Order statistics and estimating cardinalities of massive data sets

Frédéric Giroire

2005 International Conference on Analysis of Algorithms DMTCS proc. **AD**, 2005, 157–166

Order statistics and estimating cardinalities of massive data sets

Frédéric Giroire[1]

[1] *Algorithms Project, INRIA Rocquencourt, F-78153 Le Chesnay , France.* frederic.giroire@inria.fr

We introduce a new class of algorithms to estimate the cardinality of very large multisets using constant memory and doing only one pass on the data. It is based on order statistics rather that on bit patterns in binary representations of numbers. We analyse three families of estimators. They attain a standard error of $\frac{1}{\sqrt{M}}$ using M units of storage, which places them in the same class as the best known algorithms so far. They have a very simple internal loop, which gives them an advantage in term of processing speed. The algorithms are validated on internet traffic traces.

Keywords: cardinality, estimates, very large multiset, traffic analysis

1 Introduction

Problem. The problem considered here is to estimate the number of distinct elements n, that is the *cardinality*, of very large multisets of size N while using constant memory and doing only one pass on the data. No assumption is made on the structure of repetitions. What is the first idea that comes into mind to count the number of distinct words in a text for example? One may keep in memory the words already seen and check, for each new word read, if it has already been seen. It is the principle of all the algorithms for exact counting which use more or less smart dictionaries to store and look after words. They use a memory in $O(n)$ and a time in $O(N \log n)$. The motivation comes from the analysis of very large databases or of internet traffic. In these fields the huge volume of data considered makes it impossible to use algorithms with linear memory. Internet backbone networks links, as OC-192 of SONET networks, may have a capacity of 10 Gbps. Stocking the data is already difficult as a 1 TB hard disk is filled in less than 14 minutes. For a survey on the difficulties to monitor high speed links see [10]. The only algorithms that could possibly be used must have constant memory and do only one pass on the data.

The three families of algorithms. The idea is to transform the problem into a probabilistic one. We don't look any more for the exact number of different words, but for an estimate with a given precision. The starting remark is that the minimum of n uniform random variables taking their values in $[0, 1]$ is an estimate of $\frac{1}{n+1}$. So we are able to retrieve an approximation of n from this value. A hashing function from the data of the problem into $[0, 1]$ distributes the hashed values uniformly in the interval. A crucial point to observe is that the minimum is not sensitive to the structures of repetitions. The minimum of hashed values of the multiset is the one of its underlying set. An algorithm built according to this principle uses a constant memory, only one floating number is necessary to keep the minimum, and do only one very simple pass on the file or sequence, each word being read, hashed and compared to the minimum. When the multiset has been read, we have the minimum of the hashed values. We want an estimate of n and not of $\frac{1}{n+1}$. The most natural way would be to inverse this minimum, but it has an infinite expectation. Our solution is to combined two principles to obtain an estimate indirectly from this minimum: instead of using the first minimum we use the second, third or k th one, and instead of using the inverse function alone, we combine it with sublinear functions as logarithm or square root. It gives us *three families of estimates* of n: inverse of the k th minimum, its logarithm and its square root. To have estimates as precise as possible we use *stochastic averaging* as introduced by P. Flajolet in [6]: the arithmetic mean of m independent random variables of same law has same expectation but standard error divided by $\sqrt{m}$. So we do an average over several similar experiments.

Related works. There has been substantial work on approximate query processing in the database community (see [9], [8], [2]). In [12] Whang, Zanden and Taylor have introduced *Linear Counting*. The principle is to distribute hashed values into buckets and use the number of hit buckets to give an estimate of the number of values. A drawback of this method is that memory is linear. To extend it to very

large data sets, Estan, Varghese and Fisk proposed in [4] a multiscale version of this principle in their *Multiresolution Bitmap* algorithm. This one keeps a collection of windows on the previous bitmap. Its estimate has a standard error of $4.4/\sqrt{m}$ while using m words of memory. An other way to estimate cardinalities is sampling. The idea is to keep only a fraction of the values already read. This fraction may be dynamically chosen in an elegant way as in Wegner's *Adaptive Sampling*, that has been described and analyzed by Flajolet in [5]. The accuracy for this method is $1.20/\sqrt{m}$. Gibbons proposes in [9] an other method, *Distinct Sampling*, to provide approximate answers to a variety of queries using a fraction of the total data in a database. The *Probabilistic Counting* algorithm of Flajolet and Martin (see [6]) uses bit patterns in binary representations of numbers. It has excellent statistical properties with an error close to $0.78/\sqrt{m}$. The *LogLog Counting* algorithm of M. Durand and P. Flajolet (see [3]) starts from the same idea but uses a different observable. The standard error is $1.30/\sqrt{m}$ for the first version and $1.05/\sqrt{m}$ for the Super-LogLog version, but the m 'words' of memory have here a size in $\log\log n$ and not in $\log n$. Finally in [1] the authors present three algorithms to count distinct elements. The first one uses the k th minimum and corresponds basically to the inverse family estimator. The authors prove that this algorithm $(\epsilon, \delta)-$approximates n using $O(1/\epsilon^2 \log m \log(1/\delta))$ bits of memory and $O(\log(1/\epsilon) \log m \log(1/\delta))$ processing time per elements. This paper generalizes this idea by introducing new and more efficient families of estimates and gives a precise analysis for them.

