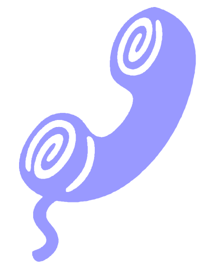
Processing Graph Streams: Upper and Lower Bounds

Graham Cormode

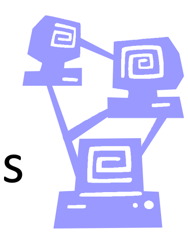
graham@research.att.com

A tale of three graphs



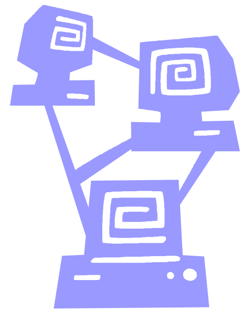
- The telephone call-graph
  - Each edge denotes a call between two phones
  - 2-3 $\times$ 10^9 calls made each day in US, maybe 0.5 $\times$ 10^9 phones
  - Can store this information (for billing etc.)

- The web graph
  - Each edge denotes a link from one web page to another
  - > 10^{10} pages, > 10^{11} billion links
  - Store pages (nodes) in memory, but maybe not all links



- The IP graph
  - Each edge denotes communication between IP addresses
  - 10^9 packets/hour/router in a large ISP, 2^{32} possible addresses
  - Not feasible to store nodes or edges

Example: IP Network Data



- Networks are sources of massive data: the metadata per hour per router is gigabytes
- Fundamental problem of data stream analysis:
  Too much information to store or transmit
- So process data as it arrives: one pass, small space:
  the *data stream* approach
- Approximate, probabilistic answers to many questions are OK
  - if there are guarantees of result quality

Models of Graph Streams

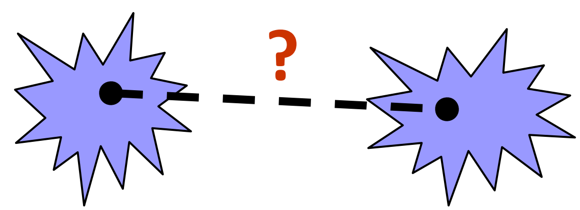
- Let $G=(V,E)$ be a graph with $|V|=n$ nodes, $|E|=m$ edges…
- We observe the edges of G in a stream, one by one
- How many times do we see each edge?
  - Exactly once (convenient, but unrealistic in some cases)
  - Many times, contributing to a weighted edge case
  - Many times, but should only count once (trickiest)
- What order do we see the edges in?
  - Grouped by incident vertex (incidence order)
  - In arbitrary (random) order
  - In arbitrary (worst-case) order
- How many passes over the data can we take (one, or many?)

Outline

- Graph Streaming Models
- Hardness Results
- Degree Sequence Computations
  - Application to counting triangles
- Semi-streaming model
- Multigraph model

Negative Results

- We care about how much space is needed to compute functions
- If graph is big, space should be sublinear in m or n
  - Sublinear in m, or streaming model collapses
- Many natural properties need at least linear in n space
- Given some (binary) property P, decision problem is to report whether P holds or does not hold on G
- Say P is "balanced" if $\forall n=|V|\ \exists$ constant c, G and $u \in V$, s.t.
  - There are $\geq cn$ v's s.t. $E \cup (u,v)$ has P
  - There are $\geq cn$ v's s.t. $E \cup (u,v)$ has $\neg P$

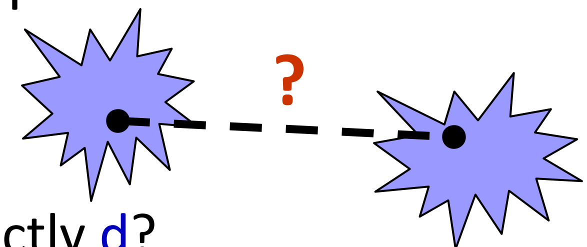


Hardness Proof

- Theorem: Deciding P in one pass requires $\Omega(n)$ space
  - Take a binary string x of length cn
  - Relabel vertex set $v_1 \ldots v_{cn}$ so that $E \cup (u, v_i)$ has $P \Leftrightarrow x_i = 1$
  - Assume there exists an algorithm using $o(n)$ space to test P
  - Feed this G to the claimed algorithm
  - Now for any i, feed (u, v_i) to the algorithm, and test
  - Result of test (correctly) recovers $x_i \Rightarrow$ must use $\Omega(n)$ space
- Holds even allowing the algorithm a constant prob of failure
  - Formally, reduce to INDEX problem in communication complexity
  - Generalizes to $\Omega(n/p)$ lower bound in p passes over input

