Detecting Bursts of Activity from Streams

Bruno Veloso
INESCTEC, Porto, Portugal
bruno.m.veloso@inesctec.pt

Carlos Martins
WEDO, Braga, Portugal
Carlos.MMartins@wedotechnologies.com

Raphael Espanha
WEDO, Braga, Portugal
Raphael.Espanha@wedotechnologies.com

Raul Azevedo
WEDO, Braga, Portugal
Raul.Azevedo@wedotechnologies.com

João Gama
INESCTEC, Porto, Portugal
jgama@fep.up.pt

Abstract

The high asymmetry of international termination rates, where calls are charged with higher values, are fertile ground for the appearance of frauds in Telecom Companies. In this paper, we present a solution for a real problem called Interconnect Bypass Fraud. This problem is one of the most expressive in the telecommunication domain and can be detected by the occurrence of burst of calls from specific numbers. Based on this assumption, we propose the adoption of a new fast forgetting technique that works together with the Lossy Counting algorithm. Our goal is to detect as soon as possible items with abnormal behaviours, e.g. bursts of calls, repetitions and mirror behaviours. The results shows that our technique not only complements the techniques used by the telecom company but also improves the performance of the Lossy Counting algorithm in terms of runtime, memory used and sensibility to detect the abnormal behaviours.

1 Introduction

Telecom fraud continues to account for more fraud complaints each year. New technology has led to an onslaught of new telecommunication fraud tactics. Around 29 Billion USD are lost annually in the Telecom sector as a result of Telecom Fraud, according with Communications Fraud Control Association's (CFCA) 2017 Global Fraud Loss Survey [A+17], a main reference in the sector for such a sensible matter. This means 1.25 % of direct impact in revenues and does not takes into account reputation costs as fraud is evolving in such a way that is impacting more on subscribers' services.

CFCA's survey confirmed also the perception that the investment made in better practices and systems is having a return, with a reduction of 23.2 % when compared with 2015 survey edition, despite of a significantly increase in fraud attempts.

In terms of fraud types, International Revenue Share Fraud continues to be the main revenue source for Fraudsters, causing a loss of 6.10 Billion USD according with CFCA, closely followed by Interconnection Bypass Fraud, causing an estimated loss of 4.27 Billion USD according to the same source. Regarding Interconnection

Bypass Fraud, recent changes in regulation, namely in the European Union space, are creating a favourable context for this type of fraud to grow in more developed countries, as interconnect rates grown to compensate losses in revenues resultant from the Roaming Like at Home directive.

And if Digital Transformation in Communication Service Providers raise new concerns, it also brings opportunities to a more effective fraud fight. As systems evolve towards more real-time and intelligent capabilities, it enables also real-time fraud controls to be implemented in a faster and cheaper way, as well as creating a space for new generation of adaptive Machine Learning based controls.

In this paper we will focus on a solution to efficiently detect Interconnect Bypass Fraud, in order to be used in real-time. This fraud explores the forward of international calls using low cost IP connections. Despite of the fact that Fraudsters are shaping usage patterns to avoid traditional fraud detection controls, Interconnect Bypass Fraud is often characterised by the occurrence of burst of calls.

Based on this assumption in this paper, we propose the adoption of fast forgetting technique together with the Lossy Counting algorithm. Our goal is to detect as soon as possible items with abnormal behaviours. This methodology, when compared with the pre-existing approaches referred in Section 3, enables to detect recent abnormal patterns like (bursts of calls, repetition and mirror behaviours).

The main contributions of this paper are summarised as follows:

- We perform experiments on a real-world data set, demonstrating that the Fast Forgetting technique significantly increases the detection of abnormal behaviours.
- We propose an extension of the Lossy Counting algorithm which includes the adoption of a new forgetting mechanism to fast detect abnormal behaviours.
- We apply the Lossy Counting algorithm with Fast Forgetting for the first time on the fraud detection domain.

