]$	$[]$	–	100	–
7	ab	$[(7,1)]$	$[(2,2),(6,2)]$	$[(7,1)]$	100	100	100
8	a	$[(7,2)]$	$[(2,2),(6,2)]$	$[(7,1)]$	100	66	50
9	b	$[(7,2)]$	$[(2,2),(6,2),(9,1)]$	$[]$	66	100	–
10	$\emptyset$	$[(7,2)]$	$[(2,2),(6,3)]$	$[]$	50	60	–
11	ab	$[(7,2),(11,1)]$	$[(2,2),(6,3),(11,1)]$	$[(11,1)]$	100	100	100

Fig. 4. Running example: max-frequency with a minimal frequency (no minimal window length). When the max-frequency is below the threshold, based on the summary a support of 0 will be reported. These situations are denoted by "–" in the column frequency.

the situation without *mwl* is because the argument "*the border can be pruned because every extension that would make it frequent again would itself be even more frequent*" is no longer conclusive to prune the border, as the extension may not satisfy the minimal window length restriction. The following example illustrates this point:

Example 7. Suppose that we have a minimal frequency threshold of $\frac{1}{2}$, and $mwl = 3$. Consider the following stream $\mathbb{S}$ (borders in $\mathbb{S}^{-mwl}$ and minimal window are indicated in the usual way):

$$\langle\, |a\ \ b\ \boxed{a\ \ b\ \ b}\rangle$$

The max-frequency of the singleton itemset a does not exceed the minimal frequency threshold in the stream; its max-frequency equals $\frac{2}{5}$. Suppose now that at timepoint 6, the itemset a arrives, resulting in the stream

$$\langle\, |a\ \ b\ \ |a\,\boxed{b\ \ b\ \ a}\rangle$$

Now the max-frequency with $mwl = 3$ will be $\frac{1}{2}$ and $startmax^{mwl}$ will be the position 1, that was infrequent in $\mathbb{S}$. Notice that this is not in contradiction with the observation that in the case without a minimal window length, position 1 is not a border anymore at timestamp 5, because without the minimal window length, the maximal border would be at position 6, with 100% frequency for item a.

Nevertheless, as the following theorem shows, we can apply pruning on the border positions in $\mathbb{S}^{-mwl}$; if the target itemset A is not frequent in the suffix of $\mathbb{S}^{-mwl}$ starting at position p, then p will never again become the maximal border under the minimal window length constraint in the stream $\mathbb{S}$.

Theorem 3.5. *Let $\mathbb{S}$ be a stream, mwl a positive integer, and $0 \le \sigma \le 1$. Let $p < |\mathbb{S}^{-mwl}|$ be a position in $\mathbb{S}^{-mwl}$. Suppose there exists an extension $\mathbb{B}$ such that*

- *$p = startmax^{mwl}(\mathbb{S} \cdot \mathbb{B})$,*
- *$mfreq(A, \mathbb{S} \cdot \mathbb{B}) = freq(A, \mathbb{S} \cdot \mathbb{B}[p, |\mathbb{S} \cdot \mathbb{B}|]) \ge \sigma$.*

Then p is a frequent border of $\mathbb{S}^{-mwl}$; i.e., $freq(A, \mathbb{S}^{-mwl}[p, |\mathbb{S}^{-mwl}|]) \ge \sigma$.

Proof. Let $\mathbb{MW} = \mathbb{S}[|\mathbb{S}| - mwl + 1, |\mathbb{S}|]$ be the minimal window of $\mathbb{S}$. From Theorem 3.4 it follows that p must be a border. We will prove that p is also a frequent border in $\mathbb{S}^{-mwl}$, by contradiction. Hence, suppose that p is not a frequent border in $\mathbb{S}^{-mwl}$ for A; i.e.,

$$freq(A, \mathbb{S}^{-mwl}[p, |\mathbb{S}^{-mwl}|]) < \sigma.$$

Suppose there exists an extension $\mathbb{B}$ as in the statement of the theorem. Then, $mfreq(A, \mathbb{S} \cdot \mathbb{B}) > \sigma$, this implies that

$$freq(A, \mathbb{MW} \cdot \mathbb{B}) > freq(A, \mathbb{S}[p, |\mathbb{S}|] \cdot \mathbb{B}).$$

Therefore, p is not the starting point of the maximal window, since suffix $\mathbb{MW} \cdot \mathbb{B}$ satisfies the minimal window length restriction and has a higher frequency. This is a contradiction and hence the theorem is proven. □

3.4.2. Max-Freq-Miner$_\sigma^{mwl}$

By Theorem 3.5 we know that a border for target itemset A in stream $\mathbb{S}$ is also a frequent border of $\mathbb{S}^{-mwl}$. We can thus maintain a summary for target itemset A over the stream $\mathbb{S}^{-mwl}$ as if only the minimal frequency threshold σ is set. Algorithm 4 shows that implementing procedure $Update_\sigma^{mwl}((S_t, \mathbb{MW}), I)$ is simple, because it is largely the same as $Update_\sigma$. Extracting all max-frequent itemsets is demonstrated in Algorithm 6.

Algorithm 4. $Update_\sigma^{mwl}((S_t, \mathbb{MW}), I)$ for *one target itemset* A on time $t+1$, *with minimal window length mwl*, and *with minimal frequency* σ

```
Require: $(S_t, \mathbb{MW})$, the $mwl$-summary of $\mathbb{S}_t$
Ensure: $(S_{t+1}, \mathbb{MW}')$ the $mwl$-summary of $\mathbb{S}_t \cdot \langle I \rangle$
    $S_{t+1} := Update_\sigma(S_t, \mathbb{MW}[1])$
    $\mathbb{MW}' := \mathbb{MW}[2, mwl] \cdot \langle I \rangle$
```

3.4.3. Example

Fig. 5 shows the summaries and the max-frequencies of the itemsets a, b and ab at various timestamps.

Running Example: Max-Freq-Miner$_\sigma^{mwl}$

$$\begin{array}{rcccccccccccc} & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \mathbb{S} & = & a & b & \emptyset & ab & \emptyset & b & ab & a & b & \emptyset & ab \\ mwl & = & 3 \\ minfreq & = & 0.4 \end{array}$$

t	s_{t-3}	$\mathbb{MW}_t$	*Summaries* ($\mathbb{S}_{t-3}$)			*Frequencies*		
			a	b	ab	a	b	ab
3	–	$a, b, \emptyset$	$[]$	$[]$	$[]$	–	–	–
4	a	$b, \emptyset, ab$	$[(1,1)]$	$[(2,1)]$	$[]$	50	66	–
5	b	$\emptyset, ab, \emptyset$	$[]$	$[(2,1)]$	$[]$	40	50	–
6	$\emptyset$	$ab, \emptyset, b$	$[(4,1)]$	$[(2,1),(4,1)]$	$[(4,1)]$	–	66	–
7	ab	$\emptyset, b, ab$	$[(4,1)]$	$[(2,2)]$	$[(4,1)]$	50	75	50
8	$\emptyset$	b, ab, a	$[]$	$[(2,2),(6,1)]$	$[]$	66	66	40
9	b	ab, a, b	$[(7,1)]$	$[(2,2),(6,2)]$	$[(7,1)]$	66	75	–
10	ab	$a, b, \emptyset$	$[(7,2)]$	$[(2,2),(6,2)]$	$[(7,1)]$	50	60	–
11	a	$b, \emptyset, ab$	$[(7,2)]$	$[(2,2),(6,2),(9,1)]$	$[]$	60	66	40

Fig. 5. Running example: max-frequency with a minimal window length and a minimal frequency. When the max-frequency is below the threshold, based on the summary a support of 0 will be reported. These situations are denoted by "–" in the column frequency.

4. Mining all itemsets

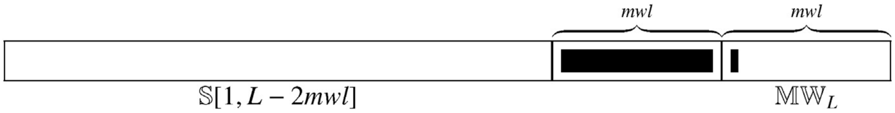
Until now, we focused on mining a single frequent itemset. Of course, in reality, the goal is to find *all* frequent itemsets in the stream. A straightforward way to do this is to apply Max-Freq-Miner$_{\sigma}^{mwl}$ for all itemsets at the same time. Hence, for all itemsets A we need to store the positions in the stream that can become a maximal border. Theorem 3.5 reduces the number of borders we need to store: either the minimal window is the maximal window, or the maximal border is a maximal border in $\mathbb{S}^{-mwl}$. Hence, it suffices to store the minimal window, and, for every itemset, its summary in $\mathbb{S}^{-mwl}$. This straightforward approach, however, is not practical. The reason is as follows: suppose that a transaction T with n items leaves the minimal window, and enters $\mathbb{S}^{-mwl}$. Then, for each of the 2^n subsets of T we have to start a new summary, even if a subset is not frequent in $\langle T \rangle \cdot \mathbb{MW}$. For large transactions, the number of new summaries would hence become intractable.

The next lemma will explain how we can avoid this exponential blow-up; if we are willing to store the last $2mwl$ transaction of the stream; that is, twice the minimal window, it suffices to only start a summary for those subsets of T that are frequent in $\mathbb{S}^{-mwl}$ just before T left the stream. Usually the number of frequent itemsets in the minimal window will be much smaller than the number of subsets in an arbitrary transaction. Of course, if the minimal support is set too low, for example, lower than $1/mwl$, we may still experience an exponential blowup. Nevertheless, the lemma below reduces the number of summaries to be generated from "all subsets of T" to "all subsets of T that were frequent in the last minimal window".

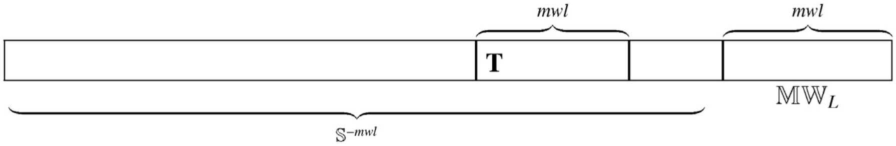
In other words, if an itemset A is mwl-frequent in $\mathbb{S}_t$, then either the maximal border p of A in $\mathbb{S}_t$ is among the last $2mwl$ positions of the stream, or A is frequent in the sub-stream $\mathbb{S}_t[p,p+mwl-1]$. In the latter case, at time point $p+mwl$, A was a subset of the transaction leaving the minimal window and A was frequent in the minimal window at time point $p+mwl-1$. This situation is also depicted in the following figure:

Border position p of itemset A:

- Either $L-2mwl < p \leq L-mwl+1$ (border at one of the black positions)



- or: $A \subseteq T = S_p$, A frequent in $\mathbb{S}[p,p+mwl-1]$ (border at **T**)



Lemma 4.1. *Let $\mathbb{S}$ be a stream of length L. $\mathbb{MW}_t$ denotes $\mathbb{S}[t-mwl+1,t]$; i.e., the minimal window at the time t.*

Let $mfreq^{mwl}(A,\mathbb{S}) \geq \sigma$, with q the starting point of the maximal window; i.e.,

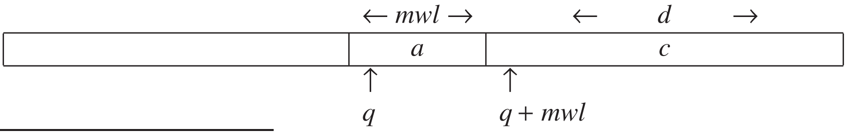
$$q := startmax^{mwl}(A,\mathbb{S})$$

then,

- *either $L-2mwl+1 < q \leq L-mwl$; i.e., q left the minimal window less than mwl steps ago, or*
- *A is frequent in $\mathbb{S}[q,q+mwl-1]$; that is, A is frequent in $\mathbb{MW}_t$ for $t=q+mwl-1$ (S_q leaves the minimal window at time $t+1$).*

Proof. Clearly $q \leq L-mwl$, otherwise the minimal window length requirement would be violated for the maximal window $\mathbb{S}[q,L]$. Therefore, either the first statement holds, or $q \leq L-2mwl+1$. We will show that if $q \leq L-2mwl+1$, then the second statement holds.