Consequences

- Easy to see that the following properties are balanced:
  - Connectedness
  - Bipartiteness
  - Is there a vertex with degree exactly d?



- All these problems are solved easily in Õ(n) space
  - E.g. connectedness: just track the components, and merge
- Is anything non-trivial possible on a graph in o(n) space?

Outline

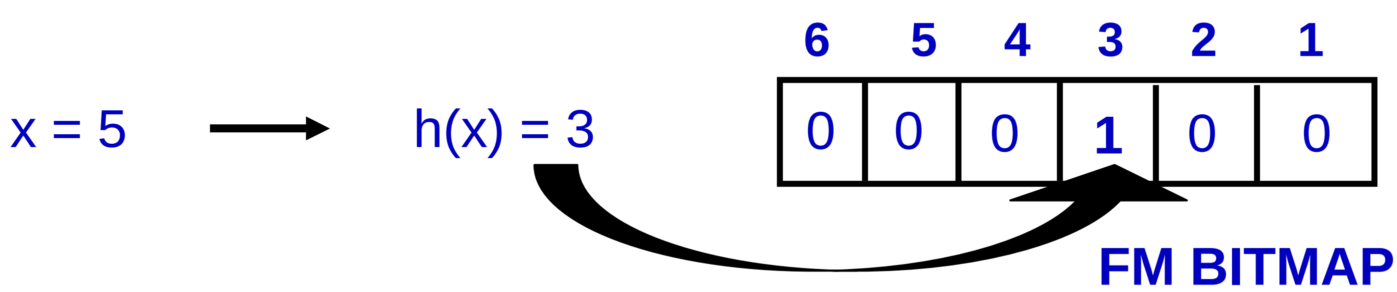
- Graph Streaming Models
- Hardness Results
- Degree Sequence Computations
  - Application to counting triangles
- Semi-streaming model
- Multigraph model

Degree Sequence Computations

- Positive results: track properties of edges and degrees
  - Given graph G, let d_v denote degree of node v
  - Frequency moments: $F_k = \sum_{v \in V} d_v^k$
  - Frequent items: find all v s.t. $d_v > \phi F_1$ for $\phi < 1$
- Can solve these problems in $\tilde{O}(1)$ space in one pass!
- Will summarize results for:
  - F_0: how many nodes with non-zero degree are seen?
  - Estimate d_v (allows finding frequent items)
  - F_2: sum of squares of degrees ($\approx$ paths of length 2)

