Quantitative Algorithmics of Massive Data Sets

Algorithmique quantitative de grands flux de données

Philippe Flajolet, INRIA, Rocquencourt

http://algo.inria.fr/flajolet
Routers in the range of Terabits/sec ($10^{14}$ b/s).

Google indexes 6 billion pages & prepares to index 100 Petabytes of data ($10^{17}$ B).

Can get a few key characteristics, QUICK and EASY

This talk: Zoom on some recent research directions ≠ geopolitics.
+ Interplay between theory and practice.
Combinatorics + algorithms + probabilities + analysis are useful!
Traces of attacks: Number of active connections in time slices.

(Raw ADSL traffic)

Incoming/Outgoing flows at 40Gbits/second. Code Red Worm: 0.5GBytes of compressed data per hour (2001). CISCO: in 11 minutes, a worm infected 500,000,000 machines.

Left: ADSL FT@Lyon $1.5 \times 10^8$ packets (21h–23h). Right: (Estan-Varghese-Fisk) different incoming/outgoing connections
The situation is like listening to a play of Shakespeare and at the end *estimate the number of different words*.

**Rules:** Very little computation per element scanned, very little auxiliary memory = **Stream algorithms**.

---

From Durand-Flajolet, LogLog Counting (ESA-2003):

Whole of *Shakespeare*, $m = 256$ small “bytes” of 4 bits each = 128 bytes

$$\text{Estimate } n^\circ \approx 30,897 \text{ vs } n = 28,239 \text{ distinct words. Error: } +9.4\% \text{ w/ 128 bytes!}$$
— Routers & Networks: attacks, flow monitoring & control

— Databases: Query optimization = estimating the size of queries; also “sketches”.

— “Raw” statistics on massive data: on the fly, fast and with little memory even on textual data ∈ “data mining”.
(Gains by a factor of 400 in data mining of the internet graph.)

 Produce short algorithms & programs with $O(10)$ instructions based on mathematically oriented engineering.
• **Estimating characteristics of large data streams**
  — counters: “beats” information theory;
  — sampling: avoid negative effects
  — cardinality estimators;
  — gather statistics for mice and elephants;
  — profile indicators \((F_2, \text{etc})\)

\[ \sim \diamond \text{Gains by factors in range 100-1000} (!) \]

• **No analysis \implies no algorithm!** Algorithms are inherently probabilistic.
  — analytic combinatorics, complex asymptotics, Mellin transforms
  — Interplay of analysis and design \(\sim\) highly optimized algorithms.
1 **ICEBERGS (& COMBINATORICS)**

abcdaaaaabcafccecaacccaaacacbbbbbaabbbbbbb

Definition: A $k$-iceberg is present in proportion $> 1/k$.

One pass detection of icebergs for $k = 2$ using 1 registers is possible.

— Trigger a gang war: equip each individual with a gun.
— Each guy shoots a guy from a different gang, then commits suicide: Majority gang survives.
— Implement sequentially & adapt to $k \geq 2$ with $k - 1$ registers. (Karp, Shenker, Papadimitriou. 2003)
How to find an integer while posing few questions?

— Ask if in (1—2), (2—4), (4—8), (8—16), etc?
— Conclude by binary search: cost is \(2^{\log_2 n}\).

The **dyadic paradigm** for unbounded search:

- Ethernet proceeds by period doubling + randomization.
- Wake up procedures for **mobile communication** (Lavault+)
- **Adaptive data structures**: e.g., extendible hashing tables.

♡ **Approximate Counting**
Approximate counting & randomization

The oldest algorithm (Morris CACM:1977), analysis (F, 1985).

Maintain counter subject to $X := X + 1$.

Algorithm: Approximate Counting

Initialize: $C := 1$;

Increment: do $C := C + 1$ with probability $2^{-C}$;

Output: $2^C - 2$. 
Expect $C$ near $\log_2 n$ after $n$ steps, then use only $\log_2 \log n$ bits.

**Theorem:** Using alternate bases, count till $n$ probabilistically using $\log_2 \log n + \delta$ bits, with accuracy about $0.59 \cdot 2^{-\delta/2}$.

Beats information theory(!?): 8 bits for counts $\leq 2^{16}$ w/ accuracy $\approx 15\%$. 
Symbolic methodology:

Generating Functions:

\[ (f_n) \mapsto f(z) := \sum_n f_n z^n. \]

\[
\begin{align*}
& (a_1^*) b_1 (a_2^*) b_2 (a_3^*) \\
\Rightarrow & \frac{1}{1-a_1} b_1 \frac{1}{1-a_2} b_2 \frac{1}{1-a_3} \\
& \text{since } \frac{1}{1-f} = 1 + f + f^2 + \cdots \simeq (f)^*.
\end{align*}
\]

Perform probabilistic valuation \(a_j \mapsto q^j; b_j \mapsto 1 - q^j:\)

\[
H_3(z) = \frac{q^{1+2} z^2}{(1 - (1 - q)z)(1 - (1 - q^2)z)(1 - (1 - q^3 z))}.
\]

Approximate Counting

Mean\( (X) - \log_2 n: \)

Asymptotic methodology: Mellin transform (FISe*)

\[
f^*(s) := \int_0^\infty f(x)x^{s-1} \, dx.
\]

Need singularities in complex plane.