Denote the number of occurrences of A in $\mathbb{S}[q,q+mwl-1]$ by a, the number of occurrences of A in stream $\mathbb{S}[q+mwl,L]$ by c, and the length of $\mathbb{S}[q+mwl,L]$ by d:



We know that $freq(A,\mathbb{S}[q,L]) \geq \sigma$, i.e., $(a+c)/(mwl+d) \geq \sigma$, and we have to show that the frequency of A in $\mathbb{S}[q,q+mwl-1]$; i.e., a/mwl, is also greater than or equal to σ. Assume, for the sake of contradiction, that $a/mwl < \sigma$. We then have

$$\frac{a+c}{mwl+d} \geq \sigma > \frac{a}{mwl} \Rightarrow \frac{c}{d} > \frac{a}{mwl} \Leftrightarrow \frac{c}{d} > \frac{a+c}{mwl+d}.$$

This result shows that A is even more frequent in $\mathbb{S}[q+mwl,L]$ than it is in $\mathbb{S}[q,L]$, which is not possible because q is the starting point of the maximal window of length at least mwl for A in $\mathbb{S}$, and $|\mathbb{S}[q+mwl,L]| \geq mwl$, since $q \leq L-2mwl+1$. Hence, our assumption is wrong, meaning that $a/mwl \geq \sigma$.

4.1. Algorithm

Hence, we do not need to maintain a summary for all itemsets, but we only need to store those borders for itemset A that were once the starting point of a minimal window in which A was frequent. Furthermore, we can prune any border that does not satisfy the minimal frequency threshold in $\mathbb{S}^{-mwl}$. The only price we have to pay is that we need to check for the frequent itemsets in the mwl windows $\mathbb{S}[L-2mwl+2,L]$, …, $\mathbb{S}[L-mwl,L]$.

The algorithm to update the summary when a new itemset I arrives is as follows: for every itemset A for which *we are maintaining a summary*, update the summary with the itemset that leaves the minimal window. Next we prune the infrequent borders. We start pruning at the oldest border and move towards the youngest one, because the frequency, in $\mathbb{S}^{-mwl}$ of the borders is strictly increasing. Thus we can stop pruning if we encounter a frequent border. Remember that we keep track of the sum $total = a_1 + \cdots + a_r$ in every summary, to efficiently assess the frequency of a border. Then, for all subsets of I 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 $q \leq L-2mwl+1$. Furthermore, we always keep the last $2 \cdot mwl-1$ 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* $\mathbb{S}[L-2mwl+2,L]$, …, $\mathbb{S}[L-mwl,L]$. This can be done efficiently with a small adaptation to efficient incremental algorithms that have already been proposed in the literature [20], or with an incremental version of existing frequent itemsets miners. Algorithm 5 shows the procedure for updating all summaries. Fig. 6 illustrates the $UpdateAll_{mwl,\sigma}$ on the running example. Notice, e.g., that when at step 7, $s_4 = ab$ is appended to $\mathbb{S}^{mwl}$. In the original algorithm Max-Freq-Miner$_\sigma^{mwl}$, this arrival resulted in a summary being started for all its subsets that did not yet have a summary; i.e., for a and ab (see Fig. 6). For Max-Freq-Miner-All$_\sigma^{mwl}$, however, these summaries are not created as a and ab were not frequent in $\mathbb{MW}_6 = \langle ab \ \emptyset \ b \rangle$. Algorithm 6 shows how to extract the max-frequent itemsets and their max-frequency.

Algorithm 5. $UpdateAll_{mwl,\sigma}((S_t^{A_1}, \ldots, S_t^{A_n}, \mathbb{MW},)\mathbb{MW}_2, I)$ for *all itemsets* on time $t+1$, *with minimal window length mwl*, and *with minimal frequency* σ.

```
Require: $(S_t^{A_1}, \ldots, S_t^{A_n}, \mathbb{MW})$, the $mwl$-summary for all itemsets of $\mathbb{S}_t$
and $\mathbb{MW}_2$ the second minimal window
Ensure: $(S_{t+1}^{B_1}, \ldots, S_t^{B_m}, \mathbb{MW}')$ the $mwl$-summary for all itemsets of
$\mathbb{S}_t \cdot \langle I \rangle$ and $\mathbb{MW}_2{}'$ the second minimal window of $\mathbb{S}_t \cdot \langle I \rangle$
    $\mathbb{MW}' := \mathbb{MW}[2,mwl] \cdot \langle I \rangle$
    $\mathbb{MW}_2{}' := \mathbb{MW}_2[2,mwl] \cdot \mathbb{MW}[1]$
    for all $A$ frequent in $\mathbb{MW}_t$ and $A \subseteq I$ do
        if $S_t^A$ exists then
            $S_{t+1}^A := Update_\sigma(S_t^A, \mathbb{MW}[1])$
        else
            $S_{t+1}^A := [(t+1, 1)]$
    for all $A$ such that $S_t^A$ exists and $A$ is not frequent in $\mathbb{MW}_t$ do
        $S_t^A := Update_\sigma(S_t^A, \mathbb{MW}[1])$
```

Algorithm 6. Pseudo-code of $Get\_mfreq_{mwl,\sigma}$ $((S_t^{A_1}, \ldots, S_t^{A_n}, \mathbb{MW}), \mathbb{MW}_2, \sigma)$ for extracting the max-frequent itemsets with minimal window length mwl and with minimal

Running Example: Max-Freq-Miner-All$_\sigma^{mwl}$

		1	2	3	4	5	6	7	8	9	10	11
$\mathbb{S}$	=	a	b	$\emptyset$	ab	$\emptyset$	b	ab	a	b	$\emptyset$	ab
mwl	=	3										
$minfreq$	=	0.4										

t	s_{t-3}	$\mathbb{MW}_t$	*Summaries* $(\mathbb{S}_{t-3})$			*Frequencies*		
			a	b	ab	a	b	ab
3	–	$a, b, \emptyset$	[]	[]	[]	–	–	–
4	a	$b, \emptyset, ab$	$\underline{[]}$	[]	[]	∗50	66	–
5	b	$\emptyset, ab, \emptyset$	$\underline{[]}$	[(2, 1)]	[]	∗40	50	–
6	$\emptyset$	$ab, \emptyset, b$	[]	[(2, 1)]	[]	–	66	–
7	ab	$\emptyset, b, ab$	$\underline{[]}$	[(2, 1)(4, 1)]	$\underline{[]}$	∗50	75	∗50
8	$\emptyset$	b, ab, a	$\underline{[]}$	[(2, 1)(4, 1)]	$\underline{[]}$	66	66	∗40
9	b	ab, a, b	[]	[(2, 1)(4, 1)(6, 1)]	[]	66	75	–
10	ab	$a, b, \emptyset$	[(7, 1)]	[(2, 1)(4, 1)(6, 2)]	$\underline{[]}$	50	60	–
11	a	$b, \emptyset, ab$	[(7, 2)]	[(2, 1)(4, 1)(6, 2)]	$\underline{[]}$	60	66	∗40

∗: an element s_{t-3} is only added to the summary of $\mathbb{S}_{t-3}$ at timestamp t if it was frequent in the minimal window $\mathbb{MW}_{t-1}$. Sometimes the frequency of an itemset is reached within the last mwl elements of the summarized stream $\mathbb{S}_{t-3}$, and is not a border in its summary. These cases are marked by a star.

Fig. 6. Running example: max-frequency with a minimal window length and a minimal frequency. When the max-frequency is below the threshold, based on the summary a support of 0 will be reported. These situations are denoted by "–" in the column frequency. The algorithm is modified for mining all itemsets; we only start a summary for an itemset entering $\mathbb{S}^{-mwl}$ that was frequent in $\mathbb{MW}$ the step before. The summaries for which this implies a difference are underlined.

frequency threshold σ. Parameter $\mathbb{MW}^2$ is the double minimal window.

```
1:  frequentItemsets :=[]
2:  freqItemsets := {(A,p,f) | $mfreq^{mwl}(A, \mathbb{MW}_2 \cdot \mathbb{MW}) = f \geq \sigma, p = startmax^{mwl}(A, \mathbb{MW}_2 \cdot \mathbb{MW})$}
3:  for all A such that $S_t^A$ exists or $A \in$ freqItemsets do
4:    if $A \in$ freqItemsets then
5:      (mFreqPos, mFreq) :=freqItemsets[A]
6:    else
7:      (mFreqPos, mFreq) :=(0, 0)
8:    if $S_t^A$ exists then
9:      $c := count(A, \mathbb{MW})$
10:     for $(p,a) \in S_t^A$ do
11:       $c := c + a$
12:       if mFreq $\leq \frac{c}{t-p+1}$ then
13:         mFreqPos :=p
14:         mFreq :=$\frac{c}{t-p+1}$
15:   if mFreq $> \sigma$ then
16:     frequentItemsets :=frequentItemsets · (A, mFreqPos, mFreq)
17: return frequentItemsets
```

4.2. Complexity

Let R be the maximum number of borders in any summary while mining a stream. Let F be the number of frequent itemsets in the stream with respect to the minimal window length *mwl* and the minimal frequency threshold σ. Both the $UpdateAll_{mwl,\sigma}$ and $Get\_mfreq_{mwl,\sigma}$ need to be executed at each timestamp of the stream.

The procedure $UpdateAll_{mwl,\sigma}$ consists of two main for-loops. The first for-loop (lines 2–6) can be executed at most $O(F)$ times. In the worst case, the code in this for-loop needs to investigate every border of the current summary (if $Update_\sigma$ is executed). As a result, the first for-loop takes at most $O(FR)$ time. Note however, that in practice, only one or zero borders will be pruned at a time. Also note that our experiments (on the blocks-10 and compounds-60 streams) show that there are on average only about 10 borders present in all summaries. Thus the real-world complexity will be in the order of $O(F)$ rather than $O(FR)$. The second for-loop (line 7) can also execute at most $O(F)$ times (if all frequent itemsets are currently frequent). In the worst-case all borders need to be removed, implying $O(R)$ time. In total, the procedure $UpdateAll_{mwl,\sigma}$ consumes in the worst-case $O(FR)$ time.

An important step in $Get\_mfreq_{mwl,\sigma}$ is finding max-frequent itemsets in $\mathbb{MW}_2 \cdot \mathbb{MW}$. Section 7.1.2 describes how we can find the max-frequent itemsets in $\mathbb{S}_t$ $[t-2mwl+1,]$ such that the starting position is smaller than or equal to $t-mwl+1$. This procedure resembles the iterative candidate-generation-then-pruning structure of Apriori. This strategy traverses a portion of the itemset-lattice. The number of states visited in the lattice is polynomial in F, the number of frequent itemsets. Visiting a new state requires merging at least two position summaries. Let G be the size of the largest frequent itemset. We can estimate the running time of this step by $O(mwl \times G \times poly(F))$, in which $poly(F)$ is a polynomial of F.

There are at most $O(F)$ frequent itemsets, thus the for-loop (lines 3–16) can be executed at most $O(F)$ times. Within the for-loop the procedure loops (lines 10–14) over all borders of the summary of the current itemset (if the summary exists). The total complexity of the for-loop is thus $O(mwl \times G \times poly(F) + FR)$.

The combined worst-case time complexity is thus $O(mwl \times G \times poly(F) + FR)$. In Section 6 we show that the number of borders grows (in the worst-case) sublinearly with respect to the length of the stream. In Section 7 we show that our algorithm can easily process long streams (+4 million transactions) with large itemsets (about 85 items). In our experience the performance of the algorithms depends largely on the number of frequent itemsets.

4.3. Optimizations

Whenever a non-target itemset arrives at the head of the stream, all four algorithms try to prune borders from the summary. Even if we could have predicted at the previous timepoint that such an attempt would be in vain.

Example 8. Suppose we are mining the following stream $\langle |a\ b\ b\ |a\ a\ a\ a\ b\rangle$ with Max-Freq-Miner. The border positions for itemset a are indicated by vertical bars. The frequency between the first and second border is $\frac{1}{3}$. From the second border to the end of the stream, the frequency is equal to $\frac{4}{5}$. The second border can only be pruned from the summary if there is a before block with a higher frequency than an after block. Concretely, if x non-target itemsets extend the stream, and thus $\frac{1}{3} \geq 4/(5+x)$ holds, the second border can be pruned. In other words, at this timepoint we can be certain that the second border will remain in the summary for at least another seven more timepoints ($x \geq 7$).