In terms of organisation, this document contains six sections. Section 2 is dedicated to the problem definition. Section 3 presents the state of the art on frequent itemsets using Lossy Counting. Section 4 describes our approach, including the Lossy Counting algorithm with the fast forgetting. Section 5 describes the experiments and discusses the results obtained. Finally, Section 6 draws the conclusions and suggests future developments.

2 Problem Definition

The high asymmetry of international termination rates vis-à-vis domestic ones, where international calls are charged by the operator where the call terminates at a significantly higher value, are fertile ground for the appearance of fraud in Telecommunications. There are several types of fraud that exploit this type of differential being the Interconnect Bypass Fraud one of the most expressive.

In this type of fraud, one of several intermediaries responsible for delivering the calls forwards the traffic over a low cost IP connection, reintroducing the call in the destination network already as a local call, using VOIP Gateways. In this way, it charges the entity that sent the traffic the amount corresponding to the delivery of international traffic, but once it has illegally delivered as national traffic, it will not have to pay the international termination fee, appropriating this amount.

In addition to the financial aspect, this type of fraud causes image costs because both the originating subscriber of the call and the one who receives it, end up having a poor call quality.

This type of fraud is traditionally detected by analysing the call patterns of these Gateways that, once identified, have their SIM cards blocked. These gateways have evolved over time, resembling some of them, true SIM Farms, capable of manipulating identifiers, simulating standard call patterns similar to the ones of normal users and even being mounted on vehicles for making it difficult to detect them using location information.

Another important aspect for the deterrence of this type of fraud has to do with the speed of detection and blocking. If the time required for detection and blocking is not long enough for the payback to compensate for the cost and effort of system setup, the fraudster has no great incentive to bypass this operator.

A new approach is therefore necessary which, on the one hand, allows for sufficient adaptability to identify increasingly hidden fraud patterns and on the other hand allowing large-volume monitoring in real time so that, as soon as a suspicious behavior is detected an alert action is generated, or even immediate blocking, according to the degree of confidence of the particular detection.

It should be noted that this type of fraud, which was usually more prominent in developing countries where the cost of termination has always been high, has recently grown significantly in Europe as a result of the increase

in termination charges for calls originating outside Europe, in order to compensate for the decrease of revenues resulting from the application of the roaming like at home directive in the European area.

3 Related Work

The literature on data streams is abundant on approximate counting algorithms [Mut05]. Popular one-pass algorithms to approximate the most frequent items on an event stream are the SpaceSaving algorithm [MAEA05], the Frequent algorithm [CH10], the Sticky Sampling Algorithm [MM02], and the Lossy Counting algorithm [MM02]. The first two, provide a rank of the most frequent items, while the last two algorithms provide approximate counts of the frequent items. In all cases, the algorithms are designed for processing high-speed data streams, using fixed and restricted memory.

Sticky Sampling and Lossy Counting algorithms were presented in the [MM02] paper. They are based on the same idea of removing low-frequent items. While Lossy Counting is a deterministic algorithm, Sticky Sampling is a probabilistic one. Lossy counting is more accurate but Sticky Sampling requires constant space. Lossy Counting space requirements increase logarithmically with the length of the stream. Sticky sampling remembers every unique element that gets sampled, whereas Lossy Counting chops of low frequency elements quickly leaving only the high frequency ones. Sticky sampling can support infinite streams while keeping the same error guarantees as Lossy Counting.

Cormode et al. (2008) discuss hierarchical heavy hitters (ϕ-HHH) in streaming data. Given a hierarchy and a support ϕ, find all nodes in the hierarchy that have a total number of descendants in the data stream no smaller than $\phi \times N$ after discounting the descendant nodes that are also ϕ-HHH [CKMS08]. This is of particular interest for monitoring structured and networked data, like XML data, and explores the internal structure of data. Other Recent works for fast computing frequent items include [LJ11, NIY19, SN18].

4 Frequent Itemsets using `LossyCounting`

Manku and Motwani in [MM02] present the `LossyCounting` algorithm, a one-pass algorithm for computing frequency counts exceeding a user-specified threshold over data streams. Although the output is approximate, the error is guaranteed not to exceed a user-specified parameter. `LossyCounting` requires two user-specified parameters: a support threshold $s \in [0, 1]$, and an error parameter $\epsilon \in [0, 1]$ such that $\epsilon \ll s$. At any point of time, the `LossyCounting` algorithm can produce a list of item(set)s along with their estimated frequencies.

