Order Statistics and Estimating Cardinalities of Massive Data Sets.

Frédéric Giroire

Algorithms Project - INRIA Rocquencourt

Alea 2006
Counting the number of distinct elements.

- Let $F$ be a multiset. We denote by $N$ its number of elements and by $n$ number of distinct elements or \textit{cardinality}.

- Complexity of Exact Counting Algorithms
  - memory: $O(n)$
  - time: $O(N \log n)$
Surprisingly long list of applications !!

- Linguistic (distinguish a TV Schedule from a Shakespeare book, find the author of a non attributed book...)

- Genome analysis (are coding regions more correlated than non coding regions ?)

- Databases (Requests of type, in how many different cities live people owning a red car, requests optimisation).

- Networking (Network Monitoring for Carriers, DoS attacks detection, Port Scan detection, Study of the spread of a Worm)
... with very large Multisets

- Human Genom with 3 billions of pairs of bases;
- Lots of databases with tens of millions of inputs (yellow pages...), Optimization of Requests;
- Networking: links of 40 Gbps capacity (OC-748 are deployed in SONET networks)
  - A packet may arrive every 60 ns. It gives to a 2 GHz computer only 120 CPU operations to handle a packet (assuming 300 bytes packets).
  - A hard disk of one TeraByte is filled in less than 4 minutes.
Specification of the problem.

- Estimate the number of distinct elements \( n \) (cardinality) of very large multisets.
- Using constant memory.
- Doing only one very simple pass on the data.
- No assumption made on the nature of repetitions.

The question is: How to find the cardinality without even storing the distinct elements?
No Magics, but probability

Lot of work on this topic.

Table of Contents

1. → Mincount

2. Small and very large cardinalities (Extension of Hit Counting)


4. Validation et Optimisation.

5. Applications
1. Mincount.

1.1 Principles of the algorithm.

1.2 The Three Families of Estimates.

1.3 Validation.
1.1: Why it is the place to be.

- Introduction of a new class of algorithms based on order statistics (rather than on bit patterns in binary representations of numbers) OSA.

- Analysis of three families of estimators. They attain a standard error of $\frac{1}{\sqrt{M}}$ using $M$ units of storage and are in the same class as best known algorithms so far. (Advertising: Distinct Sampling Gibbons VLDB 2001, 3% using 32 KB, 4 times less efficient than OSA.)

- Very simple internal loop, advantage in term of processing speed (only 2.8 to 4 times slower than wc).

- Validation on internet traffic traces.

- The minimum $M$ of a sequence of numbers: found with a single pass and not sensitive to repetitions.

- Ideal Multiset: $n$ distinct values in $[0, 1]$ chosen uniformly at random, replicated and shuffled.

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

- Indirect use of the minimum: sublinear functions (log and square root) and $k$ th minima.
1.1: Principles of Order Statistics Algorithms (2) -
Structure of the Algorithm.

- Existence of ’good’ hash functions \( h(x) \) chosen uniformly at random in \([0, 1)\).

- Algorithms. Input: an ideal multiset \( F \).
  Output: an estimate \( \xi \) of its cardinality.

- Simulation of \( m \) experiments: precision in \( \frac{1}{\sqrt{m}} \).
Flow

\[ h \]

\[ m = 4 \]
\[ k = 3 \]

Memory 12 minima

1st bucket: \( M_1^{(1)} \) \( M_1^{(2)} \) \( M_1^{(3)} \)

Estimate: function of log of

\( M_1^{(3)} \) \( M_2^{(3)} \) \( M_3^{(3)} \) \( M_4^{(3)} \)
1.2: The three families of estimates

Algorithm \((F: \text{ multiset of hashed values}; m)\)

for \(x \in F\) do

\[
\text{if } \frac{i-1}{m} \leq x \leq \frac{i}{m} \text{ do}
\]

actualize the \(k\) minima of the bucket \(i\) with \(x\)

return \(\xi := C^{st}g^{-1}\left(\frac{1}{m} \sum_{i=1}^{m} g\left(\frac{1}{M_i^{(k)}}\right)\right)\) as cardinality estimate.