Therefore, we introduce the *checkscheduler*, for delaying prune checks as long as possible. Intuitively, this optimization should prove most valuable in the context of a long minimal window and a low minimal frequency threshold. Because borders will remain put in the summary for quite some time.

One could say that the checkscheduler is *lazy* in trying to prune borders from a summary. This laziness can be taken one step further as the following example shows.

Example 9. Suppose we are mining the following stream $\langle |a\ a\ b\ b\ |a\rangle$ with Max-Freq-Miner. Again, there are two borders, indicated by the vertical bars. The checkscheduler would, at this timepoint, predict that the second border might be pruned at the next timepoint. Indeed, if the next itemset arriving in the stream is a non-target itemset, the frequency between the border equals the frequency from the second border to the end of the stream. However, if the next three itemsets arriving in the stream are also target itemsets, the checkscheduler could have been even lazier in the sense that it could have waited until after the arrival of the last target itemset to make a prediction. This would result in just a single prediction ($x=4$), instead of four predictions ($x=1$, $x=2$, $x=3$ and $x=4$), one after the arrival of each of the four target itemsets.

A sequence of itemsets is called a *burst* of itemset A if each of the itemsets is a superset of A. If the checkscheduler waits until the end of a burst to predict the timepoint at which a border might be pruned, it is called *lazy*. We call this optimization of the checkscheduler *lazy handling*. Intuitively, lazy handling will perform best if a stream contains many bursts.

During experiments with our software prototype, we found that the lazy checkscheduler outperforms the Max-Freq-Miner$_\sigma^{mwl}$ on bursty streams. The results are listed in Section 7.

The length of the predicted minimal delay depends on the pruning methods available to an algorithm and depends on the exact value of the minimal window length and minimal frequency threshold. For each of the four algorithms we derive the equation to calculate the minimal delay.

The Max-Freq-Miner algorithm prunes a border if there exists a before block with a frequency that is higher or equal to the frequency of an after block. Let $[(a_1,p_1),\ldots,(a_r,p_r)]$ be the summary of stream $\mathbb{S}_t$. Then we can prune the youngest border (a_r,p_r) if

$$\frac{a_{r-1}}{p_r-p_{r-1}} \geq \frac{a_r}{t-p_r+1+x}.$$

Indeed, the frequency of the youngest border drops if non-target itemsets are added to the top of the stream. We find the smallest satisfying integer value of x by means of the following formula:

$$x \geq \frac{a_r(p_r-p_{r-1})}{a_{r-1}}-(t-p_r+1).$$

If the *minimal window length* is set, *Max-Freq-Miner*^{mwl} keeps a summary for $\mathbb{S}_t^{-mwl}$. Calculating the minimal delay is thus equivalent to the calculation for Max-Freq-Miner.

Two pruning criteria are employed when a *minimal frequency threshold* is set. The first is the same as in the Max-Freq-Miner algorithm. Denote the delay for this first criterion by x_1. The second pruning criterion prunes the *oldest border* if its frequency in $\mathbb{S}_t$ drops below σ. Again, the frequency of a border drops fastest if non-target itemsets arrive: at least x_2 non-target itemsets have to arrive to make the oldest border prunable:

$$\frac{total}{t-p_1+1+x_2} < \sigma.$$

Remember that *total* denotes the sum of all the itemset counts in the summary, i.e., $total = a_1 + \cdots + a_r$. Rewriting the previous equation results in

$$\frac{total}{\sigma}-(t-p_1+1) < x_2.$$

Now, we can delay checking for a before-after block violation until $t+x_1$ and for an infrequent oldest border until timepoint $t+x_2$.

When both the *minimal window length and the minimal frequency threshold* have been set, the minimal delay calculation is identical to the calculation for the Max-Freq-Miner$_\sigma$ algorithm, except that it is now carried out on the summary of $\mathbb{S}_t^{-mwl}$.

5. Greater border efficiency

In this section we will show some negative results w.r.t. exploiting the relation between the itemsets in order to increase the efficiency of mining all max-frequent itemsets. More concretely, we will show that even though the max-frequency measure is anti-monotone, the border positions of super- and subsets do not possess any straightforward relation; for any collection of sets we can find a stream and a position in that stream such that exactly the given collection of itemsets has a border at that position. The collection of itemsets does not need to be subset-closed and is not restricted in any way. As an intermediate step towards this result we first show that the border positions of the items do not contain enough information to derive the border positions of larger itemsets, hence invalidating the extensions to borders of state-of-the-art techniques for itemset frequency based on maintaining tid-lists of items [24].

5.1. Item borders only

In this subsection we answer the following question: "*Do we need to keep track of the summaries of all itemsets or is it possible to keep track of the item-summaries* (*the summaries of itemsets of length* 1) *and reconstruct the itemset-summaries* (*the summaries of itemsets of length greater than* 1) *from the item-summaries whenever needed?*" The next example will answer this question negatively by showing two streams in which all item summaries are the same for both streams, yet the summary for the itemset composed of the individual items is not. As such we can conclude that the item summaries do not contain enough information to derive the border positions of non-singleton itemsets.

Example 10. Suppose the following two streams: $\mathbb{S}^1 = \langle a\ b\ ab\ c\ c\rangle$ and $\mathbb{S}^2 = \langle a\ b\ a\ bc\ c\rangle$. The item-summaries for both streams are equal: $S_a^1 = S_a^2 = [(1,2)]$, $S_b^1 = S_b^2 = [(2,2)]$ and $S_c^1 = S_c^2 = [(4,2)]$. However, the summary for the itemset ab is different in both streams: $S_{ab}^1 = [(3,1)]$ and $S_{ab}^2 = []$, as is the summary of itemset bc: $S_{bc}^1 = []$ and $S_{bc}^2 = [(4,1)]$.

The underlying reason for this result is in fact that in the summary of an itemset we only keep enough information to predict future maximal border positions of that itemset, effectively reducing the storage needed. Storing all occurrences of all items clearly is not an option as this comes down to storing the complete stream into the memory.

5.2. Border closure

In frequent itemset mining, a set of itemsets is frequently abbreviated by its top elements. The top-elements are those itemsets that are not a strict subset of another element in the set. The set of these top elements is called the *set border*.[2] In frequent itemset mining, where

[2] Note that in the literature this is called simply border. However, as we already use the term border and to avoid a naming clash in our terminology, we call it the set border.

all subsets of itemsets are frequent if the itemset itself is frequent, the set border of a set of itemsets is in many cases exponentially smaller than the original set.

Secondly, we observe: if at timestamp p itemset ab arrives at the top of the stream, then p will be a border for all the subsets: a, b and ab.

Combining both observations leads to the question: "*Can we share borders between the summaries of related itemsets*?" For example: if itemset ab arrives, the current timestamp can be a border for ab, a and b at the same time.

Unfortunately, we will prove that for every arbitrary collection of itemsets $\mathcal{B}$ there exists a stream and a position p in that stream such that the itemsets that have a border at p is exactly $\mathcal{B}$. This theorem shows that there is no relation between the border positions of the itemsets; at a given timestamp literally any combination of itemsets can have a border. This shows that there is no obvious way to combine the summaries of the different itemsets.

Example 11. Let $A=abcd$ and $\mathcal{B}=\{abc,a\}$. In the following stream at the indicated position, exactly the itemsets abc and a have a border position and no other itemset:

$$\langle abcd\ \ ab\ \ ac\ \ bc\ \ \emptyset\ \ \emptyset\ \ |abcd\ \ abc\ \ a\ \ a\ \ \emptyset\ \ \emptyset\rangle$$

A first step in the proof consists in identifying the bag $\mathcal{L}$ of transactions that will come before the border position, and the bag of transactions $\mathcal{R}$ occurring after the border position. Remember that a position p is a border for an itemset if and only if its frequency in every block ending at position $p-1$ is less than its frequency in every block starting at position p (Theorem 3.2). In our construction this property will translate into the requirement that for every set in $\mathcal{B}$, the frequency in $\mathcal{R}$ (the transactions to the right of the border) must be larger than the frequency in $\mathcal{L}$. For the other sets, the opposite must hold.

We will denote the number of times a set J occurs as a subset of an element in a bag $\mathcal{R}$ by $count(J,\mathcal{R})$; i.e.,

$$count(J,\mathcal{R}) := \sum_{J\subseteq R}\mathcal{R}(R),$$

where $\mathcal{R}(R)$ denotes the multiplicity of the set R in $\mathcal{R}$.

Lemma 5.1. *Let A be an itemset and $\mathcal{B}\subseteq 2^A$. Then there exist bags $\mathcal{L}$ and $\mathcal{R}$ such that for all $J\subseteq A$ it holds that*

$$J\in\mathcal{B}\Leftrightarrow count(J,\mathcal{L})<count(J,\mathcal{R}).$$

Proof. We will prove this lemma by induction on the number of elements in $\mathcal{B}$.

The base case $|\mathcal{B}|=0$ is trivially fulfilled by $\mathcal{L}=\mathcal{R}=\{\}$.

In the general case, let J be a minimal set in $\mathcal{B}$ w.r.t. set inclusion. By induction we assume that the lemma holds for $\mathcal{B}\backslash\{J\}$. Let $\mathcal{L}'$ and $\mathcal{R}'$ be the bags. Let $n=count(J,\mathcal{L}')-count(J,\mathcal{R}')$. Then, add $n+1$ copies of J to $\mathcal{R}'$ to get $\mathcal{R}$, and if $|J|>1$, for all $j\in J$, add $n+1$ copies of all sets $J\backslash\{j\}$ to $\mathcal{L}'$ to get $\mathcal{L}$, otherwise $\mathcal{L}=\mathcal{L}'$. The resulting $\mathcal{L}$ and $\mathcal{R}$ satisfy the conditions of the lemma.

Example 12. Consider again $A=abcd$. We construct the bags for $\mathcal{B}=\{abc,a\}$ using the inductive procedure given in the proof:

$\mathcal{B}$	$\mathcal{L}$	$\mathcal{R}$
{}	{}	{}
{abc}	{ab,ac,bc}	{abc}
{abc,a}	{ab,ac,bc}	{abc,a,a}

Hence, the following bags $\mathcal{L}$ and $\mathcal{R}$ satisfy the inequalities of the lemma for $\mathcal{B}=\{a,abc\}$:

$$\mathcal{L} := \{ab,ac,bc\},$$

$$\mathcal{R} := \{a,a,abc\}.$$

Indeed; for example: $count(a,\mathcal{L})=2<count(a,\mathcal{R})=3$, and $count(ab,\mathcal{L})=2\nless count(a,\mathcal{R})=1$. Notice incidentally that these bags correspond to the example before; in the stream given in that example, the non-empty transactions to the left of the position p are those in $\mathcal{L}$, and the ones on the right are those in $\mathcal{R}$.