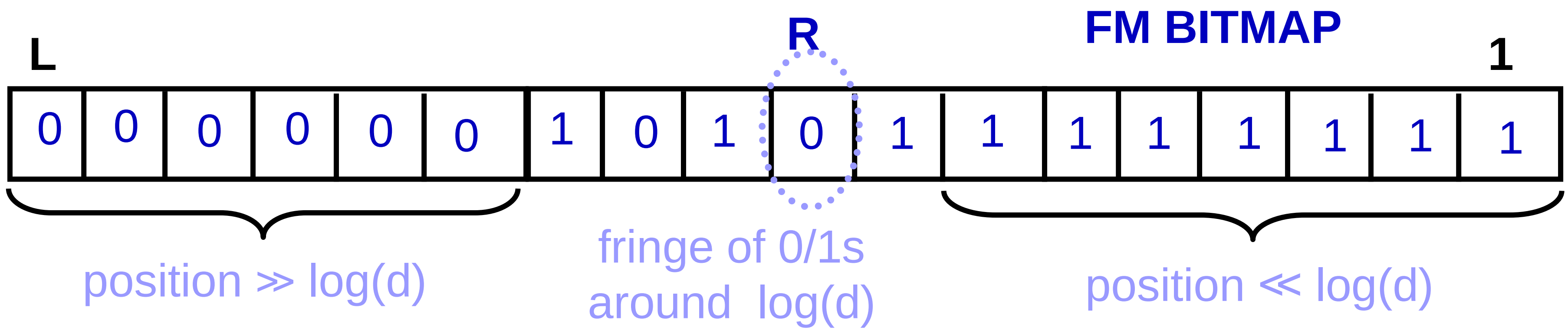
FM Sketch

- Estimates number of distinct items (F_0)
- Uses hash function mapping input items to i with prob 2^{-i}
  - i.e. Pr[h(x) = 1] = ½, Pr[h(x) = 2] = ¼, Pr[h(x)=3] = 1/8 …
  - Easy to construct h() from a uniform hash function by counting trailing zeros
- Maintain FM Sketch = bitmap array of L = log U bits
  - Initialize bitmap to all 0s
  - For each incoming value x, set FM[h(x)] = 1



FM Analysis

- If d distinct values, expect d/2 map to FM[1], d/4 to FM[2]…



  – Let R = position of rightmost zero in FM, indicator of log(d)
  – Basic estimate $d = c2^R$ for scaling constant $c \approx 1.3$
  – Average many copies (different hash fns) improves accuracy

FM Properties

- With $O(1/\varepsilon^2 \log 1/\delta)$ copies, get $(1\pm\varepsilon)$ accuracy with probability at least $1-\delta$
  - 10 copies gets ≈ 30% error, 100 copies < 10% error
  - Can pack FM into eg. 32 bits.
- Can merge together two sketches to get sketch of the union

6	5	4	3	2	1
0	0	1	0	1	1

+

6	5	4	3	2	1
0	1	1	0	0	1

=

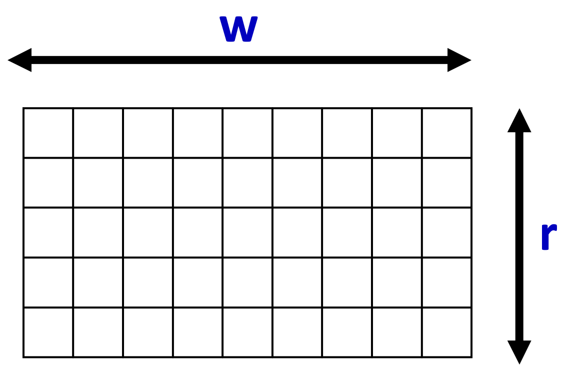
6	5	4	3	2	1
0	1	1	0	1	1

- Much subsequent work on this problem to improve bounds
  - Tighten space requirement, improve update time
  - Allow more general updates (removals of previously seen items)