4.1 The `LossyCounting` Algorithm

Let N denote the current length of the stream. The answers produced will have the following guarantees:

- All item(set)s whose true frequency exceeds $s \times N$ are output. There are *no false negatives*;
- No item(set) whose true frequency is less than $(s - \epsilon) \times N$ is outputted;
- Estimated frequencies are less than the true frequencies by at most $\epsilon \times N$.

The incoming stream is conceptually divided into buckets of width $w = \lceil \frac{N}{\epsilon} \rceil$ transactions each. Buckets are labelled with bucket `ids`, starting from 1. Denote the current bucket `id` by $b_{current}$. For an element e, denote its true frequency in the stream seen so far by f_e. The frequent elements are stored in a data structure T. T contains a set of entries of the form (e, f, Δ), where e is the element, f its estimated frequency, and Δ is the maximum possible error in f.

The pseudo-code of the `Lossy Count` algorithm, is presented in Algorithm 1 [Gam10]. It works as follows. Initially, T, is empty. Whenever a new element e arrives, if an entry for e already exists, the algorithm increments its counter f. Otherwise, a new entry is created of the form $(e, 1, \lceil \frac{N}{\epsilon} \rceil)$. At bucket boundaries, the set T is pruned. The rule for deletion is: an entry (e, f, Δ) is deleted if $f + \Delta \leq \lceil \frac{N}{\epsilon} \rceil$. When a user requests a list of item with threshold s, `LossyCounting` outputs all the entries in T where $f \geq (s - \epsilon) \times N$.

4.2 Frequent Itemsets using `Lossy Counting`

Depending on the application, the `Lossy Counting` algorithm might treat a tuple as a single item or as a set of items. In the latter case, the input stream is not processed transaction by transaction. Instead, the available main memory is filled in with as many transactions as possible. After that, they process such a batch of transactions together. Let β denote the number of buckets in memory.

```
input: S: A Sequence of Examples; ε: Error margin;
begin
    n ← 0; Δ ← 0; T ← 0;
    foreach example e ∈ S do
        n ← n + 1
        if e is monitored then
            Increment Count_e
        else
            T ← T ∪ {e, 1 + Δ}
        end
        if ⌈n/ε⌉ ≠ Δ then
            Δ ← n/ε
            foreach all j ∈ T do
                if Count_j < Δ then
                    T ← T\{j}
                end
            end
        end
    end
end
```

Algorithm 1: The `Lossy Counting` Algorithm.

As for items, `Lossy Counting` maintains a data structure T, as a set of entries of the form (set, f, Δ), where set is a subset of items, f is an integer representing its approximate frequency, and Δ is the maximum possible error in f. D is updated as follows:

- **Update itemset:** For each entry (set, f, Δ), update by counting the occurrences of set in the current batch. Delete any entry such that $f + \Delta \leq b_{current}$;
- **New itemset:** If a set set has frequency $f \geq \beta$ in the current batch and does not occur in T, create a new entry $(set, f, b_{current} - \beta)$.

Every set whose true frequency is $f \geq \epsilon \times N$ has an entry in T. Moreover, for any entry $(set, f, \Delta) \in D$, the true frequency f_{set} satisfies the inequality $f \leq f_{set} \leq f + \Delta$. When a user requests a list of items with threshold s, output those entries in T, where $f \geq (s - \epsilon) \times N$.

4.3 Fast Forgetting

The `Lossy Counting` algorithm computes the frequency of items or sets of items in a stream of buckets. However, this algorithm do not capture small variations on each bucket because the continuous increment of the most frequent items of the stream. We propose the adoption of forgetting factors to reduce the relevance of old elements, and capture small frequency variations on the stream.

T contains a set of entries of the form (e, f, Δ), where e is the element, f its estimated frequency, and Δ is the maximum possible error in f taking into account the accumulated forgetting values. δ is the accumulated forgetting factors applied on the buckets and can be described by $w + \delta \times \alpha$, where w is the width of the bucket and α is the forgetting factor.