For the proof it is important to get a grip on the before and after block for position p in which an itemset reaches respectively its maximal and minimal frequency. The following two lemmas are essential. The first extension lemma shows that we can extend streams by appending a sequence of empty transactions such that for every subset of the first transaction of the stream, the suffix of the stream in which the frequency of the itemset is maximized among all suffixes, is the whole extended stream itself. We will call such a stream *suffix-maximized*. We will use $\mathbb{Z}_n$ to denote a stream consisting of n empty transactions; i.e.,

$$\mathbb{Z}_n := \langle\overbrace{\emptyset\ \emptyset\ \cdots\ \emptyset}^{\times n}\rangle$$

Lemma 5.2. *Let $\mathbb{S}$ be a stream of length m. There exists an n_0 such that for all $n\geq n_0$ it holds that for any itemset $J\subseteq\mathbb{S}[1]$,*

$$\max_{i=1,\ldots,m+n} freq(J,last(i,\mathbb{S}\cdot\mathbb{Z}_n))=freq(J,\mathbb{S}\cdot\mathbb{Z}_n).$$

Proof. Let J be any itemset and split $\mathbb{S}$ in an arbitrary way: $\mathbb{S}=\mathbb{S}_l\cdot\mathbb{S}_r$. Let $count(J,\mathbb{S}_l)=j_l$, $count(J,\mathbb{S}_r)=j_r$, $|\mathbb{S}_l|=m_l$, and $|\mathbb{S}_r|=m_r$. Notice that, since $J\subseteq\mathbb{S}[1]$, $j_l\geq 1$. Let us consider the effect of extending $\mathbb{S}$ by appending $\mathbb{Z}_n$. We get the following frequencies for J in the suffix $\mathbb{S}_r\cdot\mathbb{Z}_n$ and in the whole stream $\mathbb{S}_l\cdot\mathbb{S}_r\cdot\mathbb{Z}_n$:

$$freq(J,\mathbb{S}_r\cdot\mathbb{Z}_n)=\frac{j_r}{m_r+n},$$

$$freq(J,\mathbb{S}_l\cdot\mathbb{S}_r\cdot\mathbb{Z}_n)=\frac{j_l+j_r}{m_l+m_r+n}.$$

The ratio between these two frequencies (whole stream divided by suffix $\mathbb{S}_r\cdot\mathbb{Z}_n$) is

$$\frac{j_l+j_r}{m_l+m_r+n}\Big/\frac{j_r}{m_r+n}=\frac{j_l+j_r}{j_r}\frac{m_r+n}{m_l+m_r+n}$$
$$=\left(1+\frac{j_l}{j_r}\right)\left(\frac{m_r+n}{m_l+m_r+n}\right)$$

It is easy to see that with increasing n, this ratio will converge in the limit to $1+(j_l/j_r)$, which is strictly greater than 1, since $j_l \geq 1$. Therefore, from a certain index n_0^j on, the frequency of J in the whole stream will be larger than in the suffix $\mathbb{S}_r \cdot \mathbb{Z}_n$. Since the choice of $J \subseteq \mathbb{S}[1]$ and the division $\mathbb{S}$ as $\mathbb{S}_l \cdot \mathbb{S}_r$ were arbitrary, we can extend the result to all subsets of $\mathbb{S}[1]$ and divisions of $\mathbb{S}$ by taking the maximum of all n_0^j, which proves the lemma. □

Example 13. Consider the stream $\mathbb{S} \coloneqq \langle abcd\ ab\ ac\ bc \rangle$. For this stream it suffices to append $n_0 = 2$ or more empty transactions to make the maximal suffix equal to the whole extended stream for all subsets of *abcd*. Let's consider the frequencies of the subsets of *abcd* in the suffixes of the stream $\mathbb{S} \cdot \mathbb{Z}_2$ (since all transactions with *d* also contain *abc*, there is no need to display the results of the itemsets with a *d*, except for *abcd* itself):

Suffix	Frequencies						
	a	*b*	*c*	*ab*	*ac*	*bc*	*abcd*
$\langle abcd\ ab\ ac\ bc\ \emptyset\ \emptyset \rangle$	**1/2**	**1/2**	**1/2**	**1/3**	**1/3**	**1/3**	**1/6**
$\langle ab\ ac\ bc\ \emptyset\ \emptyset \rangle$	2/5	2/5	2/5	1/5	1/5	1/5	0
$\langle ac\ bc\ \emptyset\ \emptyset \rangle$	1/4	1/4	**1/2**	0	1/4	1/4	0
$\langle bc\ \emptyset\ \emptyset \rangle$	0	1/3	1/3	0	0	**1/3**	0
$\langle \emptyset\ \emptyset \rangle$	0	0	0	0	0	0	0
$\langle \emptyset \rangle$	0	0	0	0	0	0	0

The second extension lemma shows the opposite of the first lemma; any stream can be extended by appending a sequence of empty transactions such that for every subset of the first transaction of the stream, the *prefix* of the stream in which the frequency of the itemset is *minimized* among all prefixes, is the whole extended stream itself. We will call such a stream *prefix-minimized*.

Lemma 5.3. *Let $\mathbb{S}$ be a stream of length m. There exists an n_0 such that for all $n \geq n_0$ it holds that for any itemset $J \subseteq \mathbb{S}[1]$,*

$$freq(J, \mathbb{S} \cdot \mathbb{Z}_n) = \min_{i=1,\dots,m+n} freq(J, (\mathbb{S} \cdot \mathbb{Z}_n)[1,i]).$$

Proof. The lemma can be proven in a similar way as the previous lemma: clearly for any itemset it holds that its frequency in any prefix of a stream $\mathbb{S} \cdot \mathbb{Z}_n$ that contains the complete $\mathbb{S}$ is more than the frequency of that itemset in $\mathbb{S} \cdot \mathbb{Z}_n$; adding empty transactions only decreases the frequency. Therefore we only consider prefixes that do not contain the full $\mathbb{S}$. Splitting a stream $\mathbb{S}$ arbitrarily into $\mathbb{S}_l \cdot \mathbb{S}_r$ gives the following frequencies for the prefix $\mathbb{S}_l$ and the whole stream $\mathbb{S} \cdot \mathbb{Z}_n$:

$$freq(J, \mathbb{S}_l) = \frac{j_l}{m_l},$$

$$freq(J, \mathbb{S} \cdot \mathbb{Z}_n) = \frac{j_l + j_r}{m_l + m_r + n}.$$

The ratio between the frequency in the whole stream and in the prefix is

$$\frac{j_l + j_r}{m_l + m_r + n} \bigg/ \frac{j_l}{m_l} = \left(1 + \frac{j_r}{j_l}\right)\left(\frac{m_l}{m_l + m_r + n}\right),$$

which goes to 0 in the limit, indicating that the frequency in the whole stream becomes eventually less than that in the prefix. Again, since the subset J of $\mathbb{S}[1]$ and the prefix were arbitrary, the result can be extended to all subsets and divisions and hence the lemma is proven. □

Example 14. Consider the stream $\mathbb{S} \coloneqq \langle abcd\ abc\ a\ a \rangle$. For this stream it suffices to append $n_0 = 1$ or more empty transactions to make the minimal prefix for all subsets of *abcd* equal to the whole stream. The frequencies of the subsets of *abcd* in the prefixes of the extended stream are as follows (since all transactions with *d* contain *abcd*, there is no need to display the results of the itemsets with *d*, except for *abcd* itself; and every transaction with *ab*, *ac*, or *bc* contains *abc* as well; therefore of these four sets, only *abc* is displayed):

Prefix	Frequencies		
	a	*abc*	*abcd*
$\langle abcd\ abc\ a\ a\ \emptyset \rangle$	**4/5**	**2/5**	**1/5**
$\langle abcd\ abc\ a\ a \rangle$	1	1/2	1/4
$\langle abcd\ abc\ a \rangle$	1	2/3	1/3
$\langle abcd\ abc \rangle$	1	1	1/2
$\langle abcd \rangle$	1	1	1

Theorem 5.4. *Let A be an itemset, and let $\mathcal{B} \subseteq 2^A$ be a collection of itemsets over A. Then, there exists a stream $\mathbb{S}$ and a position $1 \leq p \leq |\mathbb{S}|$ such that for any subset J of A it holds that p is a border for J in $\mathbb{S}$ if and only if $J \in \mathcal{B}$.*

Proof. Let $\mathcal{L}$ and $\mathcal{R}$ be bags as in Lemma 5.1; i.e., the sets that occur more as a subset of transactions in $\mathcal{R}$ than as subset of transactions in $\mathcal{L}$ are exactly the sets in $\mathcal{B}$. Put the transactions of $\mathcal{L}$, respectively $\mathcal{R}$, in an arbitrary order to get the stream $\mathbb{S}_\mathcal{L}$, respectively $\mathbb{S}_\mathcal{R}$. According to Lemma 5.2, there exists a number n_l such that for all $n \geq n_l$, $\langle A \rangle \cdot \mathbb{S}_\mathcal{L} \cdot \mathbb{Z}_n$ is suffix-maximized. Similarly, according to Lemma 5.3, there exists a number n_r such that for all $n \geq n_r$, $\langle A \rangle \cdot \mathbb{S}_\mathcal{R} \cdot \mathbb{Z}_n$ is prefix-minimized. Now choose $n_1 \geq n_l$ and $n_2 \geq n_r$ such that $n_1 + |\mathbb{S}_\mathcal{L}| = n_2 + |\mathbb{S}_\mathcal{R}|$. The following stream and position $p = 1 + |\mathbb{S}_\mathcal{L}| + n_1$ satisfy the conditions of the theorem (position p is indicated by a vertical bar):

$$\langle A \rangle \cdot \mathbb{S}_\mathcal{L} \cdot \mathbb{Z}_{n_1} | \langle A \rangle \cdot \mathbb{S}_\mathcal{R} \cdot \mathbb{Z}_{n_2}.$$

Let us now analyze which itemsets have a border at position p. Because $\langle A \rangle \cdot \mathbb{S}_\mathcal{L} \cdot \mathbb{Z}_{n_1}$ is suffix-maximized, for all itemsets $J \subseteq A$, among all before-blocks ending at position $p-1$, the frequency of J is maximal in the whole part before p. Similarly, since $\langle A \rangle \cdot \mathbb{S}_\mathcal{R} \cdot \mathbb{Z}_{n_2}$ is prefix-minimized, among all after-blocks starting at position p, the frequency of J is minimized by the whole part starting at position p. Hence, J has a border at position p if and only if its frequency in $\langle A \rangle \cdot \mathbb{S}_\mathcal{L} \cdot \mathbb{Z}_{n_1}$ is strictly smaller than its frequency in $\langle A \rangle \cdot \mathbb{S}_\mathcal{R} \cdot \mathbb{Z}_{n_2}$. As n_1 and n_2 have been chosen to make both of equal length, this means that the absolute number of occurrences of J in $\langle A \rangle \cdot \mathbb{S}_\mathcal{L} \cdot \mathbb{Z}_{n_1}$, which is $1 + count(J, \mathcal{L})$ must be smaller than the absolute number of occurrences of J in $\langle A \rangle \cdot \mathbb{S}_\mathcal{R} \cdot \mathbb{Z}_{n_2}$, which is

$1+count(J,\mathcal{R})$. Because $\mathcal{L}$, $\mathcal{R}$ satisfy by definition Lemma 5.1, this implies that $J \in \mathcal{B}$. Since J was chosen arbitrarily, the itemsets having a border at position p are exactly those in $\mathcal{B}$, which proves the theorem. □

In the next example we illustrate the proof of Theorem 5.4 by bringing all examples together.

Example 15. The construction of the stream and position p in the proof of Theorem 5.4 for the collection of sets $\mathcal{B}=\{abc,a\}$ will be the one given in Example 11. As Example 12 shows, the bags $\mathcal{L}$ and $\mathcal{R}$ that are generated using the procedure described in the proof of Lemma 5.1 are

$$\mathcal{L}=\{ab,ac,bc\},$$

$$\mathcal{R}=\{abc,a,a\}.$$

We form the streams $\mathbb{S}_{\mathcal{L}}$ and $\mathbb{S}_{\mathcal{R}}$ by putting the elements in the bags in any order. Suppose that we chose for

$$\mathbb{S}_{\mathcal{L}}=\langle ab \;\; ac \;\; bc\rangle,$$

$$\mathbb{S}_{\mathcal{R}}=\langle abc \;\; a \;\; a\rangle.$$

Then we need to find numbers n_1 and n_2 such that $abcd \cdot \mathbb{S}_{\mathcal{L}} \cdot \mathbb{Z}_{n_1}$ is suffix-maximized, $abcd \cdot \mathbb{S}_{\mathcal{R}} \cdot \mathbb{Z}_{n_2}$ is prefix-minimized, and both have the same length. According to Examples 13 and 14, the smallest numbers for which this holds are $n_1=n_2=2$, which results in the stream and position:

$$\langle abcd \;\; ab \;\; ac \;\; bc \;\; \emptyset \;\; \emptyset \;\; |abcd \;\; abc \;\; a \;\; a \;\; \emptyset \;\; \emptyset\rangle$$

6. Worst case analysis

In this section we study how *large* the summary can become in worst case. For streams of a specific length L, we will identify a stream of this length that maximizes the number of borders and we show the asymptotic behavior of this tight bound. Farey sequences play an important role in this analysis.