Dyadic superpositions of models \(\Rightarrow\) Asymptotics with fluctuations.

Mellin: Probabilistic counting, loglog counting + Lempel-Ziv compression (Jacquet-Szpa) + dynamic hashing + tree protocols (Jacquet+) + Quadtries &c.
Cultural flashes

— Morris (1977): Counting a large number of events in small memory.
— The power of probabilistic machines & approximation (Freivalds)
— The TCP protocol: Additive Increase Multiplicative Decrease (AIMD) leads to similar functions (Robert et al, 2001+)
— Probability theory: Exponentials of Poisson processes (Yor et al, 2001)
— Ethernet is unstable (Aldous 1986) but tree protocols are stable! Cf (Jacquet+).
Probabilities: Coupons & Birthdays

Let a flow of people enter a room.

— Birthday Paradox: It takes on average 23 for a birthday collision

— Coupon Collector: After 365 persons, expect a partial collection of \(~231\text{ different days}\) in the year; it would take more than 2364 to reach a full collection.

\[
\begin{align*}
\sqrt{\frac{\pi M}{2}} & \\
M & \\
M \log M & 
\end{align*}
\]

(Throw \(N\) balls into \(M\) bins)

Semi-clasical occupancy problems; generating functions; saddle points.
Birthday Paradox Counting (Gedanken!)

Take your space ship to remote planet.
How many days \( (M) \) in the year?

**Gedanken Algorithm:**

— Interview people:

39 Propergol, 42 Sargol, 11 Mordor, ..., 42 Sargol

— Record \( B = \text{time till first birthday collision} \);

— \( \mathbb{E}(B) \sim \sqrt{\frac{\pi M}{2}} \implies \text{return estimate } M \approx \frac{2B^2}{\pi} \);

— do some maths and improve!

*Sublinear algorithms ...*
Randomization and hashing

Randomization is a major algorithmic paradigm.

- Cryptography (implementation, attacks, cf RSA)
- Combinatorial optimization (smoothing, random rounding).
- Hashing and direct access methods
  - Produce (seemingly) uniform data from actual ones;
  - Provide reproducible chance

\[ \text{To be or not to be...} \sim \]
3 HIT COUNTING

First Counting Algorithm: Estimate cardinals \( \equiv \# \) of distinct elements. Based on Coupon Collector phenomena & motivated by query optimization in data bases. (Whang\(^+\), ACM TODS 1990)

![Diagram of hit counting algorithm]

**Algorithm Hit Counting**
Set up a table \( T[1..m] \) of \( m \) bit-cells.
— for \( x \) in \( S \) do mark cell \( T[h(x)] \);
Return \( -m \log V \), where \( V := \) fraction of empty cells.
— Algorithm is *independent of replications.*

— Let \( n \) be sought cardinality and \( \alpha := n/m \), *filling ratio.* Expect
  \[ V \approx e^{-\alpha} \] fraction of empty cells, by classical occupancy theory.

— Distribution is concentrated. **Invert:** \( n \approx m \log(1/V) \).

**Linear storage complexity:** Count cardinalities till \( N_{\text{max}} \) using
\[ \frac{1}{10} N_{\text{max}} \] bits, for accuracy (standard error) = 2%.

Generating functions for occupancy; Stirling numbers; basic depoissonization.
4 SAMPLING

Classical sampling (Vitter, ACM TOMS 1985)

Algorithm Reservoir Sampling (with multiplicities)
Sample $m$ elements from $S = (s_1, \ldots, s_N)$; ($N$ unknown a priori)
Maintain a cache (reservoir) of size $m$;
— for each coming $s_{t+1}$:
    place it in cache with probability $m/(t+1)$; drop random element;
Sample values (i.e., without multiplicity)? (Wegman+F).

**Algorithm:** Adaptive Sampling *(without multiplicities)*
Get a sample of size $\leq m$ according to values.
— Increase sampling depth and decrease sampling probabilities.

Sample of size $\leq m$:
depth $d = 0, 1, 2, \ldots$

Also: get $b$-sample by mild oversampling ($m = 4b$).

Analysis makes use of randomness digital trees, generating functions and Mellin transforms.
Adaptive Sampling = Second counting algorithm for cardinalities.

Let $d :=$ sampling depth; $\xi :=$ sample size.