The pseudo-code of `Lossy Counting` with fast forgetting is presented in Algorithm 2. It works as follows. Initially, T, is empty. Whenever a new element e arrives, if an entry for e already exists, the algorithm increments its counter f. Otherwise, a new entry is created of the form $(e, 1, \lceil \frac{\delta}{\epsilon} \rceil)$. At bucket boundaries, the algorithm applies the forgetting factor α to each element on the data structure T and the set T is pruned taking into account the accumulated forgetting factors. The rule for deletion is: an entry (e, f, Δ) is deleted if $f + \Delta \leq \lceil \frac{\delta}{\epsilon} \rceil$. When a user requests a list of item with threshold s, `Lossy Counting` with fast forgetting outputs all the entries in T where $f \geq (s - \epsilon) \times \delta$.

input: S: A Sequence of Examples; ϵ: Error margin; α: fast forgetting parameter
begin
  $n \leftarrow 0; \Delta \leftarrow 0; T \leftarrow 0;$
  foreach *example* $e \in S$ **do**
    $n \leftarrow n + 1$
    if e *is monitored* **then**
      Increment $Count_e$
    else
      $T \leftarrow T \cup \{e, 1 + \Delta\}$
    end
    if $\lceil \frac{n}{\epsilon} \rceil \neq \Delta$ **then**
      $\Delta \leftarrow \frac{n}{\epsilon}$
      foreach *all* $j \in T$ **do**
        $Count_j \leftarrow \alpha * Count_j$
        if $Count_j < \Delta$ **then**
          $T \leftarrow T \backslash \{j\}$
        end
      end
    end
  end
end

Algorithm 2: The `Lossy Counting` with Fast Forgetting Algorithm.

4.4 Discussion

The results of Lossy Counting are order-dependent, giving heavier weight to the counts processed last. In our case, this makes perfect sense because we want to detect as soon as possible, items with a burst in activity. These burst are potential abnormal activities that must be analysed for fraud detection. The focus in recent items is very much improved with the new forgetting mechanism we introduce.

5 Experimental Evaluation

The following subsections present the data description, the experiments, the results obtained and a final discussion. The experiments were performed with an Intel Core CPU i5-5200u 2.20 GHz Central Processing Unit (CPU), 4 GB DDR3 Random Access Memory (RAM) and 1 TB of hard drive.

5.1 Data Description

Our proposal was evaluated with an anonymised phone calls data set, provided by a telecommunication company. The data set contains information about: (i) origin numbers (A-Numbers); (ii) destination number (B-Numbers); (iii) timestamp; and (iv) result, which represents if the call is blacklisted (code – 001) or not (code – 000). The data was collected during three months between 24/07/2018 to 21/10/2018 which includes 83 366 367 examples.

5.2 The impact of Forgetting

We made a set of experiments to verify the impact of applying forgetting on the `Lossy Counting` algorithm. The Lossy Counting algorithm proposed by Manku and Motwani (2002) it was our baseline algorithm. Figure 1 presents for each day the Top 5 A-Numbers that have a frequency higher than the pruning threshold. The algorithm capture a set of 12 A-Numbers or heavy hitters that can potential indicate some kind of fraud behaviour. However, using the Lossy Counting algorithm its difficult to discriminate repetition patterns or burst generated by the A-Numbers.

When we applied our fast forgetting mechanism the algorithm showed more sensibility to capture other A-Numbers. Figure 2 shows a set of 35 heavy hitters and it is possible to verify variability on the Top 5 A-Numbers. With our technique we can rapidly observe on Figure 3: (i) burst of calls generated by the A-Numbers (ID 14); (ii) repetition pattern observed on A-Number (ID 1) which cannot be captured using the Lossy Count as we can observe on the Figure 1; and (iii) mirror patterns observed on A-Numbers (ID 8 and ID 22) and (ID 19 and ID 24).