6.1. Farey streams

Consider a stream $\mathbb{S}_L$ of length L and a target itemset A. In the following, $p_1,\dots,p_r$ will denote the borders of A in $\mathbb{S}_L$. Consider the following r blocks: $\mathbb{B}_i := \mathbb{S}[p_i,p_{i+1}]$ for $i=1,\dots,r-1$, and $\mathbb{B}_r := \mathbb{S}[p_r,]$. Let $i=1,\dots,r$, $b_i := |\mathbb{B}_i|$, and $a_i = count(A,\mathbb{B}_i)$. We visualize these settings as follows:

$$|\boxed{a_1/b_1}|\boxed{a_2/b_2}|\cdots|\boxed{a_r/b_r}.$$

From Theorem 3.2, we know that the frequencies of the target itemset in the blocks must be increasing:

$$\frac{a_1}{b_1}<\frac{a_2}{b_2}<\cdots<\frac{a_r}{b_r}.$$

Thus, every stream with r borders corresponds to such an increasing sequence of r fractions. We call this sequence of fractions the *block frequency sequence* of the stream. We can assume without loss of generality that the length of the stream is the sum of the denominators $b_1+\cdots+b_r$ (we can always omit the leading non-targets in the stream). The other direction is also true: for every increasing sequence of numbers

$$0<\frac{a'_1}{b'_1}<\frac{a'_2}{b'_2}<\cdots<\frac{a'_r}{b'_r}\le 1,$$

we can find a stream of length $b'_1+\cdots+b'_r$ with r borders, namely:

$$|\overbrace{a\dots a}^{a'_1\times}\overbrace{b\dots b}^{b'_1-a'_1\times}|\overbrace{a\dots a}^{a'_2\times}\overbrace{b\dots b}^{b'_2-a'_2\times}|\dots|\overbrace{a\dots a}^{a'_r\times}\overbrace{b\dots b}^{b'_r-a'_r\times}.$$

We will call this stream the *canonical stream associated with the sequence* $a'_1/b'_1<a'_2/b'_2<\cdots<a'_r/b'_r$. 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* [8] will be crucial.

$$\begin{array}{lcl}
F_1 & = & \frac{1}{1}\\
F_2 & = & \frac{1}{2},\frac{1}{1}\\
F_3 & = & \frac{1}{3},\frac{1}{2},\frac{2}{3},\frac{1}{1}\\
F_4 & = & \frac{1}{4},\frac{1}{3},\frac{1}{2},\frac{2}{3},\frac{3}{4},\frac{1}{1}\\
F_5 & = & \frac{1}{5},\frac{1}{4},\frac{1}{3},\frac{2}{5},\frac{1}{2},\frac{3}{5},\frac{2}{3},\frac{3}{4},\frac{4}{5},\frac{1}{1}
\end{array}$$

Fig. 7. Farey sequences of orders 1–5.

Definition 6.1. The Farey set of order k, denoted F_k is the following set of completely reduced fractions[3]:

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

The Farey sequence [8] of order k is the list where the elements of F_k are ordered in increasing order.

In Fig. 7, the Farey sequences of orders 1–5 are given.

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 Fig. 8. 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 property is based on the following straightforward observation. Let $dsum(\{a_1/b_1,\dots,\; a_r/b_r\})=\sum_{i=1}^{r} b_i$, i.e., $dsum(S)$ is the sum of the denominators of the (completely reduced) fractions in S.

[3] Note that in our definition, the fraction $0/k$ is not included in the Farey set F_k.

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

Fig. 8. Illustration of $\mathbb{F}_5$, the Farey stream of fifth order.

Lemma 6.2. *Let $S=\{a_1/b_1,\ldots,a_r/b_r\}$ be a set of r different fractions, with $0<a_i<b_i$, for all $i=1,\ldots,r$. Let k be such that $|S|>|F_k|$, then*

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

Proof. It is easy to see that $dsum(S)-dsum(F_k)=dsum(S-F_k)-dsum(F_k-S)$, and that $|S-F_k|>|F_k-S|$. Furthermore, any fraction in $S-F_k$ must have a denominator of at least $k+1$, and every fraction in F_k-S has a denominator of at most k. Therefore,

$$\begin{aligned}&dsum(S)-dsum(F_k)\\&\quad=dsum(S-F_k)-dsum(F_k-S)\\&\quad\geq(k+1)\cdot|S-F_k|-k\cdot|F_k-S|\\&\quad>|S-F_k|>0.\end{aligned}$$

Hence, $dsum(S)$ must be larger than $dsum(F_k)$. □

Theorem 6.3. *Let $\mathbb{S}$ be a stream with $|\mathbb{S}|=|\mathbb{F}_k|$. Then, the number of borders in $\mathbb{S}$ is at most the number of borders in $\mathbb{F}_k$.*

Proof. Consider the block frequency sequence $S:=\{a_1/b_1,\ldots,a_r/b_r\}$ of $\mathbb{S}$. The number of borders in $\mathbb{S}$ equals $|S|$, and the number of borders in $\mathbb{F}_k$ equals $|F_k|$. Suppose now, for the sake of contradiction, that the number of borders in $\mathbb{S}$ is larger than the number of borders in $\mathbb{F}$. Then, $|S|>|F_k|$, and thus, because of Lemma 6.2, $dsum(S)>dsum(F_k)$. This is in contradiction with the fact that $dsum(S)=|\mathbb{S}|=|\mathbb{F}_k|=dsum(F_k)$. Hence, the number of borders in $\mathbb{S}$ can maximally be the number of borders in $\mathbb{F}_k$. □

Corollary 6.4. *Let $L=dsum(F_k)$, and $r=|F_k|$, for a fixed k. A stream of length L has maximally r borders.*

6.2. 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 about *Euler's totient function* ϕ and the *Möbius function* μ.

The *Euler's totient function* ϕ maps positive integers j to the number of integers $i, 1\leq i\leq j$ that are co-prime to j; that is, $gcd(i,j)=1$.

Example 16. $\phi(5)=4$, as $1,2,3,4$ are all co-prime to 5. $\phi(8)=4$, as only $1,3,5,7$ are co-prime to 8.

It is not too hard to see that the number of elements in the Farey set F_k, i.e., $|F_k|$ equals $\sum_{i=1}^{k}\phi(i)$. Indeed, for $k=1$, this identity obviously holds. Furthermore, the number of new elements in $F_{k+1}-F_k$ are exactly those fractions $x/(k+1)$ such that $1\leq x\leq(k+1)$ and $gcd(x,k+1)=1$ ($x/(k+1)$ must be completely reduced). Hence, the number of new fractions equals the number of integers x, $1\leq x\leq k+1$ that are co-prime to $k+1$, which is $\phi(k+1)$. For some values of k, these numbers have been given in Table 1.

Table 1
The length of $\mathbb{F}_k$, a Farey stream of order k.

k	Length of $\mathbb{F}_k$	# borders of $\mathbb{F}_k$
1	$\lvert\mathbb{F}_1\rvert=\varphi(1)=1$	$\lvert F_1\rvert=\varphi(1)=1$
2	$\lvert\mathbb{F}_1\rvert+2\cdot\varphi(2)=1+1\times2=3$	$\lvert F_1\rvert+\varphi(2)=1+1=2$
3	$\lvert\mathbb{F}_2\rvert+3\cdot\varphi(3)=3+2\times3=9$	$\lvert F_2\rvert+\varphi(3)=2+2=4$
4	$\lvert\mathbb{F}_3\rvert+4\cdot\varphi(4)=9+2\times4=17$	$\lvert F_3\rvert+\varphi(4)=4+2=6$
5	$\lvert\mathbb{F}_4\rvert+5\cdot\varphi(5)=17+4\times5=37$	$\lvert F_4\rvert+\varphi(5)=6+4=10$
6	$\lvert\mathbb{F}_5\rvert+6\cdot\varphi(6)=37+2\times6=49$	$\lvert F_5\rvert+\varphi(6)=10+2=12$
7	$\lvert\mathbb{F}_6\rvert+7\cdot\varphi(7)=49+6\times7=91$	$\lvert F_6\rvert+\varphi(7)=12+6=18$
⋮	⋮	⋮

Obviously, these characterizations are still not that useful for getting insight in how the number of borders relates to the size of the streams. Therefore, we show the following asymptotic behaviors of the above sums:

$$\sum_{i=1}^{k}\phi(i)=\frac{3k^2}{\pi^2}+\mathcal{O}(k\log k),$$

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

For the first equality, it is well-known that

$$\frac{1}{k^2}\sum_{i=1}^{k}\phi(i)=\frac{3}{\pi^2}+\mathcal{O}\left(\frac{\log(k)}{k}\right).\tag{3}$$

Hence, asymptotically, $|F_k|$ becomes $3k^2/\pi^2$.

For the second sum we could not find a similar result in the literature. Therefore, we give our own proof of the asymptotic result for the second sum

$$\frac{1}{k^3}\sum_{i=1}^{k}i\cdot\phi(i)=\frac{2}{\pi^2}+\mathcal{O}\left(\frac{\log(k)}{k}\right).$$

Our proof of this result uses similar techniques as the known proofs for the asymptotic behavior of (3). This result shows that asymptotically, $dsum(F_k)$ becomes $2k^3/\pi^2$.

Lemma 6.5. *For all integers $1\leq i\leq k$, it holds that*

$$\left(2\left\lfloor\frac{k+1}{i}\right\rfloor^3+3\left\lfloor\frac{k+1}{i}\right\rfloor^2+\left\lfloor\frac{k+1}{i}\right\rfloor\right)-\left(2\left\lfloor\frac{k}{i}\right\rfloor^3+3\left\lfloor\frac{k}{i}\right\rfloor^2+\left\lfloor\frac{k}{i}\right\rfloor\right)$$

equals $6(k+1)^2/i^2$ if $i|k+1$, and is 0 otherwise.[4]

Proof. Indeed, if $i\nmid k+1$, then $\lfloor(k+1)/i\rfloor$ equals $\lfloor k/i\rfloor$. Therefore, in this case, the expression trivially evaluates to 0.

On the other hand, if $i|(k+1)$, then $\lfloor(k+1)/i\rfloor=(k+1)/i$, and $\lfloor k/i\rfloor$ equals $(k+1-i)/i$, as $k+1-i$ is the largest integer

[4] As usual, $i|k+1$ denotes that i is a divisor of $k+1$.

divisible by i that is smaller than k. Therefore, in this case, the expression equals

$$2\left(\frac{k+1}{i}\right)^3+3\left(\frac{k+1}{i}\right)^2+\left(\frac{k+1}{i}\right)-2\left(\frac{k+1-i}{i}\right)^3 -3\left(\frac{k+1-i}{i}\right)^2-\left(\frac{k+1-i}{i}\right)=\frac{6k^2+12k+6}{i^2}. \qquad \square$$

Let now $\mu(j)$ denote the *Möbius function*. $\mu(j)$ is defined as follows: $\mu(j)$ is 1 if the prime factorization of j is square-free and has an even number of factors, is -1 if the factorization is square-free and has an odd number of factors, and is 0 otherwise.

Example 17. $\mu(3)=-1$, because the prime factorization has 1 factor and is square-free. $\mu(8)=0$, as the prime factorization is 2^3, which is not square free. $\mu(6)=1$, because the factorization $6=2\cdot 3$ is square free and has an even number of terms.

$\mu(1)$ is defined to be 1, as the number of factors can be considered to be 0, which is even. The Möbius function has some nice properties when combined with Euler's totient. For example, the following identity will be very useful in the remainder of our proof:

$$\frac{\phi(k)}{k}=\sum_{i|k}\frac{\mu(i)}{i}. \tag{4}$$

We have the following interesting relationship:

Lemma 6.6. *Let k be a positive integer.*

$$\sum_{i=1}^{k} i\cdot\phi(i)=\frac{1}{6}\sum_{i=1}^{k} i\cdot\mu(i)\cdot\left(2\left\lfloor\frac{k}{i}\right\rfloor^3+3\left\lfloor\frac{k}{i}\right\rfloor^2+\left\lfloor\frac{k}{i}\right\rfloor\right).$$

Proof. We will prove this identity by induction.