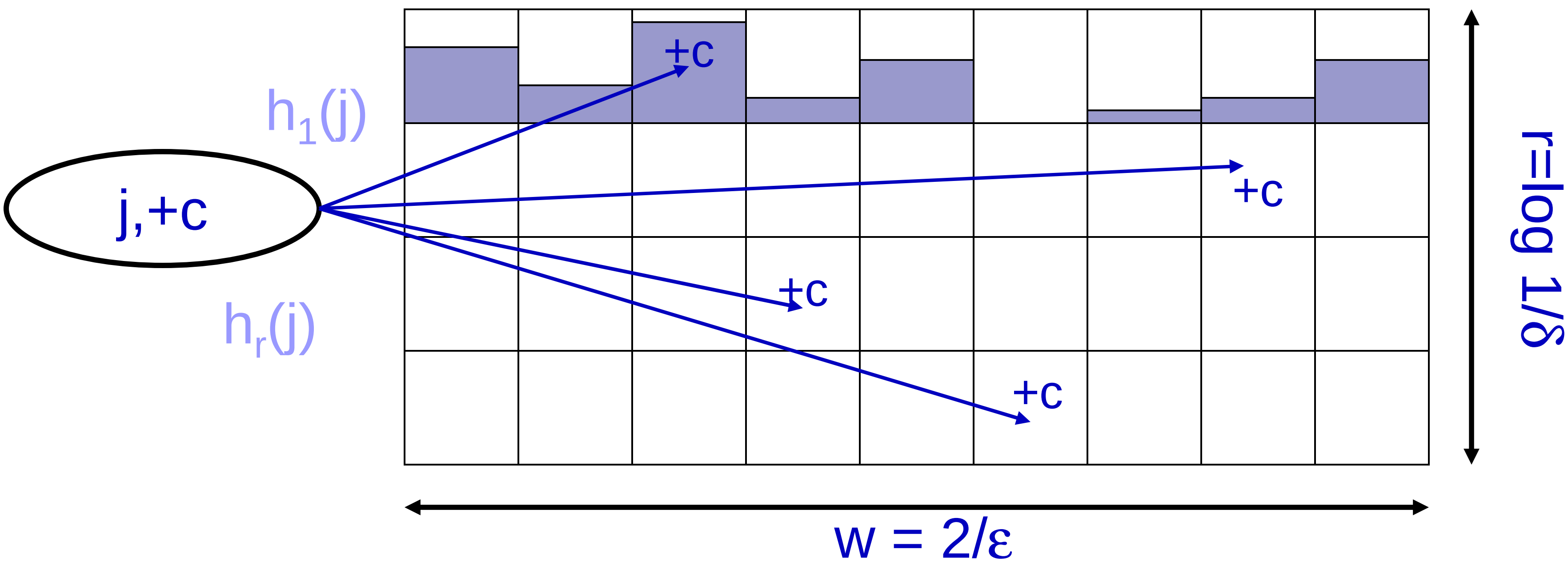
Count-Min Sketch

- Count-Min Sketch estimates node degrees [C, Muthukrishnan '04]
- Model input stream as a vector d of dimension U
  - d[v] is degree of node v
- Creates a small summary as an array of $w \times r$ in size
- Use r hash function to map vector entries to [1..w]

Array:
CM[i,j]



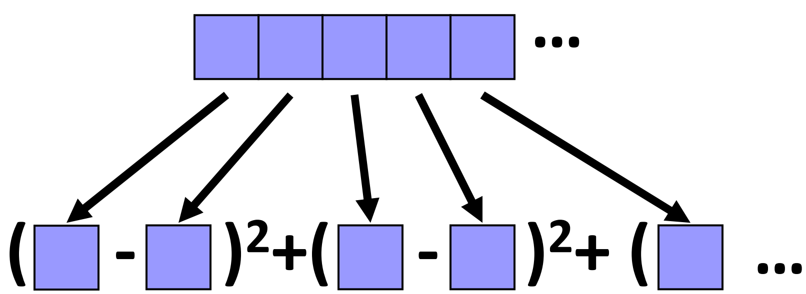
Count-Min Sketch Structure



- Each entry in vector d is mapped to one bucket per row.
- Estimate $d[j]$ by taking $\min_k CM[k,h_k(j)]$
  - Guarantees error less than εF_1 in size $O(1/\varepsilon \log 1/\delta)$
  - Probability of more error is less than $1-\delta$

F_2 approximation

- Estimate $F_2 = \text{median}_k \sum_{i=1}^{w/2} (CM[k,2i]-CM[k,2i+1])^2$
- Each row's result is $\sum_i ((-1)^{h(i)}d_i)^2 + \sum_{\lfloor h(i)/2 \rfloor = \lfloor h(j)/2 \rfloor} 2\,(-1)^{h(i)-h(j)}\, d_i\, d_j$
  - First term: $(-1)^{2h(i)} = -1^2 = 1$, and $\sum_i d_i^2 = F_2$
  - Second term: $(-1)^{h(i)-h(j)}$ is equally likely +1 or –1: expectation is 0