Results. In Section 2 we present the three families of estimators. We analyse one of them in Section 3 and give the results for all them in Section 4. We found *asymptotically unbiased* estimates of n for the three families and give their *standard error* in Theorem 4.1. We compare the families according to their trade-off between *accuracy* and *memory* in Theorem 4.2. We show that better estimates are obtained applying sublinear functions, that *precision at constant memory* gets better when k increases, that an optimal trade-off exists for the three families, a standard error of $\frac{1}{\sqrt{M}}$ using a memory of M floating numbers, and we propose a *best estimate*.

Simulations. The algorithms are validated on large data. They are tested on internet traffic traces of different sizes. Results are shown in Figure 2 in Section 5. We simulate a use of the estimator to detect denial of service attacks, see Figure 3. We also show that the families of algorithms built over the minimum are some of the fastest known algorithms. Not only they are doing only one pass on the data, but this pass is very quick. It is a crucial winning card in the context of in-line analysis of internet traffic where we have only of few tens of machine operations at disposal to process a packet.

2 Three families of estimates

Definitions and notations. Let $\mathcal{D}$ be our *data domain*. It may be the set of natural integers, the set of words of the French language or the pairs of IP addresses for example. A multiset $\mathcal{M}$ of elements of $\mathcal{D}$ of size N is given and the problem is to estimate its *cardinality*, that is, its number of *distinct* elements. What we have available is a hash function h, that transforms x, an element of $\mathcal{D}$, into $h(x)$ an element of $[0, 1]$ such as $h(x)$ is taken uniformly at random. The construction of this hash function is based on modular arithmetic and is discussed by Knuth in [11]. An *ideal multiset* of cardinality n is a sequence of n distinct values in $[0, 1]$ chosen uniformly at random, replicated in an arbitrary way and shuffled by applying an arbitrary permutation. The *input* of the families of algorithms is the ideal multiset $\mathcal{I}$ given by h applied to $\mathcal{M}$. The output is an *estimate* of its cardinality denoted by n.

Construction of estimators based on the minimum M. To estimate the number of distinct elements of the ideal multiset $\mathcal{I}$, we consider its minimum, M. An important remark is that the minimum of a sequence of numbers is found with a single pass on the elements and that it is not sensitive to repetitions. It is enough to read a number and compare this value to the minimum over the previous values. To see an element 10 times for example has no effect on the minimum. The density of the minimum of n random uniform variables over $[0, 1]$ is $\mathbb{P}(M \in [x, x + dx]) = n(1 - x)^{n-1}dx$. So its expectation for $(n \geq 1)$ is

$$\mathbb{E}[M] = \int_0^1 x \cdot n(1 - x)^{n-1}dx = \frac{1}{n + 1}. \tag{1}$$

M is roughly an estimator of $1/n$. Our hope is now to be allowed to take $1/M$ as an estimate of n. But

$$\mathbb{E}\left[\frac{1}{M}\right] = \int_0^1 \frac{1}{x} \cdot n(1 - x)^{n-1}dx = +\infty \tag{2}$$

Unluckily the integral is divergent in 0. To succeed to obtain an estimate of n we use *indirectly* the minimum M according to two different principles that can be combined.

1. Instead of using the inverse function alone, we compose it with a sublinear function f. This one is chosen in such a way that $f(1/M)$ estimates n. Its underlinearity flattens the values close to the origin so that the expectation is not infinite. Examples of these functions are (natural) *logarithm* and *square root*.

2. Instead of using the first minimum, we take the second, third or more generally the k th minima. These minima are not as close to zero as the first one allowing the computation.

This way we obtain three families of estimates namely the *Inverse Family*, the *Square Root Family* and the *Logarithm Family*. We talk about families as we have one estimator per value of k. Below their principles are explained.

Inverse family algorithm (F: multiset of hashed values; m)
for $x \in F$ do
    if $\frac{i-1}{m} \leq x \leq \frac{i}{m}$ do
        actualize the k minima of the bucket i with x
return $\xi := (k-1)\sum_{i=1}^{m} \frac{1}{M_i^{(k)}}$ as cardinality estimate.