For $k=1$, $\phi(1)=1=\frac{1}{6}\mu(1)(2+3+1)$.

For $k+1$: suppose that the equality holds for $1,\dots,k$. Then, we still need to prove the following equality in order to extend to $k+1$: $6(k+1)\phi(k+1)$ equals

$$\sum_{i=1}^{k+1} i\cdot\mu(i)\cdot\left(2\left\lfloor\frac{k+1}{i}\right\rfloor^3+3\left\lfloor\frac{k+1}{i}\right\rfloor^2+\left\lfloor\frac{k+1}{i}\right\rfloor\right)$$
$$-\sum_{i=1}^{k} i\cdot\mu(i)\cdot\left(2\left\lfloor\frac{k}{i}\right\rfloor^3+3\left\lfloor\frac{k}{i}\right\rfloor^2+\left\lfloor\frac{k}{i}\right\rfloor\right)$$
$$=\sum_{\substack{i|(k+1)\\ i<k+1}} i\cdot\mu(i)\cdot\frac{6(k+1)^2}{i^2}+6(k+1)\cdot\mu(k+1)$$

(By Lemma 6.5)

$$=6(k+1)^2\sum_{i|(k+1)}\frac{\mu(i)}{i}=6(k+1)^2\frac{\phi(k+1)}{(k+1)} \quad \text{(Identity 4)}$$
$$=6(k+1)\phi(k+1). \qquad \square$$

Theorem 6.7. *Let k be a positive integer.*

$$\frac{1}{k^3}\sum_{i=1}^{k} i\cdot\phi(i)=\frac{2}{\pi^2}+\mathcal{O}\left(\frac{\log(k)}{k}\right).$$

Proof. The proof is based on the identity given in Lemma 6.6, and the fact that we can lower and upper bound this identity, using $x-1<\lfloor x\rfloor\le x$. Therefore, we get the following lower bound on $(1/k^3)\sum_{i=1}^{k} i\cdot\phi(i)$:

$$\frac{1}{6k^3}\sum_{i=1}^{k} i\cdot\mu(i)\cdot\left(2\left(\frac{k}{i}-1\right)^3+3\left(\frac{k}{i}-1\right)^2+\left(\frac{k}{i}-1\right)\right)$$
$$=\frac{1}{6k^3}\sum_{i=1}^{k} i\cdot\mu(i)\cdot\left(2\left(\frac{k}{i}\right)^3-3\left(\frac{k}{i}\right)^2+\left(\frac{k}{i}\right)\right)$$
$$=\frac{2}{6}\sum_{i=1}^{k}\frac{\mu(i)}{i^2}+\mathcal{O}\left(\frac{\log(k)}{k}\right).$$

Because $\sum_{i=1}^{k}\mu(i)/i^2$ converges to $6/\pi^2$, we get the following asymptotic behavior for the lower bound:

$$\frac{2}{\pi^2}+\mathcal{O}\left(\frac{\log(k)}{k}\right).$$

Similarly, the same asymptotic behavior can be proven for the upper bound, thus establishing the theorem. □

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

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

Fig. 9 shows the number of borders for Farey streams of lengths up to 10^7 together with the upper bound given by this formula. As can be seen, the bound on the number of borders is almost exactly the actual worst case number.

7. Experiments

7.1. Implementation details

We have implemented a prototype of the max-frequency stream mining algorithms using Python. Max-Freq-Miner, Max-Freq-Miner^{mwl} and Max-Freq-Miner$_\sigma$ are straightforward to implement. The implementation of Max-Freq-Miner$_\sigma^{mwl}$ and Max-Freq-Miner-All$_\sigma^{mwl}$, however, is more involved. Three problems need to be dealt with. (i) We only need to create a summary for those itemsets that leave the minimal window and that were frequent in the minimal window. Thus, we need to find

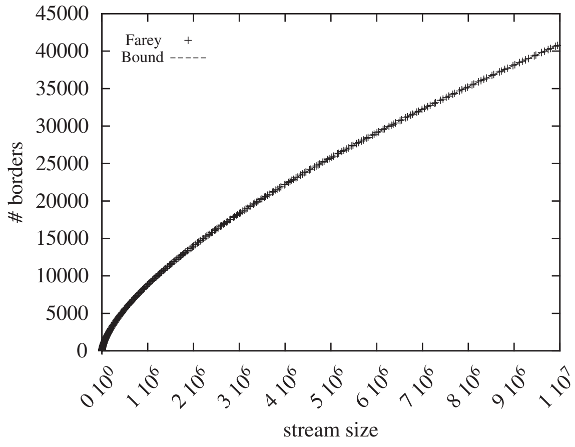


Fig. 9. Worst case number of borders.

the *frequent itemsets in the minimal window* in an efficient manner. (ii) In order to calculate the frequency of the borders in the summary on $\mathbb{S}^{-mwl}$ over the whole stream, we need to know the number of occurrences of an itemset in the minimal window. Thus, we need an efficient *counting procedure*. (iii) The starting point of the maximal window is either $L-mwl+1$ (i.e., the start of the minimal window), or a border in the summary on $\mathbb{S}^{-mwl}$, or it is a position between $L-2mwl+1$ and $L-mwl$. Thus, for all frequent itemsets occurring on these positions, we need to find the positions at which they are most frequent.

To solve the first two problems we implemented an incremental version of Eclat, see Section 7.1.1. The tid-lists used in Eclat are not deleted after mining the minimal window. They can thus be reused at the next timestamp and can also be used to efficiently count the number of occurrences of an itemset in the minimal window. The third problem is resolved by using a condensed per-item representation of the window $\mathbb{S}[L-2mwl+2,L-mwl+1]$, see Section 7.1.2.

Note that because the minimal window length is typically extremely small compared to the size of the stream, any itemset miner can be used to mine the minimal window.

7.1.1. *Incremental Eclat*

Our incremental implementation of Eclat is identical to Eclat (with the diffset optimization) [24,25], except that we keep the diffsets of the *items* in main-memory and update them incrementally, according to the itemset that is entering (resp. leaving) the minimal window. Finding the frequent itemsets is done by running the Eclat algorithm.

We can also use the incrementally updated diffsets of the items in the minimal window to *count the number of occurrences of an itemsets*. Suppose we want to count how many times itemset X occurs in the minimal window. First, we check whether all the items of itemset X are present (i.e., have a diffset). If all itemsets are present, we calculate the diffset of X by taking the union of the diffsets of the items of X. Subtracting the size of the diffset from the minimal window length results in the number of occurrences of X in the minimal window.

One could opt to keep all the diffsets (for items and itemsets) in memory and update all of them incrementally, instead of the method we employ. While this alternative "incrementalization" of Eclat will definitely prove valuable in the case where several of the frequent itemsets are relatively long, storing all these diffsets will require a large amount of memory.

7.1.2. *Position summaries*

Call the window of length *mwl* just behind the minimal window, i.e., $\mathbb{S}[L-2mwl+2,L-mwl+1]$, the second minimal window, denoted by $\mathbb{MW}_2$. We keep a *condensed per-item representation* of the second minimal window in memory, called position summaries. A position summary is similar to the summaries used in our main algorithm, except that no positions are pruned and the summary is always on a substream of length *mwl*.

How do we find the frequencies of the frequent itemsets in the second minimal window? For each item, occurring in $\mathbb{MW}_2$, we keep a list of pairs (p,c), stating that the item occurs c consecutive times starting at position p. Next we apply a two-stage iterative algorithm, similar to the candidate generation and pruning in Apriori, to find for all frequent itemsets, their maximal frequency and the position at which this frequency is attained. The iterative algorithm starts with copies of the position summaries of the items present in the second minimal window.

Stage 1: Given the position summary of an itemset and the count of that itemset in the minimal window, it is easy to calculate position producing the highest frequency.[5] The position summaries of infrequent itemsets are pruned.

Stage 2: In the second step, we combine the frequent items into candidate frequent two-itemsets. The position summary of a two-itemsets ab can be derived from the position summaries of a and b. We keep two pointers, one pointing to a position in the position summary of a, the other pointing to a position in the position summary of b. At each iteration, we check whether there is an overlap between the sequence of a's and the sequence of b's as defined by the position-count-pairs pointed to by the pointers. The pointer pointing to the oldest position (count is used to break a tie) is moved to the next position-count-pair. It is easy to generalize this procedure to n position summaries. A new iteration starts.

7.2. *Datasets*

7.2.1. *Synthetic datastreams*

Table 2 provides an overview of the synthetic datasets. Column *Probability distribution* defines which probability distribution was used to model the occurrence of items in a stream. For a uniform distribution the uniform chance is provided. For a sine distribution the period of the sine function is provided. A sine distribution equals a sine function plus with 0.5, i.e., the probability goes up and down between 0 and 1. A Gaussian peak distribution looks like a Gauss probability distribution with a mean and a standard deviation but it is not normalized, i.e., the probability rises from 0 to 1 and then drops back to 0.

The bursty 10, bursty 20 and sparse & bursty streams are composed of "bursts" of an itemset, i.e., an itemset occurs multiple consecutive times. In the case of bursty 10, the length of a burst is modeled by a normal distribution with mean 10 and standard deviation 5. For bursty 20 a normal distribution with mean 20 and standard deviation 5 was used. The sparse & bursty stream was created with a normal distribution with mean 40 and standard deviation 5. Moreover, during the

[5] Note that we, again, need to count the number of occurrences of an item(set) in the minimal window.

Table 2
Overview of the synthetic data streams used in the experiments. The column probability distribution contains information on the probability distribution used to generate the items.

Name	Length	#	Item	Probability distribution
Short	10 000	5	a	Uniform 0.5
			b	Sine period 100
			c	Gaussian peak (4000, 100)
			d	Gaussian peak (6000, 200)
Bursty 10	10 000	5	a	Uniform 0.57
			b	Uniform 0.57
			c	Uniform 0.57
Bursty 20	10 000	5	a	Uniform 0.57
			b	Uniform 0.57
			c	Uniform 0.57
Sparse & bursty	10 000	5	∅	Uniform 0.5
			a	Uniform 0.3
			b	Sine period 200
			c	Gaussian peak (3000, 1000)
			d	Gaussian peak (4000, 1000)
			e	Gaussian peak (5000, 1000)

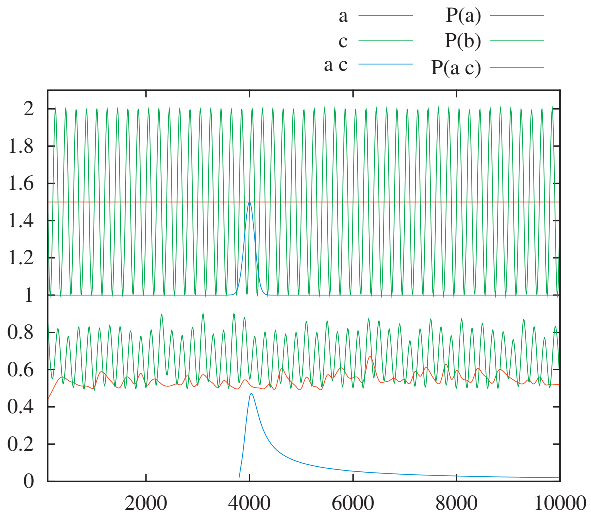


Fig. 10. A comparison between the actual frequency used while constructing one of the short streams and the frequency as mined by Max-Freq-Miner$_{\sigma}^{mwl}$ in the actual stream. Top: frequency used for constructing itemsets *a*, *b* and *ac*. Bottom: mined frequency of itemsets *a*, *b* and *ac*.

creation of the sparse & bursty streams, a burst of the empty itemset was inserted with a 50% chance.

Does the max-frequency measure discover the frequencies embedded in these synthetic streams? Fig. 10 shows the embedded frequency and the max-frequency of three itemsets in one of the short streams. We clearly see the embedded frequency (at the top of the plot) reappear in the max-frequency (at the bottom of the plot), proving the practical relevance of max-frequency.

7.2.2. Real-world datastreams: alarms

From a data mining company we received a dataset of alarms. An *alarm* occurs when a sensor value exceeds a prespecified threshold value. Alarms are grouped in *blocks*. For example: the alarms of the sensors on a single machine form a single block. Blocks are grouped into *compounds*. For example: the alarms of the sensors of all the machines on a single floor of a factory form a single compound.

