Space Efficient Mining of Multigraph Streams

Graham Cormode
cormode@bell-labs.com

S. Muthukrishnan
muthu@cs.rutgers.edu

Data Streams

Many large sources of data are generated as streams of updates:

- IP Network traffic data
- Text: email/IM/SMS/weblogs
- Scientific/monitoring data

Must analyze this data which is high speed (tens of thousands to millions of updates/second) and massive (gigabytes to terabytes per day)

Answers can be approximate with guarantees

Database Work

Much work in databases on processing streams

Many emerging systems: Gigascope, STREAM, Telegraph, Aurora, NiagaraCQ etc.

Much work on (approximate) query answering: building primitives for different queries.

Most work so far has assumed simple model of data: sets, vectors, flat files etc.

Often data has more complex structure eg graph

Graph Model

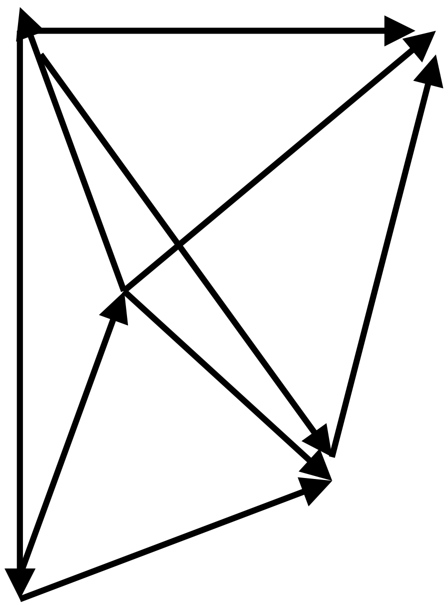
Consider network traffic data: defines a communication graph

eg edge: (source, destination)

or edge: (source:port, dest:port)

Defines a (directed) multigraph

We are interested in the underlying (support) graph on n nodes



Want to focus on number of distinct communication pairs, not size of communication

Multigraph Problems

Let $G[i,j] = 1$ if (i,j) appears in stream: edge from i to j. Total of m distinct edges

Let $d_i = \Sigma_{j=1}^{n} G[i,j]$: degree of node i

Find aggregates of d_i's:

- Estimate heavy d_i's (people who talk to many)
- Estimate frequency moments: number of distinct d_i values, sum of squares
- Range sums of d_i's (subnet traffic)
- Quantile queries over d_i's (applications in sensors)

Can't we just…

Can't we just use standard stream techniques… sketches, samples, etc.?

No – challenge is dealing with repetition of edges.

Want to ensure that, eg, sending a message to 1,000 others is much more visible than sending 1,000,000 messages to 1 other…

Existing techniques will only see the large volume, not large number of distinct destinations…

Primitives

Use approximate distinct counters as black box.

Eg, Flajolet-Martin (FM) sketches, Gibbons' distinct sampling [Gib01].

Input: multiset of items

Output: F_0 = number of distinct items to $(1 \pm \varepsilon)$ factor with probability $1-\delta$.

Cost: Requires space $O(1/\varepsilon^2 \log 1/\delta)$

(very small for moderate ε and tiny δ)

Can't we just…

Can't we just plug distinct counters into some existing algorithms and run on multigraph streams?

No – there's no guarantee of correctness / space bounds.

Need a more careful approach, and proof.

Will see several attempts to do this, which fail both in theory and in practice

Heavy Hitters Distinct

Find i's such that $d_i > \phi \sum_i d_i$
 Finds the people that talk to many others

Indicates unusual net activity (port scans, worms)

Can try to take existing HH algs and put in approximate counter data struture:

- Count Sketch [CCFC02]
- Count Min Sketch [CM04]
- Lossy Counting [MM02]