Square Root Family Algorithm (F: multiset of hashed values; m)
for $x \in F$ do
    if $\frac{i-1}{m} \leq x \leq \frac{i}{m}$ do
        actualize the k minima of the bucket i with x
return $\xi := \frac{1}{\left(\frac{1}{(k-1)} + \frac{m-1}{(k-1)!^2}\Gamma(k-\frac{1}{2})^2\right)} \left(\sum_{i=1}^{m} \frac{1}{\sqrt{M_i^{(k)}}}\right)^2$ as cardinality estimate.

Logarithm Family Algorithm (F: multiset of hashed values; m)
for $x \in F$ do
    if $\frac{i-1}{m} \leq x \leq \frac{i}{m}$ do
        actualize the k minima of the bucket i with x
return $\xi := m \cdot \left(\frac{\Gamma(k-\frac{1}{m})}{\Gamma(k)}\right)^{-m} \cdot e^{-\frac{1}{m}\sum_{i=1}^{m} \ln M_i^{(k)}}$ as cardinality estimate.

Simulating m experiments. The precision of the algorithms is given by the *standard error* of their estimate ξ, denoted by $\mathbb{SE}[\xi]$ and defined as the standard deviation divided by n. To have more precise estimates we use *stochastic averaging* introduced by P. Flajolet in [6]. It is based on the fact that the *arithmetic mean* of m independent random variables of same law with expectation μ and standard deviation σ has same expectation μ and its standard deviation is $\sigma/\sqrt{m}$. Hence taking the arithmetic mean, denoted by $\mathcal{M}$, over several similar experiments increases the precision of the algorithms.

Doing m experiments involves m different hashing functions. But to hash all the elements m times is particularly time consuming and to build m independent hashing functions is not an easy task. To avoid these difficulties we simulate these m experiments. The principle is to distribute the hashed values among m different *buckets*. That is done by dividing $[0,1]$ in m intervals of size $1/m$. A hashed value x falls in the i th bucket if $\frac{i-1}{m} \leq x < \frac{i}{m}$. Our algorithms keep the k th minimum of the i th bucket, denoted by $M_i^{(k)}$ in the analysis, for i from 1 to m. Then they build m estimates with them and compute their arithmetic mean $\mathcal{M} := \sum_{i=1}^{m} f(1/M_i^{(k)})$. Finally we have to inverse it, in a way to obtain an asymptotically unbiased estimate. An example of analysis may be seen in Section 3.

3 The logarithm of the second minimum

We present here the complete analysis of the estimator of the logarithm Family built on the second minimum. It was chosen because it shows the typical difficulties that may be encountered, in particular a bias has to be corrected to find an asymptotically unbiased estimate. This section deals with the steps of the construction of this estimator and the proof of Proposition 3.1. This may also be seen as part of the proof of Theorem 4.1.

Computations of the expectation and of the variance. We saw in Section 2 that the inverse of the first minimum had an infinite expectation. The inverse of the second one has an infinite variance. But taking the logarithm, a sublinear function, of the second minimum allows us to have now converging integrals. The computations of the expectation and of the variance are a kind of preliminary to the construction of

the estimator. It guarantees that an estimator built on this function has finite expectation and variance and it gives us the inverse function over which is built the estimate ξ.

The computation is based on the use of special functions, more precisely on the identification between the formula of the derivative of the function $\Psi(z) = \frac{d}{dz}\Gamma(z)$ in two different ways: from its integral form and from this expression with Γ functions. The density of the second minimum is $x(1-x)^{n-2}dx$, so we have:

$$\begin{aligned}
\mathbb{E}[\ln \tfrac{1}{M^{(2)}}] &= n(n-1)\int_0^1 \ln \tfrac{1}{x} x(1-x)^{n-2}dx \\
&= n(n-1)\int_0^1 \ln x(1-x)^{n-1}dx - n(n-1)\int_0^1 \ln x(1-x)^{n-2}dx \\
&= n(n-1)\tfrac{d\mathcal{B}}{d\alpha}(1,n) - n(n-1)\tfrac{d\mathcal{B}}{d\alpha}(1,n-1) \\
&= n(n-1)\,(-\tfrac{\gamma}{n} - \tfrac{\psi(n+1)}{n}) - n(n-1)(-\tfrac{\gamma}{n-1} - \tfrac{\psi(n)}{n-1}) \\
&= H_n - 1
\end{aligned}$$

The computations for the variance are similar except that the second derivative of $\mathcal{B}$ integral is involved in this case. We obtain $\mathbb{V}[\ln \frac{1}{M^{(2)}}] = \frac{\pi^2}{6} - 1 - \psi'(n+1)$.

Construction of an unbiased estimate and evaluation of its standard error. We want to build precise estimates using the arithmetic mean $\mathcal{M}$ of m experiments. $\mathcal{M}$ has the same expectation as $\ln \frac{1}{M^{(2)}}$, that is $g(n) = H_n - 1$. Proposition 3.1 tells us that we can inverse it using $g^{-1}(x) = e^{x+1}$.