Inverse Family \(g(x) = x\) \(C^{st} = m(k - 1)\)

Square Root Family \(g(x) = \sqrt{x}\) \(C^{st} = \frac{1}{\left(\frac{1}{(k-1)} + \frac{m-1}{(k-1)!^2} \Gamma(k-\frac{1}{2})^2\right)}\)

Logarithm Family \(g(x) = \log x\) \(C^{st} = m \cdot \left(\frac{\Gamma(k-\frac{1}{m})}{\Gamma(k)}\right)^{-m}\)
1.2: The logarithm of the third minimum.

Proposition 1. Consider the algorithm of the Logarithm Family built on the third minimum.

1. Its estimate defined as

\[ \xi := m2^m \cdot \Gamma \left( 3 - \frac{1}{m} \right)^{-m} \cdot e^{-\frac{1}{m}(\log M_1^{(3)} + \ldots + \log M_m^{(3)})} \]  \hspace{1cm} (2)

is asymptotically unbiased in the sense that

\[ \mathbb{E}[\xi] \sim n \quad \text{as} \quad n \to \infty. \]  \hspace{1cm} (3)

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

\[ \text{SE}[\xi] \sim \sqrt{\Gamma \left( 3 - \frac{1}{m} \right)^{-2m} \Gamma \left( 3 - \frac{2}{m} \right)^m - 1}, \quad m \geq 2. \]  \hspace{1cm} (4)
1.2: Comparison of the algorithms.

The precision at constant memory is the standard error expressed in function of the memory $M$.

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

$$ P_{\text{inv}}(M) \sim \sqrt{\frac{k}{k-2} \frac{1}{M}} \quad (5) $$

$$ P_{\text{sq}}(M) \sim \frac{2}{\sqrt{M}} \sqrt{\frac{k}{k-1} \left( \frac{\Gamma(k)}{\Gamma(k-\frac{1}{2})} \right)^2 - 1} \quad (6) $$

$$ P_{\text{log}}(M) \sim \sqrt{k \Psi'(k) \frac{1}{M}}, \text{ with } \Psi(z) = \frac{d}{dz} \Gamma(z) \quad (7) $$
1.2: Comparison

1. For any $k$, $\mathcal{P}_{log}(M) \leq \mathcal{P}_{sq}(M) \leq \mathcal{P}_{inv}(M)$.

   → Better estimates with square root or logarithm.

2. For the three families, precision at constant memory improves for larger $k$.

3. When $k$ goes to infinity, we have

   $$\mathcal{P}(M) \xrightarrow[k \to \infty]{} \frac{1}{\sqrt{M}}$$

   → Optimal trade-off between precision and memory for the three families: a standard error of $1/\sqrt{M}$.

4. Best estimate. Near optimal efficiency reached quickly ($\frac{1.09}{\sqrt{M}}$ for the logarithm family when $k = 3$).
### 1.3: Validation - Precision

<table>
<thead>
<tr>
<th></th>
<th>$m$</th>
<th>4</th>
<th>16</th>
<th>64</th>
<th>256</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$\frac{1}{M^{(3)}}$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td>50</td>
<td>25</td>
<td>12</td>
<td>6.2</td>
<td>3.1</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>53</td>
<td>25</td>
<td>14</td>
<td>6.1</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>Ind – 230m</td>
<td>31</td>
<td>10</td>
<td>10</td>
<td>1.4</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>Auck – 9M</td>
<td>10</td>
<td>5.1</td>
<td>2.8</td>
<td>1.7</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td><strong>$\frac{1}{\sqrt{M^{(3)}}}$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td>36</td>
<td>18</td>
<td>9.1</td>
<td>4.5</td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>38</td>
<td>18</td>
<td>9.0</td>
<td>4.6</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td>Ind – 230m</td>
<td>28</td>
<td>5.9</td>
<td>11</td>
<td>2.3</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>Auck – 9M</td>
<td>11</td>
<td>2.1</td>
<td>5.2</td>
<td>0.3</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td><strong>$\ln\frac{1}{M^{(3)}}$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td>34</td>
<td>16</td>
<td>7.9</td>
<td>3.9</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>35</td>
<td>16</td>
<td>8.0</td>
<td>4.0</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>Ind – 230m</td>
<td>26</td>
<td>1.3</td>
<td>11</td>
<td>3.2</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Auck – 9M</td>
<td>11</td>
<td>0.2</td>
<td>6.2</td>
<td>0.6</td>
<td>2.9</td>
<td></td>
</tr>
</tbody>
</table>
1.3: Validation - Timings