**Theorem:** $X := 2^d \xi$ estimates the cardinality of $S$ using $m$ words: unbiased, with standard error $\approx 1.20/\sqrt{m}$.

- $1.20 \approx 1/\sqrt{\log 2}$: with $b = 1,000W$, get 4% accuracy.
- Related to folk algorithm for leader election on channel: “Talk, flip coin if noisy; sleep if Tails; repeat!”
- Related to “tree protocols with counting” $\gg$ Ethernet. Cf (Greenberg-F-Ladner JACM 1987).
Cardinality Estimators

\[ F_0 = \text{Number of different values} \]

- 1983–1985: (F-Martin, FOCS+JCSS) Probabilistic Counting
- 1987–1990: (Whang et al) Hit Counting
- 1984–1990: (Wegner) (F90 COMP) Adaptive Sampling
- 1996: (Alon et al, STOC) \( F_p \) statistics \( \sim \) later
- 2000: (Indyk FOCS) Stable Law Counting \( \sim \) later
- 2001: (Estan-Varghese SIGCOMM) Multiresolution Bitmap
- 2003: (Durand-F ESA) Loglog Counting
- 2005: (Giroire) MinCount
- 2006: (DFFM) HyperLoglog

Note: suboptimal algorithms can be useful!
5 PROBABILISTIC COUNTING

Observables of hashed values!

Third Counting Algorithm
for cardinalities

0.1xxxx → \( P = \frac{1}{2} \)
0.01xxx → \( P = \frac{1}{4} \)
0.001xx → \( P = \frac{1}{8} \)

Algorithm: Probabilistic Counting
Input: a stream \( S \); Output: cardinality \( |S| \)
For each \( x \in S \) do /* \( \rho \) ≡ position of leftmost 1-bit */
    Set BITMAP[\( \rho(\text{hash}(w)) \)] := 1; od;
Return \( P \) where \( P \) is position of first 0.
— $P$ estimates $\log_2(\varphi n)$ for $\varphi \doteq 0.77351 = \frac{e^\gamma}{\sqrt{2}} \prod_{m \geq 2}^* m^{\varepsilon(m)}$, $\varepsilon(m) := (-1)^{\sum \text{bits}(m)}$.

**A.** Average over $m$ trials $A = \frac{1}{m} [A_1 + \cdots + A_m]$; return $\frac{1}{\varphi} 2^A$. Error $\approx \frac{1}{\sqrt{m}}$.

**B.** Stochastic averaging needs only **one** hash function.

\[
x \mapsto h(x) \mapsto \begin{cases} 
00 & \text{count } n_{00} \\
01 & \text{count } n_{01} \\
10 & \text{count } n_{10} \\
11 & \text{count } n_{11}.
\end{cases}
\]

**Implementation:** Hash + switch + update.

**Theorem [FM]:** Prob. Count. is asymptotically unbiased. Accuracy is $\frac{0.78}{\sqrt{m}}$ for $m$ Words of size $\log_2 N$. 
Data mining of the Internet graph
(Palmer, Gibbons, Faloutsos\textsuperscript{2}, Siganos 2001)
Internet graph: 285k nodes, 430k edges.

For each vertex $v$, define ball $B(v; R)$ of radius $R$.
Want: histograms of $|B(v, R)| \ R = 1 \ldots 20$
Get CPU 400 speedup factor (minutes rather than day)

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{histogram.png}
\caption{Histogram of diameters}
\end{figure}

+ Sliding window usage (Motwani et al) (Fusy & Giroire for MinCount).
6 LOGLOG COUNTING

Fourth Counting Algorithm for cardinalities: (Durand-F, 2003/DFFM, 2006)

- Hash values $\rightarrow \rho(h(x)) = \text{position of leftmost 1-bit} = \text{a geometric RV } G(x)$.
- To set $S$ associate $R(S) := \max_{v \in S} G(v)$.

Max of geometric RVs are well-known (Prodinger*). $R(s)$ estimates $\sim \log(\hat{\varphi} \text{card}(S))$, with $\hat{\varphi} := e^{-\gamma} \sqrt{2}$.

- Do stochastic averaging with $m = 2^{\ell}$:

Return $\frac{m}{\hat{\varphi}} 2^{\text{Average}}$. 
Engineering HyperLogLog

©Durand+F+Fusy+Meunier, 2006.

++ Optimize by limiting effect of discrepant values: use harmonic means of $2^R \sim$ HyperLogLog (≪Chassaing-Gerin, 2006). Asymptotically near-optimal?.

+ Correct nonasymptotic regime by switching to Hit Counting.

+ Extend beyond nominal capacity by random allocation correction.

**Theorem.** HyperLogLog needs $m$ “bytes”, each of length $\log_2 \log N$.

Accuracy is: $\frac{1.05}{\sqrt{m}}$.