Proposition 3.1 *1. The estimate returned by the algorithm of the Logarithm Family built on the second minimum defined as*

$$\xi := m \cdot \Gamma\left(2 - \frac{1}{m}\right)^{-m} \cdot e^{\frac{1}{m}(f(M_1^{(2)})+\ldots+f(M_m^{(2)}))} \tag{3}$$

is asymptotically unbiased in the sense that

$$\mathbb{E}[\xi] \underset{n\to\infty}{\sim} n. \tag{4}$$

2. Its standard error, defined as $\frac{1}{n}\sqrt{\mathbb{V}(\xi)}$, satisfies

$$\mathbb{SE}[\xi] \sim \sqrt{\Gamma\left(2 - \frac{1}{m}\right)^{-2m} \Gamma\left(2 - \frac{2}{m}\right)^{m} - 1}, m \geq 2. \tag{5}$$

Proof. The proof is done in two steps. In its first part we consider a simplification of the problem where a same number of hashed values falls into each bucket: we called it the *equipartition hypothesis*. The random values $M_i^{(k)}$ are independent and have same expectation. In the second part we prove, using Lemma 3.1, that we obtain the same asymptotic results without this hypothesis.

First step of the proof.

$$\begin{aligned}
\mathbb{E}[e^{\mathcal{M}}] &= \mathbb{E}[e^{\frac{1}{m}(\ln \frac{1}{M_1^{(2)}}+\cdots+\ln \frac{1}{M_m^{(2)}})}] = \mathbb{E}[(M^{(2)})^{-\frac{1}{m}}]^m \\
&= \int_0^1 x^{-\frac{1}{m}} \tfrac{n}{m}(\tfrac{n}{m} - 1)x\,(1-x)^{\frac{n}{m}-2}dx \\
&= \tfrac{n(n-m)}{m^2}\mathcal{B}(2 - \tfrac{1}{m}, \tfrac{n}{m} - 1) \\
&= \tfrac{n}{n-1}\Gamma(2 - \tfrac{1}{m})\tfrac{\Gamma(\frac{n}{m})}{\Gamma(\frac{n}{m} - \frac{1}{m})}
\end{aligned}$$

For n large we have $\mathbb{E}[e^{\mathcal{M}}] \underset{n\to\infty}{\sim} \Gamma(2 - \frac{1}{m})^m \cdot \frac{n}{m}$. The expression $\frac{1}{m}\Gamma(2 - \frac{1}{m})^m$ appeared as bias. So we have $\mathbb{E}[\xi] \underset{n\to\infty}{\sim} n$. ξ is an asymptotically unbiased estimate of n.

Similar computations give us

$$\mathbb{E}[e^{2\mathcal{M}}] \underset{n\to\infty}{\sim} n^2 \cdot \Gamma(2 - \frac{2}{m})^m. \tag{6}$$

Thus we have the second point of the proposition.

Second step of the proof. We remove now the *equipartition hypothesis*. The numbers N_i of hashed values falling into the buckets are no longer the same but are following a multinomial law. That is $\mathbb{P}_{\{N_1=n_1,\cdots,N_m=n_m\}} = \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m}$. This fact introduces two difficulties: there is now a dependency between the $M_i^{(2)}$ and they have no more the same expectation.

If we now consider the computation of the estimate ξ, the main factor becomes:

$$\mathbb{E}[e^{\mathcal{M}}] = \mathbb{E}\left[e^{\frac{1}{m}(\ln \frac{1}{M_1^{(2)}} + \cdots + \ln \frac{1}{M_m^{(2)}})}\right] = \sum_{reps} \mathbb{P}_{rep} \times \mathop{\mathbb{E}}_{given\ rep}[M_1^{-1/m} \cdots M_m^{-1/m}] \tag{7}$$

The expectation is now a sum over all possible repartitions of the hashed values in the buckets, that is the different values for the N_i. When the repartition is fixed, the M_i become independent and

$$\mathbb{E}[e^{\mathcal{M}}] = \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n} \binom{n}{n_1, \cdots, n_m} \mathop{\mathbb{E}}_{N_1=n_1}[M_1^{-1/m}] \cdots \mathop{\mathbb{E}}_{N_m=n_m}[M_m^{-1/m}] \tag{8}$$

We use now a 'determinization' lemma to compute an equivalent of this kind of formulas. Let first define slow variation functions:

Definition 3.1 (Function with Slow Variation) *A function f of n is with slow variation if there is a function ϵ such that $\epsilon(n) \to 0$ as $n \to \infty$ and if, for all $j \leq \sqrt{n} \ln n$,*

$$f_{n+j} = f_n(1 + O(\epsilon(n))). \tag{9}$$

Lemma 3.1 ('Determinization' of random allocations) *Let f be a function of n*

1. *with growth at most polynomial and*
2. *with slow variation (in the sense of Definition 3.1).*

Then, for m fixed, if

$$S_n := \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n} \binom{n}{n_1, \cdots, n_m} f_{n_1} \cdots f_{n_m}, \tag{10}$$

we have

$$S_n = (f_{n/m})^m (1 + O(\epsilon(n))). \tag{11}$$

Proof omitted.