Can’t we just…

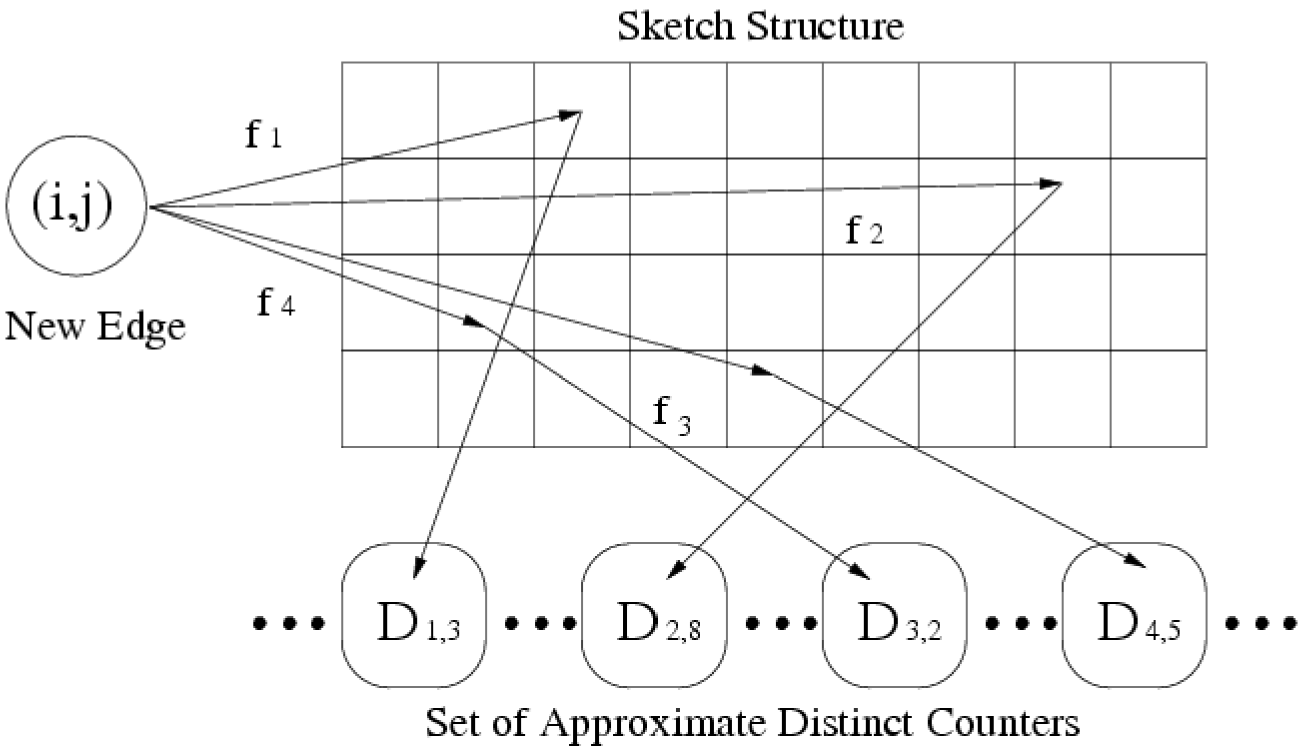
Can’t we just pick any of these?

No:

- Count sketch
  Relies on adding and subtracting counts.
  But subtracting two approximate counts doesn’t give good estimate of the difference
- Lossy counting
  Also need to subtract/compare and delete
  Can’t show correctness/space bounds

Count-Min + Count-Distinct

Count-Min sketch only uses additions, so can apply:



Correctness / Accuracy

Focus on point query accuracy: estimate d_i.

Prove estimate has only small bias in expectation:

$$\begin{aligned} \mathrm{E}(|D[k,f(i)]|) &= (1\pm\varepsilon)(\textstyle\sum_{j,f(j)=f(i)} d_j) \\ &= (1\pm\varepsilon)d_i + (1\pm\varepsilon)\textstyle\sum_{j\neq i,f(j)=f(i)} d_j \\ &= d_i \pm \varepsilon d_i + (1\pm\varepsilon)\tfrac{\varepsilon(m-d_i)}{2}. \end{aligned}$$

$$\Pr[|D[k,f(i)]| - (1\pm\varepsilon)d_i > \varepsilon(1\pm\varepsilon)(m-d_i)] < \tfrac{1}{2}$$