**Distributed implementation:** exchange registers (a few kilobytes) and can even **compress**, to reduce storage by extra factor of 3.
Whole of Shakespeare:

\[ m = 256 \text{ small “bytes” of 4 bits each} = 128 \text{bytes} \]

Estimate \( n^\circ \approx 30,897 \) against \( n = 28,239 \) distinct words
Error is +9.4% for 128 bytes(!!)
**Features:** Errors \( \approx \text{Gaussian} \), seldom more than \( 2 \times \) standard error.

**Mahābhārata:** 8MB, 1M words, 177601 diff.

**HTTP server:** 400Mb log pages 1.8 M distinct req.

<table>
<thead>
<tr>
<th>( m )</th>
<th>( 2^6 ) (50by)</th>
<th>( 2^{10} ) (0.8kby)</th>
<th>( 2^{14} ) (12kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs:</td>
<td>8.9%</td>
<td>2.6%</td>
<td>1.2%</td>
</tr>
<tr>
<td>( \sigma ):</td>
<td>11%</td>
<td>2.8%</td>
<td>0.7%</td>
</tr>
</tbody>
</table>
Example: Mining the human genome

(by Frédéric Giroire)

Patterns of length 13 in sections of human genome (3Gbp).
Number of different patterns of length ℓ in Chr. 20 (60Mbp).
Summary

Analytic results ($\lg \equiv \log_2$): Alg/Mem/Accuracy

<table>
<thead>
<tr>
<th>CouponCC</th>
<th>AdSamp</th>
<th>ProbC</th>
<th>LogLog</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\approx \frac{N}{10}$ bits</td>
<td>$m \cdot \lg N$ Words</td>
<td>$m \cdot \lg \frac{N}{m}$ Words</td>
<td>$m \cdot \lg \lg \frac{N}{m}$ Bytes</td>
</tr>
<tr>
<td>$\approx 2%$</td>
<td>$\frac{1.20}{\sqrt{m}}$ W</td>
<td>$\frac{0.78}{\sqrt{m}}$ W</td>
<td>$\approx \frac{1.30-\sqrt{1.05}}{\sqrt{m}}$ By</td>
</tr>
</tbody>
</table>

$F_0$ statistics, $N = 10^8$ & 2% error

— Coupon Collector Counting = 1 Mbyte + used for corrections
— Adaptive Sampling = 16 kbytes + sampling, unbiased
— Probabilistic Counting: = 8 kbytes + sliding window
— Multiresolution bitmap (analysis?) = 5 kbytes?
— MinCount ©Giroire = 4 kbytes + sliding window
— Loglog Counting = 2 kbytes + mice/elephants
DoS attacks: Largest elephants; (here) elephant = number of packets to a given destination.

— By LogLog Counting and $p$-filtering: estimate $\text{card}(\Phi)$ and $\text{card}(\Phi_{pM} \cup \Phi_{pE})$ (Jean-Marie+Gandouët06).
— By Bloom filters (Azzana+Robert+Chabchoub)
Alon-Matias-Szegedy: \( F_p := \sum_v (f_v)^p \) where \( f_v \) := frequency of value \( v \).

\[ \dim = n \quad \longrightarrow \quad \dim \approx \log n \]

Johnson–Lindenstrauss embeddings
dimension reduction
\textbf{Indyk}’s beautiful ideas

Use of random Gaussian projections for \( F_2 \); Stable laws for \( 0 \leq p \leq 2 \).
Indyk’s $F_p$ algorithm

- Stable law of parameter $p \in (0, 2)$: $\mathbb{E}(e^{itX}) = e^{-|t|^p}$.
No second moment; no 1st moment if $p \in (0, 1)$.

$c_1 X_1 + c_2 X_2 \overset{\mathcal{L}}{=} \mu X$, with $\mu := (c_1^p + c_2^p)^{1/p}$.

**Algorithm:** Frequency Moments $F_p$;

Initialize $Z := 0$;
For each $x$ in $S$ do $Z := Z + \text{Stable}_\alpha(x)$.
Return $Z$.
Estimate $F_p$ parameter from $m$ copies of $Z$-values.

Remark: Use of $\log(|Z|)$ to estimate seems better than median(?)
Conclusions

For huge data streams, using reduced/minimal storage, one can:
— Sample positions and (distinct) values;
— Count events and cardinalities \( (F_1, F_0) \);
— Estimate profiles of data \( (F_p), 0 < p \leq 2 \);
— Get informations on elephants and mice.

The algorithms are based on randomization \( \mapsto \) Analysis fully applies
— They tend to work exactly as predicted on real-life data;
— They often have a very elegant structure;
— Their analysis involves numerous methods of AofA: “Symbolic modelling by generating functions, Singularity analysis, Saddle-point and analytic depoissonization, Mellin transforms, Stable laws, etc.”
That's All, Folks!