Let us come back to the computation of $\mathbb{E}[e^{\mathcal{M}}]$. We saw in Section 3 that $\mathbb{E}[M_1^{-1/m}]_{N_1=n_1} = \Gamma(2 - 1/m)\frac{n_1}{n_1-1}\frac{\Gamma(n_1/m)}{\Gamma(n_1/m-1/m)}$. Its growth is in $O(n^{1+1/n})$ and it has slow variation. Lemma 3.1 applies. We have then:

$$\mathbb{E}[e^{\mathcal{M}}] = \left(\mathop{\mathbb{E}}_{N_1=n/m}[M_1^{-1/m}]\right)^m (1 + O(\epsilon(n))). \tag{12}$$

This formula gives the same equivalent when n goes to infinity as with the *equipartition hypothesis*. We apply the same method for the computation of the standard error. So, in the general case, we may use the same unbiased estimate and it has asymptotically the same standard error. This ends the proof of Proposition 3.1. □

4 Analysis of the three families of estimates.

Here we give the results of the analysis of the three families of estimates in Theorem 4.1, compare them in Theorem 4.2 using *precision at constant memory* introduced in Definition 4.1 and proposed a *best estimate*.

4.1 Summary.

We studied the three families of estimates: the inverse of the k th minimum, the logarithm and the square root. The steps of the analysis are the same as in Section 3 for the logarithm of the second minimum and the proof is omitted here. The results are presented in Theorem 4.1. Most of the presented results correspond to the case where n is asymptotic: it gives more synthetic formulas and anyway we consider large sets of data for our problem. Some of the results are given in an even more synthetic form, for m large (but small in comparison to n).

Theorem 4.1 *Consider the three following families of algorithms*

1, the inverse family,
2, the square root family,
3, the logarithm family,

applied to an ideal multiset of unknown cardinality n.

1. *The estimates returned by the three families of algorithms ξ_1, ξ_2, ξ_3, defined respectively as*

$$\xi_1 := (k-1)\sum_{i=1}^{m}\frac{1}{M_i^{(k)}}, \tag{13}$$

$$\xi_2 := \frac{1}{\left(\frac{1}{(k-1)!}\Gamma(k-1)+\frac{m-1}{(k-1)!^2}\Gamma(k-\frac{1}{2})^2\right)}\left(\sum_{i=1}^{m}\frac{1}{\sqrt{M_i^{(k)}}}\right)^2 \text{ and} \tag{14}$$

$$\xi_3 := m\cdot\left(\frac{\Gamma(k-\frac{1}{m})}{\Gamma(k)}\right)^{-m}\cdot e^{-\frac{1}{m}\sum_{i=1}^{m}\ln M_i^{(k)}}, \tag{15}$$

   are asymptotically unbiased in the sense that, for $i=1,2,3$

$$\mathbb{E}[\xi_i] \underset{n\to\infty}{\sim} n. \tag{16}$$

2. *Their standard error, defined as $\frac{1}{n}\sqrt{\mathbb{V}(\xi_i)}$, satisfy*

$$\mathbb{SE}[\xi_1] \underset{n\to\infty}{\sim} \frac{1}{\sqrt{k-2}}\cdot\frac{1}{\sqrt{m}}, \tag{17}$$

$$\mathbb{SE}[\xi_2] \underset{n\to\infty}{\sim} \frac{2}{\sqrt{m}}\sqrt{\frac{1}{k-1}\left(\frac{\Gamma(k)}{\Gamma(k-\frac{1}{2})}\right)^2-1}, m \text{ large}, \tag{18}$$

$$\mathbb{SE}[\xi_3] \underset{n\to\infty}{\sim} \sqrt{\psi'(k)}\cdot\frac{1}{\sqrt{m}}, m \text{ large}. \tag{19}$$

4.2 Comparison.

We now determine the degree of performance of the algorithms, to compare them and to find the best. As all the estimators need only to do one pass over the data, we don't compare them on their speed but on the *memory* they use. The memory unit is the number of floating numbers used by the algorithm, corresponding to the m minima stored during the execution. The rest of the memory, like integers for loops or temporary variables, is not considered here as significant. As we are dealing with probabilistic algorithms, we also consider the *precision* of the estimates, measured by the *standard error*. As an estimate can have a good precision and be using a lot of memory, we introduce a third metric that is a trade-off between both: the *precision at constant memory*. The principle is to allow the algorithms to use a fixed amount of memory, M floating numbers, and to see what precision is reached.