Table 3
Characteristics of the blocks and compounds streams.

Stream	Length	Largest itemset	Median itemset length
Blocks-10	3 968 417	85	0
Compounds-10		28	0
Blocks-60	661 403	87	0
Compounds-60		31	0

From the raw timestamped data a stream of itemsets was constructed by making 10-s and 60-s slices of blocks (resp. compounds). There are 3873 blocks and 290 compounds in the data. All four streams are sparse, as the median length of an itemset is zero. The characteristics of the four streams are presented in Table 3. Because very large itemsets occur in the streams and most of them occur at the end of the stream, we have to set appropriate values for the minimal window length and the minimal frequency threshold. Indeed, without a *mwl* and σ each and every subset of the 87-item itemset would become frequent at some point, and thus require a summary. Clearly, we would run out of memory instantaneously. By setting the minimal window length and minimal frequency threshold, only for those itemsets that are frequent in the minimal window a summary is started.

7.3. Memory efficiency

A crucial point for the performance and applicability of the stream mining algorithms in this paper is the number of borders in the summaries. Large summaries may require too much memory to remain feasible. The performance of the algorithms with minimal window length (Max-Freq-Miner^{mwl}, Max-Freq-Miner$_{\sigma}^{mwl}$ and Max-Freq-Miner-All$_{\sigma}^{mwl}$) will also degrade: the max-frequency in these algorithms is computed by comparing the frequency of all the borders. Note that if the minimal window length and minimal frequency threshold have been set, some additional memory is used to store the minimal window, the second minimal window and for mining both. However, as noted before, the minimal window length is constant and very small compared to the size of the stream.

Therefore we computed, for several different streams and various values of *mwl* and σ, the mean, the median and the maximum number of borders (summed over all summaries) maintained during the run of the mining algorithms. The results are shown in Figs. 11 and 12 for a short and a sparse & bursty stream (results for the bursty 10 and 20 streams are omitted as they are very similar to the results of the short streams). As expected, the number of borders tends to zero for high values of σ (i.e., virtually no border is frequent). On the other hand,

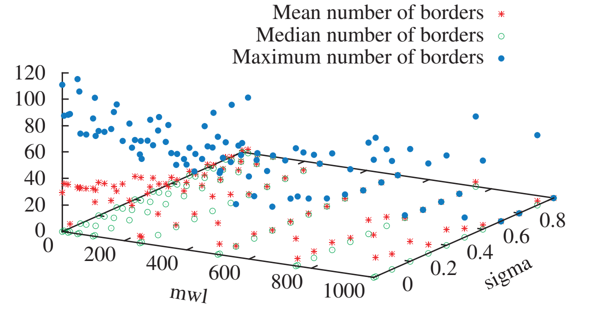


Fig. 11. Mean and median number of borders in all summaries while mining a *short* stream.

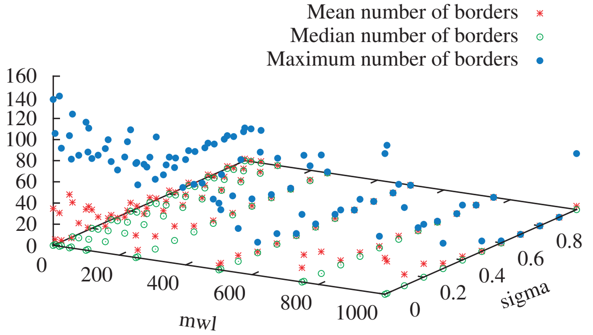


Fig. 12. Mean and median number of borders in all summaries while mining a *sparse & bursty* stream.

if the minimal frequency threshold is low (0 and 0.01), there are on average between 30 and 35 borders in all the summaries together while mining the short, bursty 10 and bursty 20 streams, and the maximum number of borders does not exceed 120. Evidently, more borders are needed to mine the sparse & bursty stream, as this stream features five items. Nonetheless, these results show a highly efficient memory usage: in the median case, about 10 borders are present.

How about the memory consumption in our real-world datasets? We show the results of the blocks-10 and compounds-60 streams. Figs. 13 and 14 show the mean and median number of borders used, over all summaries, in the respective streams. Despite the fact that the blocks-10 stream consists of 3873 distinct items, at most 4000 borders need to be maintained. Even more strikingly, on average only 2.5 borders are present, summed over all summaries. Similarly, the compounds-60 stream consist of 290 distinct items and require, on average, less than eight borders to track all frequent itemsets.

Fig. 15 shows the running time required to mine the blocks-10 stream. Notwithstanding that this stream contains itemsets of up to 85 items, with suitable minimal window length and minimal frequency threshold they can be effectively mined.

7.3.1. Conclusion

On average (and even in the maximum case), a very limited amount of memory is consumed by the summaries, both in the synthetic and in the real-world dataset. We also showed that our algorithm can effectively cope, both in terms of memory and time, with long streams containing many items and large itemsets.

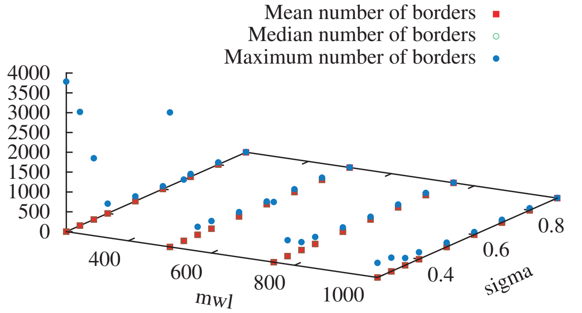


Fig. 13. Mean, median and maximum number of borders in all summaries while mining the blocks-10 stream.

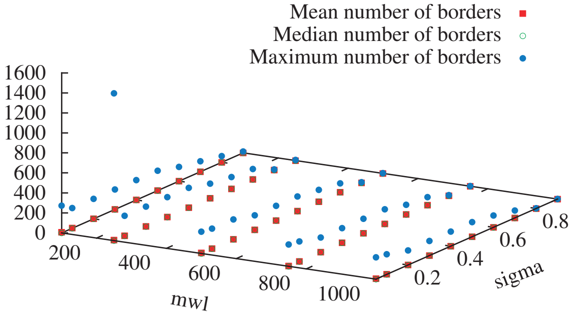


Fig. 14. Mean and median number of borders in all summaries while mining the compounds-60 stream.

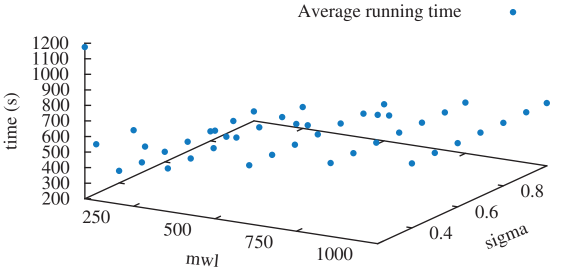


Fig. 15. The running time of Max-Freq-Miner$_{\sigma}^{mwl}$ on the blocks-10 stream. At the origin $mwl = 100$ and $\sigma = 0.25$, the running time starts to increase dramatically. Further lowering the minimal frequency threshold will make too many itemsets max-frequent.

7.4. Checkscheduler and lazy handling

The checkscheduler and lazy handling are straightforward optimizations Max-Freq-Miner-All$_{\sigma}^{mwl}$. In this section we investigate their effectiveness both on synthetic and real-world datasets.

7.4.1. Sparse & bursty streams

A stream is called *sparse* if many of its itemsets are empty. A stream is called *bursty* if an itemset mostly occurs on multiple consecutive time points. A sparse &

bursty stream thus has many stretches of empty itemsets, interrupted by bursts of an itemset. On such streams, one would except a lazy handling checkscheduler to be advantageous, because a burst forms a solid foundation for a slowly decreasing frequency.

The experiment consists of running the mining algorithm with and without the lazy handling checkscheduler on the following parameter values: *mwl*=1, 20, 100, 250, 500, 750, 1000 and σ=0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99. Each parameter combination is run five times on each of the sparse & bursty streams.

Fig. 16 shows the relative difference in the average running times of the algorithm *with* and *without* the lazy handling checkscheduler. If the minimal frequency threshold is not too high, say below 0.3, a speed-up of at least 10% is achieved. In the most extreme case, i.e., $mwl = 1$ and $\sigma = 0$, a speed-up of about 50% is achieved. However, with higher frequency thresholds, the speed-up becomes a slow-down. Also, longer minimal windows incur lesser speed-ups. This comes as no surprise, because scheduling a check for an itemset implies assessing the frequency of that itemset in the minimal window. The cost of computing the frequency in the minimal window grows linearly with the minimal window length.

7.4.2. *Bursty streams*

The sparseness is favorable for the lazy handling checkscheduler. If no check is scheduled, nothing happens. Therefore we test in this experiment whether the performance gain is preserved when sparseness is left out of the equation.

Each algorithm is run five times with the lazy handling checkscheduler and five times without the lazy handling checkscheduler on the bursty 10 and bursty 20 streams. The parameter values are *mwl*=1, 20, 100, 250, 500, 750, 1000 and σ=0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99.

Fig. 17 shows the relative difference in running time between the mining algorithm with and without a lazy handling checkscheduler. The speed-up effect on small minimal window length and minimal frequency threshold values is more pronounced on the bursty 20 streams. Nonetheless, the speed-up is far less pronounced than what we have seen on the sparse & bursty streams.

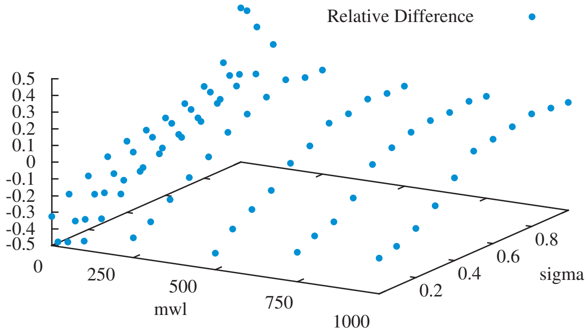


Fig. 16. The relative difference in average running time of the mining algorithms with and without the lazy handling checkscheduler on the sparse & bursty streams. Negative numbers imply that the algorithm with checkscheduler is faster, whereas positive numbers imply the algorithm without checkscheduler is faster.

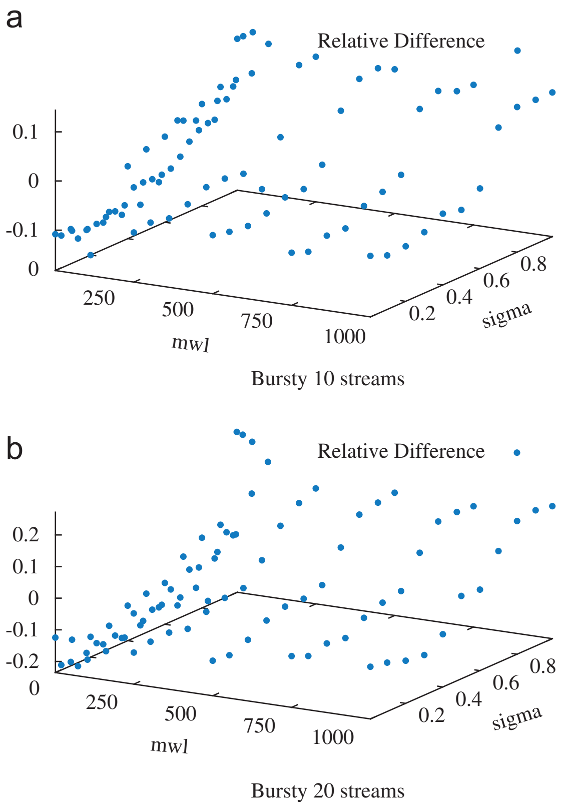


Fig. 17. The relative difference in running time of the mining algorithms with and without lazy handling checkscheduler on the bursty 10 and bursty 20 streams. Negative numbers imply that the algorithm with checkscheduler is faster, whereas positive numbers imply the algorithm without checkscheduler is faster.

7.4.3. *Short streams*

Thirdly, the "burstiness" is also eliminated from the streams. The possible parameter values are unaltered with respect to the previous experiments, but this time the short streams are mined.