|                | \( N \)   | \( n \)  | \( wc \) | \( OSA \) | \( sort - u|wc \) |
|----------------|-----------|---------|---------|---------|-----------------|
| \textit{TAU.ip} | 680,000   | 10,000  | 0.05    | 0.14    | 15.41           |
| access – log – cut | 830,000   | 100,000 | 0.07    | 0.25    | 35.53           |
| \textit{3aleatoire.dat} | 5,000,000 | 4,993,301 | 0.20    | 0.79    | 56.             |
| \textit{auck.ip}  | 14,000,000 | 9,000,000 | 1.96    | 2.64    | 303.19          |

- Throughput: 3.3 to 6.3 M elements per second, 63 to 100 MB/s.
- OSA only \textbf{2.8 to 4.0 slower than wc}.

2. Limits of \textsc{Mincount} and Extension of Hit Counting
2. Extension of Hit Counting

Limites de Mincount (taille des flots à traiter).

- Le comptage des grandes cardinalités, collisions des valeurs hachées.

- Le comptage des petites cardinalités, boites ’vides’.

Deux types de solutions :


2. Apcount tronqué, on n’utilise que les boites ’pleines’.

- Context of networking: *data stream* more appropriate than *data set*:
  
  
  \[
  \text{(source IP, destination IP), time stamp)}.
  \]

- Request: how many distinct elements of the stream have been seen in the last window of one hour?
3. **Table of Content.**


3.1 **Presentation of a new algorithm, SLIDING MINCOUNT.** It is a new method to give an estimate of the number $n$ of active flows among packets in a data stream in a sliding window. An accuracy of order $\delta$ is attained using a memory of order $\frac{1}{\delta^2} \log(N)^2$ bits.

3.2 **Analysis.**

3.3 **Validation.**
3.1 Difficulties

• **Static Mincount** uses the minimum to build an estimate.

• **Question**: how to have this minimum for each window?

• For an OC-748, Assuming 300 byte packets, 60 billion packets may arrive in 1 hour.

• **No assumption on the traffic**.

• **Store all the hashed values that may become a minimum in the future**: seems impossible !!!
3.1 Maintaining the minimum over a sliding window.

• Solution: store all the packets that may become a minimum in the future: LFPM.

• Simple but crucial remark.
  Let $P_1 = (h_1, t_1)$ and $P_2 = (h_2, t_2)$ be two packets. If $t_1 < t_2$ and $p_1 \geq p_2$, then $P_1$ can not be minimum in the future.
3.1 LFPM and Minimum Records
3.1 The Internal Loop.

while(true) do
    Read packet \((H_n, t_n)\).
    Delete the packets \((H_i, t_i)\) of \(L\) with \(t_i < t_n - W\).
    Delete the packets \((H_i, t_i)\) of \(L\) with \(H_i \geq H_n\).
    Add \((H_n, t_n)\) at the end of \(L\).
end do

Figure 1: The internal loop to maintain the LFPM of a data stream.
3.1 Answer a Request.

REQUEST:

Input: “what is approximately the number of flows that have been active in the last $w$ units of time?”

Answer:

for $i$ from 1 to $m$ do

\[ M_i \leftarrow \text{leftmost hashed value of } L_i \text{ whose timestamp is larger than } t - w \]

end for

\[ \xi \leftarrow m \Gamma(2 - 1/m)^{-m} \exp \left( \frac{1}{m} (\ln(M_1) + \ldots + \ln(M_m)) \right) \]

return $\xi$
3.2: Analysis of the algorithm.

- The precision is the same as the one of the static algorithm.
- The memory size is the size of all the LFPM.
  - Distribution of the size of the LFPM at a fixed time.
  - Study of the maximal required memory over a long running time.
3.2: Distribution of the size of the LFPM at a fixed time.