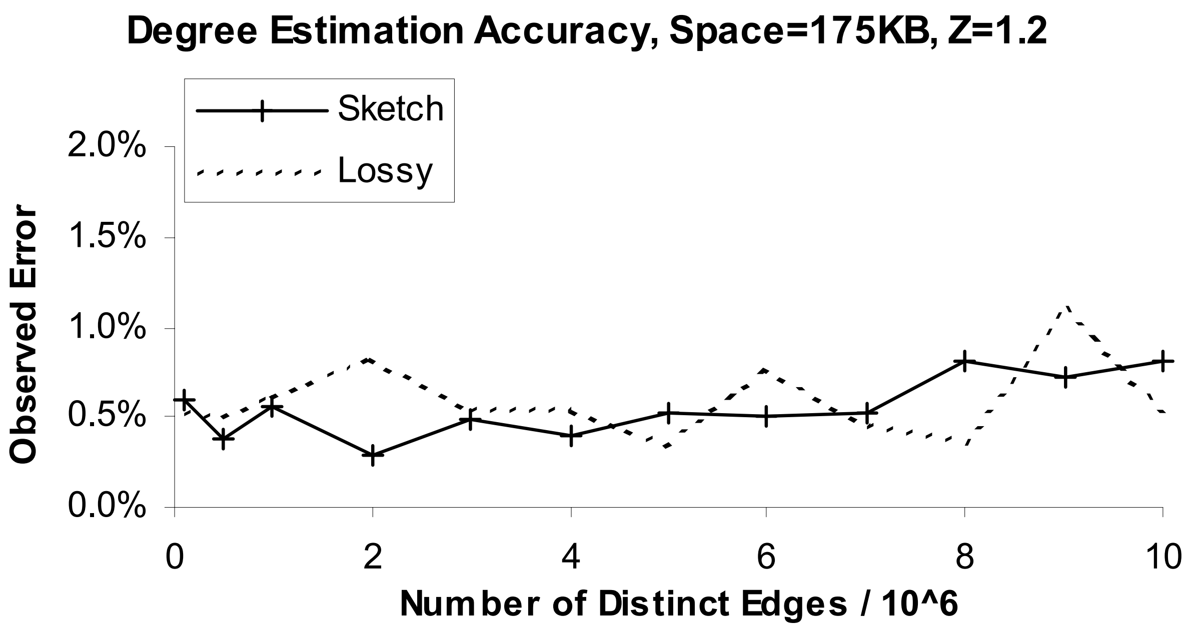
$$\Pr[|D[k,f(i)]| = d_i + (1\pm\varepsilon)(\varepsilon m)] > \tfrac{1}{2}$$

So probability that minimum of log n repetitions is still bad is very small (< 1/n)

Result

Can estimate any d_i given i with error εm in space $O(\varepsilon^{-3} \log^2 n)$. Time per update is $O(\log^2 n)$.

Use this to find heavy d_i's: for each update (i,j), test whether i is heavy.



Frequency Moment Estimation

Second frequency moment estimation (F_2) is key to many streaming algorithms.

Equivalent to self-join size over relations.

Define this as $M_2 = \sum_{i=1}^{n} d_i^2$

On graph, informs about neighborhood size: d_i^2 is (roughly) node pairs having path through i

Can't we just...

Can't we just:

- Use AMS sketch, replacing counters with approximate counters for +1s and −1s
  - No - making the estimate requires subtractions
  - Accuracy is not guaranteed
  - Works badly in practice too
- Pick some nodes by sampling from domain and track info about these exactly
  - No - expectation correct, but variance too high.
  - Resulting estimate is way off.

Minwise hashing

Use a technique based on min-wise hashing

Allows us to sample almost uniformly from set of edges

- For each edge in stream, compute $h((i,j))$
- Store info on v if $h((v,j))$ is smallest so far
- Collect every edge (v,j) matching v until more than $1/\varepsilon^2$, then switch to approx counting
- Estimate of M_2 is $m * (2d - 1)$, d = number of edges seen matching on v

Can’t we just…

Can’t we just use the approximate counting from the get-go?

No – we need slightly more accuracy when the number of edges is smaller.

Fortunately, we can keep small number of edges exactly, switch over at threshold, and space required is asymptotically the same.

Results

Expectation of estimate = $(1+\varepsilon)M_2$

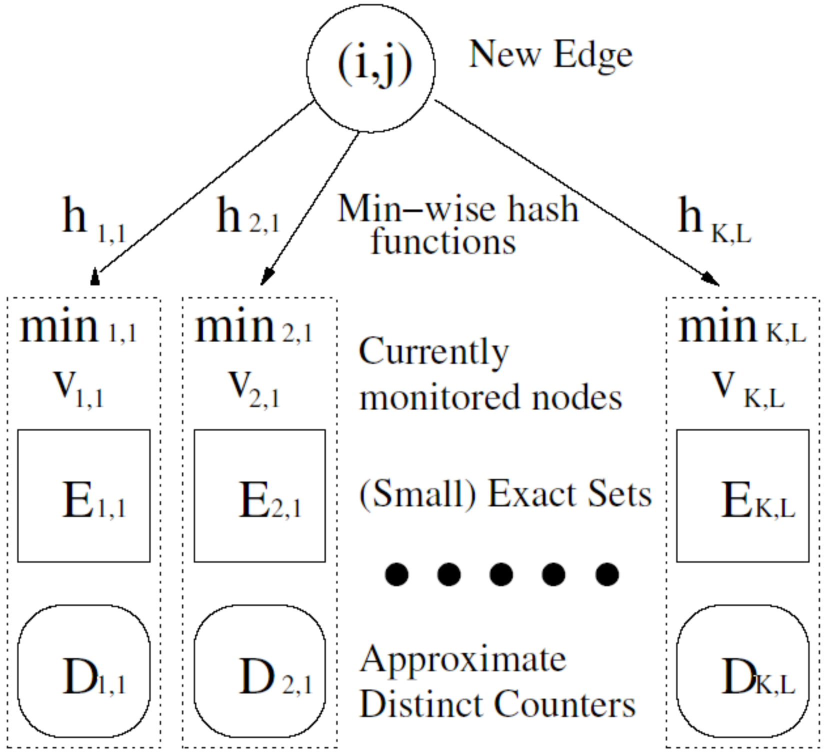
Variance = $(1+\varepsilon)n^{1/2}M_2^2$