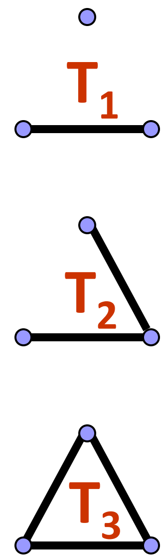
F_2 accuracy

- For $w=8/\varepsilon^2$ can show an (ε, δ) approximation
  - Expectation of each (row) estimate is F_2, variance $\leq \varepsilon^2 F_2^2$
  - Probability that each estimate is within $\pm\, \varepsilon F_2$ is constant
  - Median of $\log(1/\delta)$ estimates reduces failure probability to δ
- Result: $O(1/\varepsilon^2 \log 1/\delta)$ size sketch estimates $(1\pm\varepsilon)F_2$
- In Practice: Can be very fast, very accurate!
  - Used in Sprint 'CMON' tool

Counting Triangles via Frequency Moments

- Given (undirected) graph G, approximate number of triangles T
- Generate triple stream TS from E
  - for each e=(u,v), create (u, v, w) triple for all w
  - T_i denotes number of node triples with exactly i edges
- Consider $F_i(TS)$ over stream of triples

$$\begin{pmatrix} F_0(TS) \\ F_1(TS) \\ F_2(TS) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix}$$



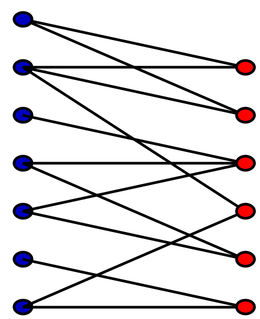
- Solve for $T = T_3 = F_0 - 3/2\ F_1 + ½\ F_2$
- $O(\varepsilon^{-2} \log 1/\delta)$ space gives $\pm\varepsilon(nm)$ with $1\text{-}\delta$ probability [BKS 02]
  - T must be large for this to be a good (relative) approximation

Outline

- Graph Streaming Models
- Hardness Results
- Degree Sequence Computations
  - Application to counting triangles
- Semi-streaming model
- Multigraph model

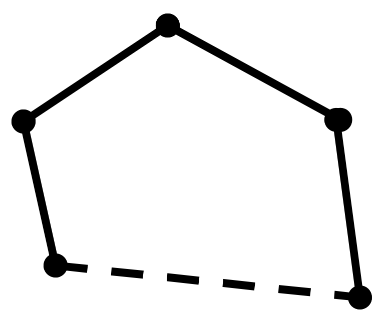
Semi-Streaming Model

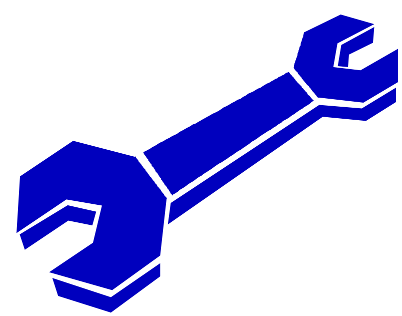
- In the semi-streaming model, we have space Õ(n) space
  - By analogy with semi-external model
- Can now easily solve connectivity, bipartiteness
  - Results on “harder” problems like matching and path length



Semi-Streaming Spanners

- Distance Estimation by constructing spanners in one pass
  - t-spanner: subgraph of G so no path stretched by a factor > t
- t-spanner with $\tilde{O}(n^{1+2/(t+1)})$ edges in $\tilde{O}(n^{1+2/(t+1)})$ space [Elkin 07]
  - Add edge e to spanner unless it completes a cycle of length $\leq$ t+1
  - A t-spanner since each edge has a path of length at most t connecting its end
  - Counting shows <t+1 cycle-free graphs have $O(n^{1+2/t+1})$ edges

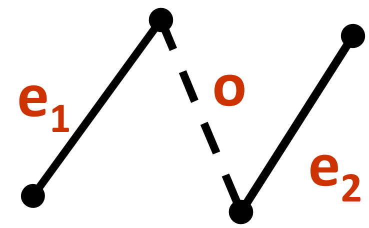




Semi-Streaming Matching

- Unweighted matching (find largest matching)
  - A maximal matching (in one pass) is 2-approx of max matching
  - $3/2 + \varepsilon$ matching with $\tilde{O}(1/\varepsilon)$ passes
- Sequence of results for edge weighted maximum matching
  - 6 approximation via combinatorial argument [FKMSZ04]
  - 5.828 via tighter argument [McGregor 05]
  - 5.24 latest claimed bound [DasSarma, Lipton, Nanongkai 09]