Definition 4.1 (precision at constant memory) *The* precision at constant memory M *of an estimate ξ, $\mathcal{P}_{CM}(\xi)$ is the standard error of the estimator with largest parameter m that is compatible with a memory M. This quantity is expressed as a function of M. For our three families of estimates we have:*

$$\mathcal{P}_{CM}(\xi) = \frac{\sqrt{km}}{\sqrt{M}}\mathbb{SE}[\xi]$$

Theorem 4.2 *1. The* precision at constant memory M *of the three families of estimates are*

$$\mathcal{P}_{CM}(\xi_1) := \sqrt{\frac{k}{k-2}\frac{1}{M}} \tag{20}$$

$$\mathcal{P}_{CM}(\xi_2) := \frac{2}{\sqrt{M}}\sqrt{\frac{k}{k-1}\left(\frac{\Gamma(k)}{\Gamma(k-\frac{1}{2})}\right)^2-1} \tag{21}$$

$$\mathcal{P}_{CM}(\xi_3) := \sqrt{k\Psi'(k)\frac{1}{M}}, \text{with } \Psi(z)=\frac{d}{dz}\Gamma(z) \tag{22}$$

```
WHILE  hashed value
        IF  h < M^(2)
                IF  h < M
                        M^(2) = M
                        M = h
                ELSE M^(2) = h
```

Fig. 1: Internal loop for $k = 2$.

2. *For any k,*
$$\mathcal{P}_{CM}(\xi_3) \leq \mathcal{P}_{CM}(\xi_2) \leq \mathcal{P}_{CM}(\xi_1). \tag{23}$$
3. Precision at constant memory *is a decreasing function of k for the three families.*
4. *When k goes to infinity, we have*
$$\mathcal{P}_{CM}(\xi_i)_{i=1,2,3} \underset{k\to\infty}{\to} \frac{1}{\sqrt{M}} \tag{24}$$

Theorem 4.2 presents the *precision at constant memory* of the three families of estimators. The proof is omitted here. Four main results can be extracted from the theorem. **First result.** As expressed in point 2, better estimates are obtained when we apply sublinear functions, as square root or logarithm. For example, in the case of the third minimum ($k = 3$), $\mathcal{P}_{CM}(\xi_1) = 1.73/\sqrt{M}$, $\mathcal{P}_{CM}(\xi_2) = 1.26/\sqrt{M}$ and $\mathcal{P}_{CM}(\xi_3) = 1.09/\sqrt{M}$. **Second result.** As expressed in point 3, precision at constant memory is gained when k increases for all three families. The fact that precision increases with k is expected. For example, for the logarithm family for the first, second and third minimum the $\mathcal{P}_{CM}$ are respectivly 1.28, 1.13 and 1.09. **Third result.** As expressed in point 4, there exists an optimal trade-off between precision and memory for the three families: a standard error of $1/\sqrt{M}$ with a memory of M floating numbers. Let us remark that it also means that precision gained using sublinear functions, if very striking for small k, decreases when k increases. **Fourth result: Best estimate.** The optimal efficiency is reached quickly. As a matter of fact, the constant for the estimate of the logarithm family using only the third minimum is 1.09. We consider the estimate as the *best estimate* and we use it in the analysis of internet traffic in Section 5.3. Let remark that the fact the best efficiency is attained means that the introduction and study of other families of estimates of the same kind built for example with a 'more' sublinear function as $\ln^2$ would not procure decisive *practical* gains in terms of precision at constant memory.

5 Simulations

Our algorithms are motivated by the processing of very large multisets of data, in particular in the field of in-line analysis of internet traffic. Most backbone networks operated today are Synchronous Optical NETworks (SONET). In this kind of networks using optical fibers the different links are classified according to their capacity from OC-1 (51.84 Mbps) to OC-192 (10 Gbps) (widely used) and OC-768 (40 Gbps) (planned). It is crucial for carriers to know caracteristics of the traffic, and in particular the number of connections, for design network purposes. See [7] for a survey on Network monitoring.

5.1 Execution time

At 10 Gbps speed a new packet arrives every 240 ns., assuming an average packet size of 300 bytes, see [10] for a survey on monitoring very high speed links and [7] for packet size distribution. This allows only 595 operations per packet on the fastest processor (2.5 GHz) ignoring the significant time taken by the data to enter the processor. This reduces to 150 operations for a 40 Gbps link. Thus execution times of algorithms are crucial not only for their efficiency and but even for their feasability. Our algorithms are mainly composed of a very simple internal loop that finds the k th minimum of a multiset, see figure 4.2 for the case $k = 2$. We call *cost of an execution of an algorithm* the number of elementary operations, here the number of comparisons and assignments, done during it. We talk of *mean cost of an algorithm* to design the expectation of the cost of the diverse possible excutions.