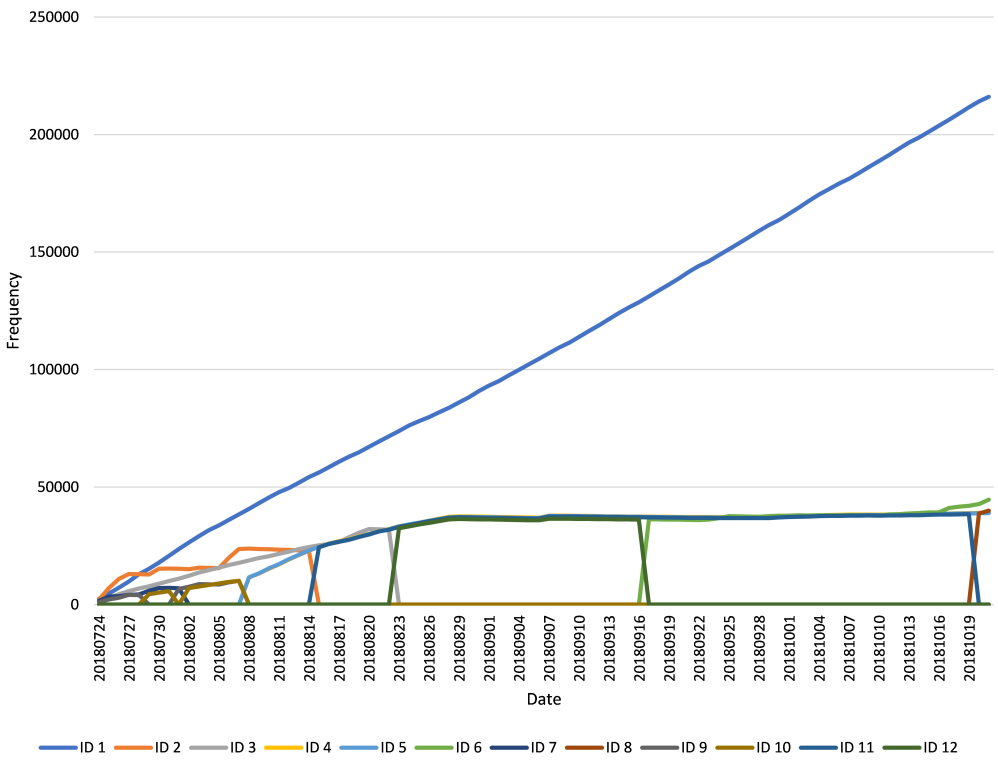

Figure 1: The top-5 most active A-numbers over time.

Table 1 presents the forgetting parameter sensitivity (α). We can observe that adjusting the forgetting parameter we can detect more Top 5 A-Numbers. When $\alpha = 1.0$ the algorithm works like the Lossy Counting proposed by Manku and Motwani (2002), i.e., we don't apply forgetting on the data. When $\alpha = 0.0$ we forget everything on the previous bucket of data.

Table 1: Forgetting Parameter Sensitivity; α is the forgetting parameter and UAN is Unique A-Numbers

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
UAN	211	210	203	192	180	175	158	123	93	66	12

Table 2 presents the performance of our proposal when compared with Lossy Counting algorithm. The Lossy Counting with Fast Forgetting, maintains the computer complexity of the Lossy Counting algorithm proposed by Manku and Motwani (2002) which is according to Martino et al. (2013) $O(\frac{1}{\epsilon}log(\epsilon N))$ [MNS13], reduces the runtime and the memory required to store the data structure T, and increases the speedup.

Table 2: Performance comparison of the Lossy Counting (LC) vs Lossy Counting with Fast Forgetting (LCFF)

Algorithm	Runtime (s)	Memory (MB)	SpeedUp (Examples/s)
LC	88	75.8	947 345
$LCFF$	72	28.8	1 157 866

6 Conclusions

The Interconnect Bypass Fraud is the most expressive fraud tactic in the telecommunication domain. The fraudsters explores the forwarding of international calls using low cost IP connections to increase their profits. This type of fraud is characterised by the occurrence of burst of calls.