Semi-Streaming Weighted Matching



- For each edge $e = (v, w)$ build a matching M:
  - Consider edges in M incident on e, C
  - If $wt(e) > 2wt(C)$, then $M \leftarrow M \cup \{e\} / C$, $M' \leftarrow M' \cup C$
- For edge o in optimal solution OPT and $e \in (M \cup M')$
  - Charge $wt(o)$ to e if $o=e$ or e alone prevented o being picked
  - Split $wt(o)$ between e_1 and e_2 if both prevent o being picked
  - If o charged to e, then $wt(o) \leq 2wt(e)$ by defn of alg.
- At end, each edge charged against is either:
  - In M and charged to by at most two o in OPT
  - In M' and charged to by at most one o in OPT
  - $wt(OPT) \leq 2(wt(M') + 2wt(M)) \leq 2(3wt(M)) = 6wt(M)$ ❑

Semi-Streaming Model

- Meta-question: when is semi-streaming model applicable?
  - Social networks: average degree is < 100 (so $m = O(n)$)
  - Diameters are small – so constant stretch does not help!
  - Algorithms can assume each edge is seen exactly once
    - Reasonable for web exploration (incidence order)
    - Questionable for IP, call graph monitoring

Outline

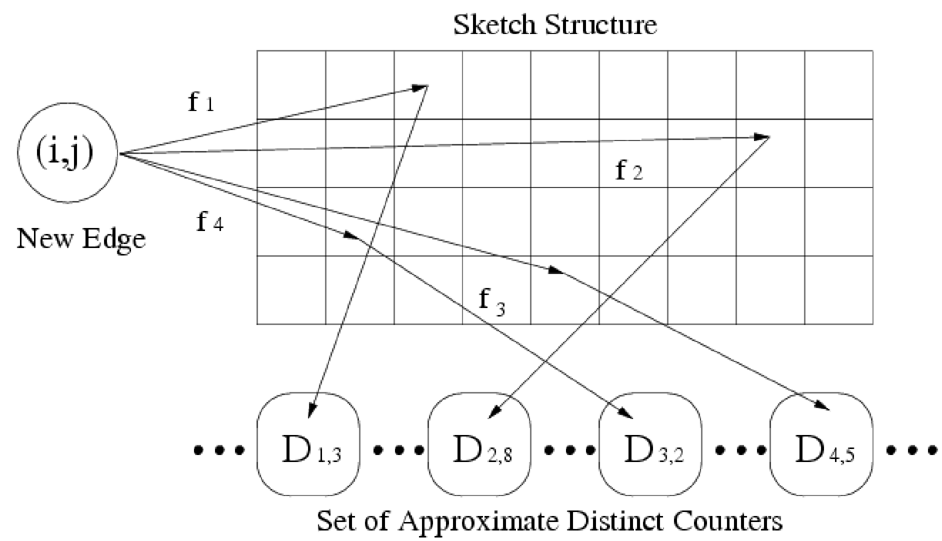
- Graph Streaming Models
- Hardness Results
- Degree Sequence Computations
  - Application to counting triangles
- Semi-streaming model
- Multigraph model

Multigraph Model

- Each edge seen many times, but only counts once
  - E.g. observing communications, want to study support graph
- Some algorithms seen earlier robust to repeated edges
  - Spanner construction, matching: will make same decisions
  - F_0 is by definition robust to repetition
- Others not robust
  - Will inflate degree sequence computations
  - Any random sampling algorithm will get confused
- Need to use "duplicate insensitive" methods, such as F_0