See paper for details

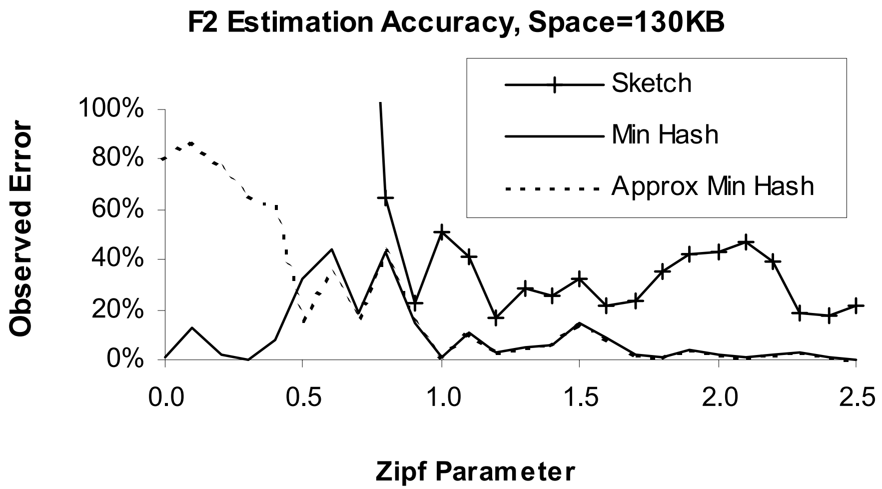
Repeat with different hash functions enough times to increase accuracy.

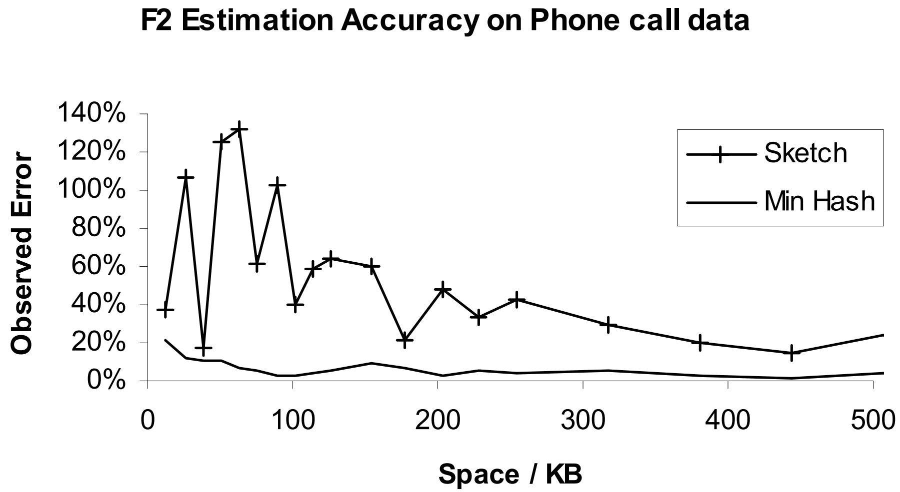
Space = $O(\varepsilon^{-4}\ n^{1/2} \log n)$

Small space sufficient in practice.



Experimental Study





Importance of using provable methods is shown.

Plausible heuristics often get terrible accuracy

Extensions

- We applied similar techniques to other problems: range queries, quantile queries (details in paper)
- "Duplicate insensitivity" also important in, eg sensor networks where results are broadcast (see tech report by Kollios et al)
- Problems such as $(F_2\ (F_0))$: cascaded aggregates Other cascaded aggregates are interesting, eg $F_2(F_2)$, $\mathrm{Median}(F_2)$ etc… arbitrary aggregates
- Some results extend to sliding window and arbitrary deletions case, M_2 still open

Conclusions

- Now we have results for many basic aggregates in data streams, applications such as graph streams require "cascaded aggregates"
- Naively combining results doesn't just work: they fail both in theory and in practice
- Careful combinations and proofs needed to get accurate solutions.