We explore the use of a new fast forgetting mechanism for the Lossy Counting algorithm. This technique was designed to capture as soon as possible abnormal behaviours on phone calls.

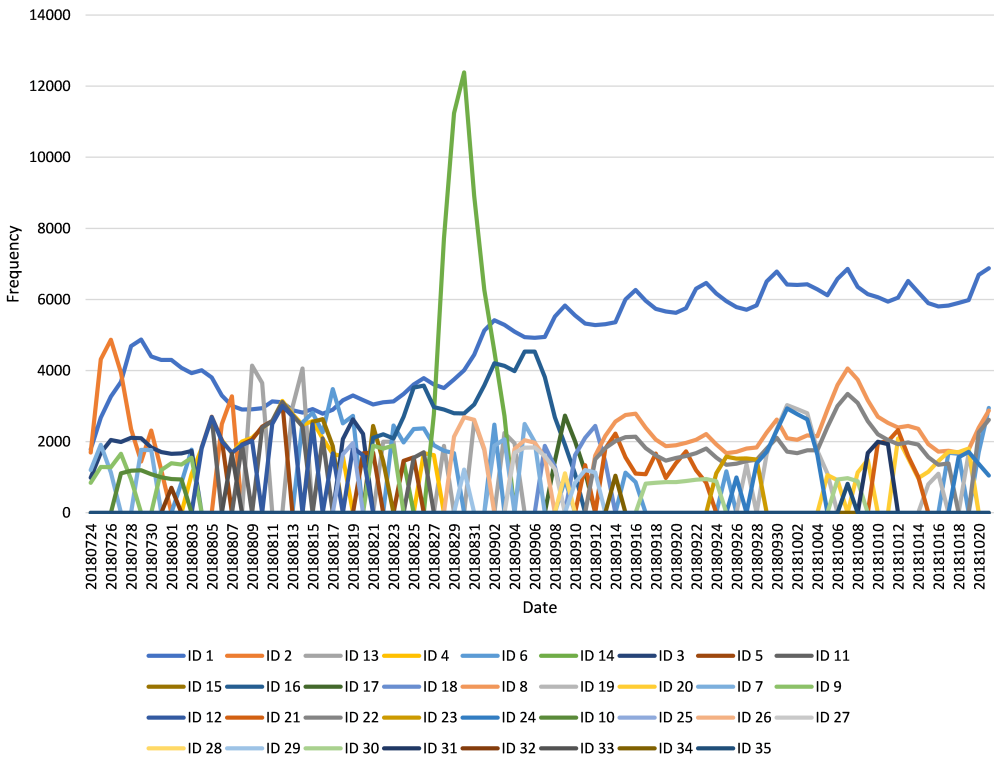

Figure 2: Lossy Counting with Fast Forgetting $\alpha = 0.99$. The figure reports the top-5 most active A-Numbers over time.

The contributions of this paper are at: (i) application level, to reduce the impact of Interconnect Bypass Fraud on the telecommunication companies; (ii) and at the methodology level, with the extension of the Lossy Counting algorithm with a fast forgetting mechanism to rapidly detect abnormal behaviours.

The experiments shows an inability of the Lossy Counting algorithm to detect recent items with abnormal behaviours (burst of calls, repetition and mirror behaviours). The results show that our proposal improved the detection of these recent items. Furthermore, our forgetting mechanism reduces the execution and memory used to compute the data stream, increasing the speedup of the algorithm.

For future research we explore the use of ranges to detect A-Numbers with similar behaviours.

6.0.1 Acknowledgements

Authors acknowledge the project ML-ABA - Machine Learn based Adaptive Business Assurance, Individual Demonstration Projects, NUP: FCOMP-01-0202-FEDER-038204, a project co-funded by the Incentive System for Research and Technological Development, from the Thematic Operational Program Competitiveness of the national framework program - Portugal2020. We also acknowledge the support of the project FailStopper (DSAIPA / DS /0086/2018).

References

[A+17] Communications Fraud Control Association et al. 2017 global fraud loss survey. *Press Release, June*, 2017.