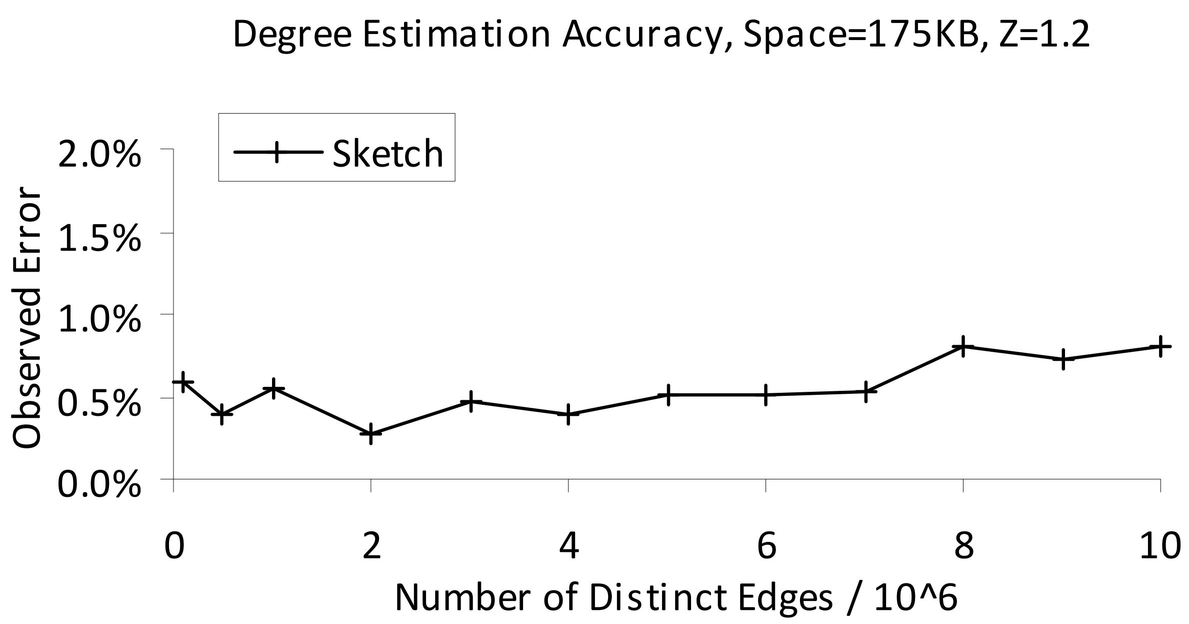
Distinct Frequent Items

- Given v, estimate $d_v = |\{ w : (v, w) \in E\}|$
  - Large d_v indicates unusual net activity (port scans, worms)
- Take existing frequent items algs and put in F_0 sketch
  - Care needed: if algorithm subtracts two estimated counts, accuracy is not preserved, as $(1 \pm\varepsilon)X - (1\pm \varepsilon)Y \neq (1\pm \varepsilon)(X-Y)$
- Count-Min sketch only uses additions, so can apply:



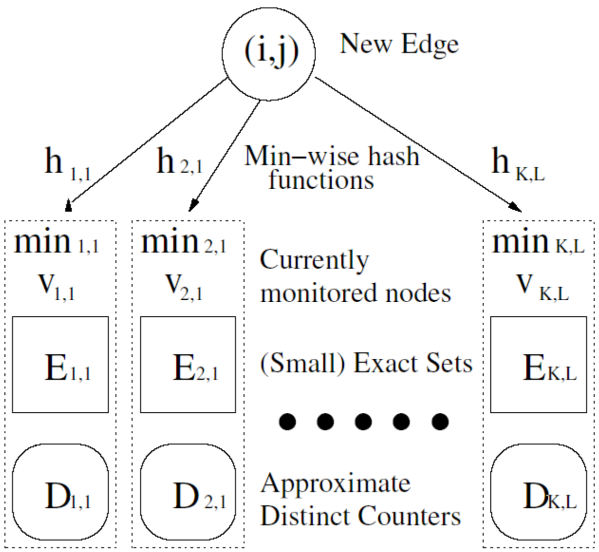
Result

- Can prove estimate has only small bias in expectation
- Estimate any d_v given v with error εF_0 in space $O(\varepsilon^{-3} \log^2 n)$.
- Time per update is $O(\log^2 n)$.