Fig. 18 shows the relative difference in the average running time of the algorithms with and without lazy handling checkscheduler. In this case, the lazy handling checkscheduler is only a slow-down for all combinations of *mwl* and σ.

7.4.4. *Alarms*

We also investigated the effect of the checkscheduler on the time required to mine the four alarms streams. Fig. 19 shows the run times of Max-Freq-Miner-All$_{\sigma}^{mwl}$ for various σ- and *mwl*-values when mining the blocks-10 alarms stream. Again, we see that the difference in run time is in favor of the miner without checkscheduler. Although the stream is sparse, large itemsets become very frequent (up to 25% frequency) at the end of the stream. The checkscheduler requires counting the number of itemsets in the minimal window. This becomes expensive if many (larger) itemsets are frequent.

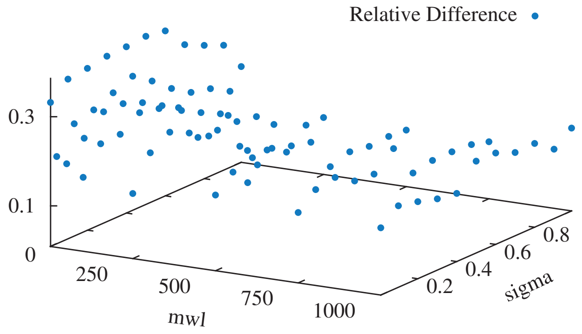


Fig. 18. Relative difference in running time of the mining algorithms with and without lazy handling checkscheduler on the short streams. Negative numbers imply that the algorithm with checkscheduler is faster, whereas positive numbers imply the algorithm without checkscheduler is faster.

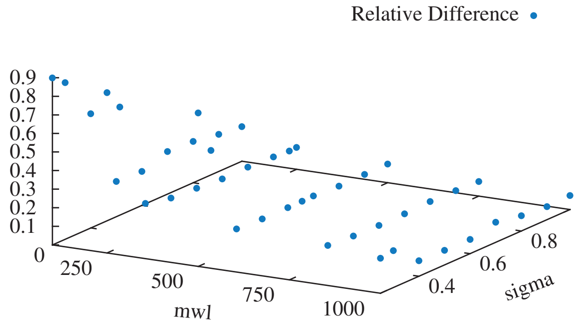


Fig. 19. The relative difference in running times of Max-Freq-Miner-All$_{\sigma}^{mwl}$ on the blocks-10 stream of alarms, with and without lazy handling checkscheduler. Positive number implies that the algorithm without checkscheduler is faster.

7.4.5. Conclusion

On the synthetic datastreams, four factors have an effect on the speed-up achieved by a lazy handling checkscheduler: (i) the minimal window length, (ii) the minimal frequency threshold, (iii) the sparseness of the stream and (iv) the length of the bursts. On the real-world datastreams, the lazy handling checkscheduler always underperformed. We have yet to figure out the specific reason for this behavior.

Surprisingly enough, lazy handling on its own and the checkscheduler on its own are not able to achieve substantial speed-ups. The combination is more efficient than the sum of the parts.

7.5. Border positions

In theory a border can occur at any position in the stream, but where are the borders positioned in practice? To find out how the borders are distributed over the stream in practice, we let the algorithm output the positions of all the borders in the active summaries, ($mwl=1$ and $\sigma=0$), while mining on one of the short streams. A plot was generated from this data. On the x-axis we have the length of the stream thus far. The y-axis shows the positions of the border *relative to the length of the stream*, i.e., the range of the y-axis is [0, 1]. The plots for items a, b and c are shown in Figs. 20–22. From the plots of a and b (i.e., items which occur throughout the stream) we can deduce that there are three kinds of borders. First of all we see a lot of borders *at the end* (the most recent itemsets) of the streams. Because the windows are short and the itemsets potentially plentiful in this part of the stream, a lot of borders is needed to capture all potential maximal windows. This illustrates that our algorithm *can effectively find sudden bursts* of an itemset. The second kind of border can be found *between the relative positions*0.8 *and* 0.2. These borders mark the starting of a period with a relatively high occurrence of the target itemset. Thirdly, we have the borders under relative position 0.2. These are the borders that are *at the start* (the oldest itemsets) of the stream. These borders

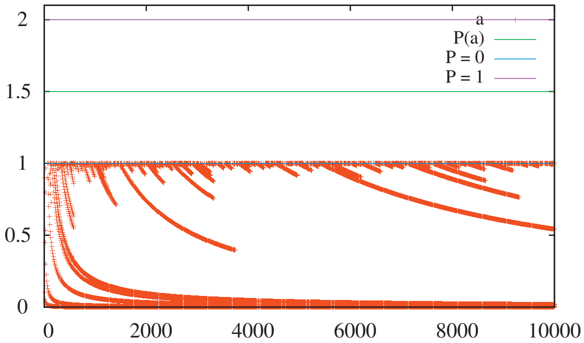


Fig. 20. Position of the borders in the summary of item a, relative to the length of the stream. Parameters of the algorithm: $mwl=1$ and $\sigma=0$.

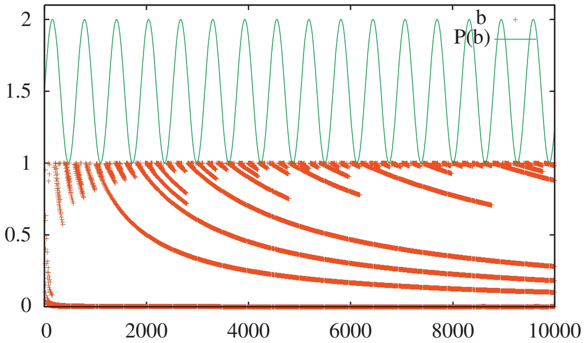


Fig. 21. Position of the borders in the summary of item b, relative to the length of the stream. Parameters of the algorithm: $mwl=1$ and $\sigma=0$.

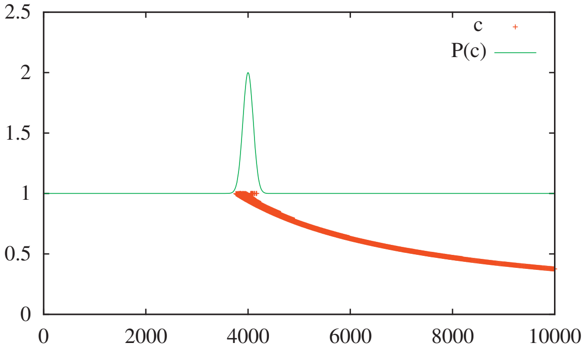


Fig. 22. Position of the borders in the summary of item c, relative to the length of the stream. Parameters of the algorithm: $mwl=1$ and $\sigma=0$.

capture the average frequency of the target itemset in the whole stream. If the minimal window length is set to a higher value we get largely the same plot, except that the relative border positions start somewhat lower. If the minimal frequency threshold is set, we will see the same three regions, where the third kind of border will now be at more recent positions.

In Fig. 22 we see that there is only a single "fang". A lot of borders are created at the moment that the frequency of item c is rising rapidly. Once the peak-frequency is attained few new borders are created and the oldest borders start absorbing the younger ones. Eventually only a single border remains.

8. Conclusion

The max-frequency measure is apt for mining frequent itemsets in a datastream. It truthfully captures the frequency of itemsets, as is shown by Fig. 10. Frequent itemsets can be mined efficiently, both in terms of time and space. The efficiency of the algorithm can be derived theoretically by linking the number of borders to the concept of Farey streams, from number theory. From this connection we can deduce that, even in the worst case, the number of borders increases slower than the length of the stream. Experiments on datasets, in which itemsets occur in varying patterns, show that the number of borders in a summary, in practice, remains relatively stable over time.

We showed that no edge can be gained from exploiting the subset relation between itemsets. Simply storing item-summaries results in space issues. Sharing borders among summaries linked by the subset relation on itemsets is impossible because at any given point a position might be a border for itemsets A_1 and A_3 but not for itemset A_2, where we have that $A_1 \subset A_2 \subset A_3$. As a result, further optimizations of the algorithm will need to be based on non-trivial insights.

From the experiments with our software prototype on both synthetic and real-world datasets, we draw three conclusions. Firstly, on average very few borders are needed to derive the max-frequency of all itemsets in a stream. Secondly, streams that are sparse & bursty over the whole course of the stream benefit from the lazy handling checkscheduler. Thirdly, even stream with large transactions can be mined efficiently if suitable values for the minimal window length and minimal frequency threshold have been set.

References

[1] A. Bifet, G. Holmes, B. Pfahringer, R. Gavaldà, Mining frequent closed graphs on evolving data streams, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 591–599.

[2] T. Calders, N. Dexters, B. Goethals, Mining frequent items in a stream using flexible windows, in: ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams (IWKDDS), Springer, 2006, p. 12.

[3] T. Calders, N. Dexters, B. Goethals, Mining frequent itemsets in a stream, in: ICDM, 2007.

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

[5] J. Cheng, Y. Ke, W. Ng, Maintaining frequent closed itemsets over a sliding window, Journal of Intelligent Information Systems 31 (2008) 191–215.

[6] J. Cheng, Y. Ke, Q. Ng, A survey on algorithms for mining frequent itemsets over data streams, Knowledge and Information Systems 16 (2008) 1–27.

[7] Y. Chi, H. Wang, P.S. Yu, R.R. Muntz, Catch the moment: maintaining closed frequent itemsets over a data stream sliding window, Knowledge and Information Systems 10 (2006) 265–294.

[8] J.H. Conway, R.K. Guy, Farey Fractions and Ford Circles, Springer-Verlag, 1996 (In The Book of Numbers (New York)), pp. 152–154, 156.

[9] G. Cormode, M. Hadjieleftheriou, Methods for finding frequent items in data streams, The VLDB Journal 19 (2010) 3–20.

[10] X.H. Dang, W.K. Ng, K.L. Ong, Online mining of frequent sets in data streams with error guarantee, Knowledge and Information Systems 16 (2008) 245–258.

[11] E.D. Demaine, A. López-Ortiz, J.I. Munro, Frequency estimation of internet packet streams with limited space, in: ESA, 2002, pp. 348–360.

[12] L. Golab, D. DeHaan, E.D. Demaine, A. López-Ortiz, J.I. Munro, Identifying frequent items in sliding windows over on-line packet streams, in: Internet Measurement Conference, 2003, pp. 173–178.

[13] R. Jin, G. Agrawal, An algorithm for in-core frequent itemset mining on streaming data, in: ICDM, 2005, pp. 210–217.

[14] R.M. Karp, S. Shenker, C.H. Papadimitriou, A simple algorithm for finding frequent elements in streams and bags, ACM Transactions on Database Systems 28 (2003) 51–55.

[15] Y. Kim, J. Ryu, U. Kim, Fia: frequent itemsets mining based on approximate counting in data streams, in: Neural Information Processing, Springer, 2009, pp. 312–322.

[16] D. Lee, W. Lee, Finding maximal frequent itemsets over online data streams adaptively, in: ICDM, 2005, pp. 266–273.

[17] H. Li, S. Lee, Mining frequent itemsets over data streams using efficient window sliding techniques, Expert Systems with Applications 36 (2009) 1466–1477.

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

[19] B. Mozafari, H. Thakkar, C. Zaniolo, Verifying and mining frequent patterns from large windows over data streams, in: Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, Cancún, México, April 7–12, 2008, pp. 179–188.

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

[21] E. Wang, A. Chen, A novel hash-based approach for mining frequent itemsets over data streams requiring less memory space, Data Mining and Knowledge Discovery 19 (2009) 132–172.

[22] E. Wang, A. Chen, Mining frequent itemsets over distributed data streams by continuously maintaining a global synopsis, Data Mining and Knowledge Discovery (2011) 1–48.

[23] J.X. Yu, Z. Chong, H. Lu, A. Zhou, False positive or false negative: mining frequent itemsets from high speed transactional data streams, in: VLDB, 2004, pp. 204–215.

[24] M. Zaki, Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering 12 (2000) 372–390.

[25] M. Zaki, K. Gouda, Fast Vertical Mining Using Diffsets, 2003.