[CH10] Graham Cormode and Marios Hadjieleftheriou. Methods for finding frequent items in data streams. *The VLDB Journal*, 19(1):3–20, February 2010.

[CKMS08] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Finding hierarchical heavy hitters in streaming data. *ACM Trans. Knowl. Discov. Data*, 1(4):2:1–2:48, February 2008.

[Gam10] João Gama. *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press, 2010.

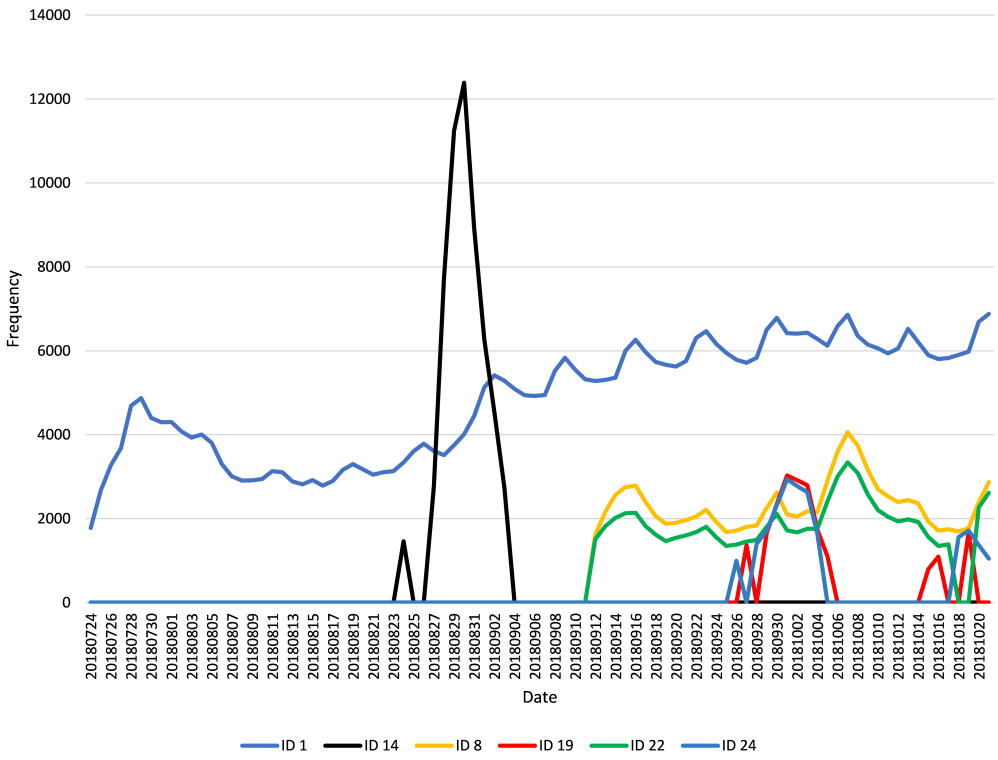

Figure 3: Lossy Counting with Fast Forgetting $\alpha = 0.99$. A-Numbers with abnormal behaviours (bursts, repetition and mirror)

[LJ11] Chao-Wei Li and Kuen-Fang Jea. An adaptive approximation method to discover frequent itemsets over sliding-window-based data streams. *Expert Syst. Appl.*, 38:13386–13404, 2011.

[MAEA05] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th International Conference on Database Theory*, ICDT'05, pages 398–412, Berlin, Heidelberg, 2005. Springer-Verlag.

[MM02] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 346–357. VLDB Endowment, 2002.

[MNS13] Giovanni Da San Martino, Nicolò Navarin, and Alessandro Sperduti. A lossy counting based approach for learning on streams of graphs on a budget. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1294–1301. AAAI Press, 2013.

[Mut05] S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, August 2005.

[NIY19] Takumi Nishina, Koji Iwanuma, and Yoshitaka Yamamoto. A skipping fp-tree for incrementally intersecting closed itemsets in on-line stream mining. *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–4, 2019.

[SN18] B. Subbulakshmi and A. Periya Nayaki. Compact pattern stream tree algorithm. 2018.