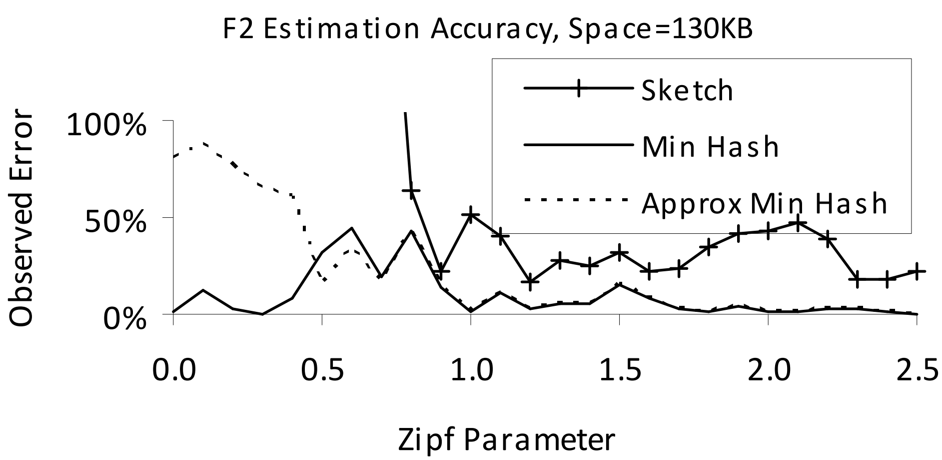
Distinct F_2 estimation

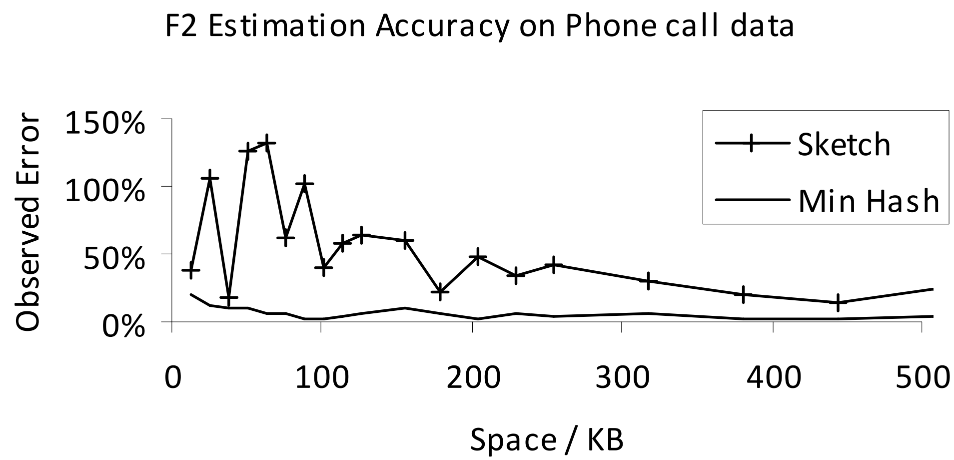
- CM sketch approach of subtracting counts no longer works
- Use a min-wise hashing technique
- Sample almost uniformly from E
  - For each edge in stream, compute h((i,j))
  - Store info on v if h((v,j)) is smallest so far
  - Collect all edges (v,j) matching v till $>1/\varepsilon^2$, then *approx count* (F_0) these edges
  - Estimate of F_2 is m * (2d –1), d = number of edges seen matching on v



Experimental Study

- Expectation of estimate = $(1+\varepsilon)F_2$
- Variance = $(1+\varepsilon)n^{1/2}F_2^{\ 2}$
- Repeat enough times to increase accuracy.
- Space = $O(\varepsilon^{-4}\ n^{1/2} \log n)$





Concluding Remarks

- Graph streaming yields challenging models of computation
  - Many natural graph questions are “hard” in these models
- Some global properties can be computed very compactly
  - Degree sequence computation, high degree nodes
- Even in semi-streaming, local properties are only *approximated*
  - E.g. distance between nodes stretches by constant factors
- Some results take multiple passes (e.g. PageRank) – realistic?

- Next challenge: assume some small-world scale-free model and prove stronger algorithmic results for graph streaming?