**Theorem 2.** At a given time $t$, let $n$ be the number of distinct flows over the window $[t-W,t]$. Then the probability distribution of the size $L_n$ of the LPFM at time $t$ has the following characteristics

1. $$\mathbb{P}[L_n = k] = \frac{C_{n,k}}{n!},$$

where $C_{n,k}$ is the Stirling number of first kind.

2. $$\mathbb{E}[L_n] = H_n,$$

where $H_n$ is the harmonic number ($H_n = \sum_{k=1}^{n} \frac{1}{k} \sim \ln(n)$)

3. $$\mathbb{V}[L_n] = H_n - \sum_{k=1}^{n} \frac{1}{k^2} \sim \ln(n),$$
4. If $n$ is large enough, the distribution of the size $L_n$ of the LFPM is close to a gaussian law of expectation $\ln n$ and variance $\ln n$, that is

$$\frac{L_n - \ln(n)}{\sqrt{\ln(n)}} \xrightarrow{n \to \infty} \mathcal{N}(0, 1)$$
3.2: Memory used by the algorithm.

Theorem 3. At a given time $t$, let $n$ be the number of distinct flows over the window $[t - W, t]$. Then the probability distribution of the sum $L_n^{tot}$ of the sizes of the $m$ LFPM’s (one for each bucket) at time $t$ has the following characteristics:

1. 
   
   $\mathbb{E}[L_n^{tot}] \sim m \ln(n/m)$, 

2. 
   
   $\mathbb{V}[L_n^{tot}] \sim m \ln(n/m)$, 

3. When $n$ goes to infinity, the distribution converges (in law) to a gaussian law of expectation $m \ln(n/m)$ and variance $m \ln(n/m)$, that is

   \[ \frac{L_n^{tot} - m \ln(n/m)}{\sqrt{m \ln(n/m)}} \xrightarrow{n \to \infty} \mathcal{N}(0, 1) \]
3.2: Study of the maximal required memory over a long running time.

Number of changes of the LFPM is of the order of several billions!!!

**Theorem 4.** Let $n_{\text{max}}$ be the maximal number of distinct flows that can be observed over a window of length $W$. Consider a long but finite datastream, on which the internal loop of SlidingMinCount, and let $S$ be the number of packets in the datastream. Then the maximal value taken by the sum of the sizes of the $m$ LFPM's is of order at most

$$m \ln(n/m) + \sqrt{m \ln(n/m)2 \ln(S)},$$

i.e. it will never be larger than this value up to a few units.
3.3: Validation - Simulations.

We consider here an **ideal data stream**. We choose the window to have at any time $1M$ active flows in it. We observe the behaviour of **SLIDING MINCOUNT** over $5M$ packets of this data stream. The algorithm simulates $m = 2^{10}$ experiments.

Figure 2: Size of LFPM during an execution
Figure 3: Estimate of Sliding Mincount during an execution
Run for your train.
Distributed algo.

- The packet arrival rate may be too high and exceeds the capacity of a simple computer CPU. The algorithm can easily be distributed. Send a packet every 5 packets to each of the routers.

- Merge the 5 LFPMS.

- It applies especially well in a backbone network: each PoP is made of several routers interconnected in a star.
Applications.

1. Surveillance des réseaux.
2. Etude du génome.
Applications: Code Red attacks

Detection of denial of service attack.

Connections peak during the spreading of the Code Red Worm.
Applications Bioinfo.

- **Nature du génome** (Bernoulli, Markov...).

- **Etude d’homogénéité du génome ou existe-t-il des zones plus corrélées que d’autres ?** But : peut-on reconnaître
  - des zones codantes ;
  - de grandes répétitions ;
  - des zones non séquencées ?
Applications Bioinfo.

1. Nombre de sous-mots différents de taille 1 à 30 dans le chromosome 20

2. Apparition des motifs de tailles 6, 8, 10, 12 dans le chromosome 20

3. Mots différents de taille 13 dans des séquences consécutives du génome
Le génome est divisé en 10, 100 et 1000 séquences consécutives. La figure donne le nombre de sous-mots différents de taille 13 pour chacune des séquences.

Genome. Taille 13. Pieces de taille 300 M, 30 M et 3 M. \(4^{13} \approx 70M\)

Mots différents de taille 13 dans des séquences consécutives du génome