Proposition 5.1 *We consider here that k is fixed. The mean cost to find the k th minimum of n uniform random variables on [0,1] is such as*
$$\mathbb{E}_n[\mathit{Cost}] \underset{n\to\infty}{=} n + o(n). \tag{25}$$

	m	4	8	16	32	64	128	256	512	1024
$\frac{1}{M^{(3)}}$	$\%th$	50	35.4	25	17.7	12.5	8.8	6.25	4.4	3.1
	$Random$	53.2	33.5	25.9	17.9	14.3	9.4	6.1	4.3	3.0
	$Ind-230m$	31.0	32.9	10.4	1.9	10.5	9.4	1.4	0.3	0.05
	$Auck-9M$	10.3	4.0	5.1	10.0	2.8	3.9	1.7	1.8	3.0
$\frac{1}{\sqrt{M^{(3)}}}$	$\%th$	36.3	25.7	18.1	12.8	9.1	6.4	4.5	3.2	2.3
	$Random$	38.4	26.5	18.0	13.4	9.0	6.2	4.6	3.1	2.1
	$Ind-230m$	27.9	18.0	5.9	1.9	10.7	11.4	2.3	0.5	0.15
	$Auck-9M$	10.6	4.7	2.1	4.7	5.2	0.08	0.3	2.1	2.9
$\ln \frac{1}{M^{(3)}}$	$\%th$	34.0	23.1	16.0	11.2	7.9	5.6	3.9	2.8	2.0
	$Random$	34.9	22.3	16.0	11.8	8.0	5.5	4.0	2.6	1.9
	$Ind-230m$	25.8	3.5	1.3	4.9	10.6	12.2	3.2	1.2	0.3
	$Auck-9M$	10.7	6.1	0.2	0.6	6.2	2.7	0.6	2.7	2.9

Fig. 2: Results of the simulations

The proof is trivial and omitted. For the first minimum, the number of comparisons is n as we compare each value with the temporary minimum and the number of assignments is H_{n+1} as the probability to have a new minimum is $\frac{1}{t+1}$ of the t-th value.

Fact 5.1 *Let $\mathcal{I}$ be the ideal multiset taken as input of size N and cardinality n. For almost all the configurations of repetitions, the mean cost, Cost, of the three families of algorithms is such as*

$$\mathbb{E}_N[\mathit{Cost}] \underset{n\to\infty}{=} N + o(N). \tag{26}$$

This has to be compared, for example, to the internal loop of the algorithm of Probabilistic Counting ([6]) or LogLog Counting ([3]) which have to find the first 0 of an infinite random binary number, what is done in average with three operations instead of only one. The main problem to state a result for all cases is that *the cost is sensitive to the structure of the repetitions*. When a hashed value of the sequence is processed, there are basically two cases. Either this value is bigger than the k th minimum and we do only one comparison, or it is smaller and we do also few comparisons to place it among the minima and few assignments to update the minima. For almost all the configurations of repetitions, the number of times this actualization is made is negligible and the mean cost is one comparison. But there exist cases where the repetitions of elements corresponding to the minima are no more negligible to the repetitions of all the other hashed values.

5.2 Validation of the algorithms

Data. To validate the three families of algorithms we ran some simulations using files of different kinds and sizes. These files are referred in Figure 2 as Random, Ind-230m and Auck-9M. The two last ones are traces available on the website of the NLANR Measurement and Network Analysis Group (http://moat.nlanr.net). Traces are sequences of anonymized headers of all the packets carried by a link during a given period of time. Our analysis consists in estimating the number of *distinct connections*, identified by the pair source-destination IP addresses, for each trace. Ind-230m (for 230,000 connections), has been collected in a POP in Indiana on an OC-3, for a window of time of 1 minute 30; Auck-9M has been collected in the University of Auckland for a window of time of 15 minutes. We know that 4,322,835 packets corresponding to 230,292 distinct connections were carried on the link in Indiana and 13,846,690 packets corresponding 9,083,474 connections on the link in Auckland. The third set of results, identified as Random, corresponds to mean results over simulations on 500 files of size 10,000. Each file was built with integers uniformly taken between 0 and $2^{32}-1$. We want here to estimate the number of distinct numbers in these files. In the table of Figure 2 each line (% th excepted) corresponds to results on a given input.

Protocol and Results. We ran estimators of each of the three families on these inputs. Table of Figure 2 shows typical results for the estimators built with the third minimum. The three horizontal blocks

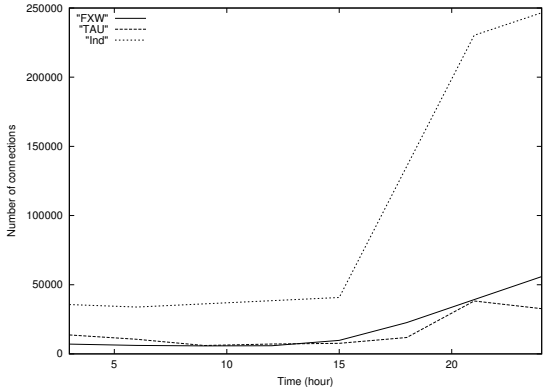

Fig. 3: Connections Peak during the Spreading of the Code Red Worm

correspond each to results for the estimator of one family. Each estimator was executed on each file several time with different numbers of buckets ($m = 4, 8, \ldots, 1024$). Therefore each column corresponds to a simulation of m experiments leading to a given precision. *A number in the table* is the difference in percentage between the estimate given by the algorithm and the exact value. For example the estimate of different connections in Ind-230m given by the algorithm built on $\frac{1}{M^{(3)}}$ for $m = 8$ is $306,058$. The exact value is $230,292$. The precision is $((306,058 - 230,292)/230292) \times 100 = 32.9\%$. The lines referred as *%th* indicate the standard error given by the theory. For example, for $m = 256$, we evaluate the number of connections respectively at a precision of 1.4 %, 4.7 % and 4.0 % for 6.25 %, 4.5 % and 3.9 % predicted by the theory. The third set of results, Random, corresponding to mean results over simulations on 500 files, validates the precision given in Theorem 4.1. As a matter of fact, the results are close to expected from the theory. For example for $m = 32$, we have 17.9, 13.1 and 11.8 for the three families to compare with 17.7, 12.8 and 11.2 expected. It validates the algorithms: the asymptotic regime is quickly reached and the hashing function distributes well the values.

5.3 Code Red attacks

We simulate also an analysis of internet traffic to show a typical use of the algorithms. We use here the proposed best algorithm, the estimator of the logarithm family using the third minimum. The NLANR Measurement and Network Analysis Group is doing daily network monitoring. We analyse their traces of July 19th 2001 when a Code Red Worm variant started spreading. The Code Red Worm was not designed to do a maximum of damage but to spread very fast. More than 359,000 computers were infected in less than 14 hours and at the peak of the infection frenzy, more than 2,000 new hosts were infected each minute. We considered three sets of traces: one monitored in the Indiana University MegaPOP (Ind), one in the FIX West facility at NASA Ames (FXW) and the last one in Tel Aviv University (TAU). The traces correspond to a window of 1 minute 30 every three hours. We use the algorithm to estimate within 4 % (m=256) the number of active connections using this link during each of these period of times. Results are shown in Figure 3. It is of course a very rough analysis and more data for other links, other days for example would be needed to give precise conclusions about the spread of the worm. But we are able to detect a change of the activity of the network caused by the infected hosts in the network. We see a very net increases of the number of active connections starting from 3 pm. For the Ind link, the usual load seems to be around 35,000 connections, 33842 at 6 am. At its peak at midnight we estimate a number of 246,558 connections, around 7 times more. Same observation for TAU and FXW: respectively 7,629 and 9,793 connections at 3 pm and 32,670 and 55,877 at midnight. So, by monitoring a link using our algorithm, we are able to see, using constant memory, unusual increase of the traffic, detect an attack and to give rough indications about its propagation and extent in some parts of the network.

Acknowledgements

I would like to thank my PhD advisor, Philippe Flajolet, for introducing me to the subject of the paper, his help and numerous remarks on this work.

References

[1] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 1–10. Springer-Verlag, 2002.

[2] Jeffrey Considine, Feifei Li, George Kollios, and John Byers. Approximate aggregation techniques for sensor databases. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 449, Washington, DC, USA, 2004. IEEE Computer Society.

[3] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *Proceedings of the European Symposium on Algorithms*, 2003.

[4] C. Estan, G. Varghese, and M. Fisk. Bitmap algorithms for counting active flows on high speed links. In *Technical Report CS2003-0738*, UCSE, Mars 2003.

[5] Philippe Flajolet. Adaptive sampling. In M. Hazewinkel, editor, *Encyclopaedia of Mathematics*, volume Supplement I, page 28. Kluwer Academic Publishers, Dordrecht, 1997.

[6] Philippe Flajolet and P. N. Martin. Probabilistic counting. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, pages 76–82. IEEE Computer Society Press, 1983.

[7] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, K. Papagiannaki, and F. Tobagi. Design and Deployment of a Passive Monitoring Infrastructure. In *Passive and Active Measurement Workshop*, Amsterdam, April 2001.

[8] Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. In *SIGMOD Conference*, 2001.

[9] Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *The VLDB Journal*, pages 541–550, 2001.

[10] G. Iannaccone, C. Diot, I. Graham, and N. McKeown. Monitoring very high speed links. In *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.

[11] Donald E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 1973.

[12] K.-Y. Whang, B. T. V. Zanden, and H. M. Taylor. A linear-time probabilistic counting algorithm for database applications. In *TODS 15, 2*, pages 208–229, 1990.