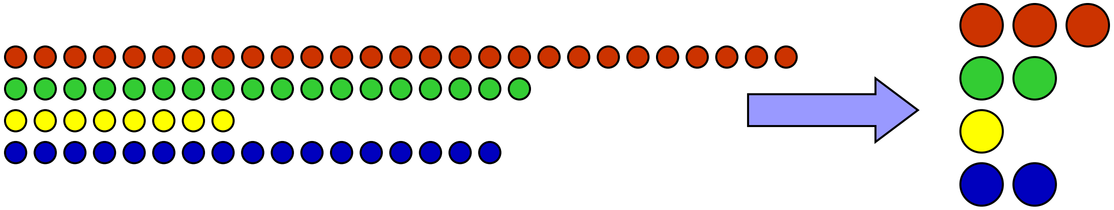
Introduction to Distributed Data Streams

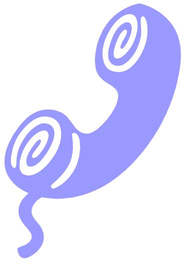


Graham Cormode

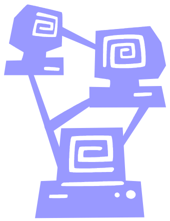
graham@research.att.com

Data is Massive

- Data is growing faster than our ability to store or index it
- There are 3 Billion Telephone Calls in US each day (100BN minutes), 30B emails daily, 4B SMS, IMs.



- Scientific data: NASA's observation satellites generate billions of readings each per day.
- IP Network Traffic: can be billions packets per hour per router. Each ISP has many (hundreds) routers!



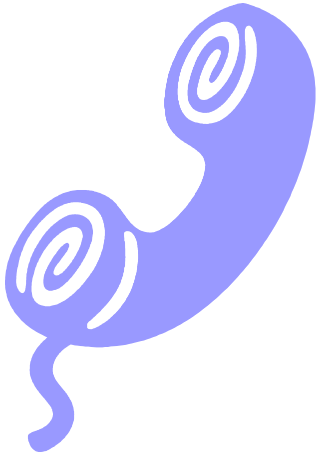
- Whole genome sequences for individual humans now available: each is gigabytes in size

Massive Data Analysis

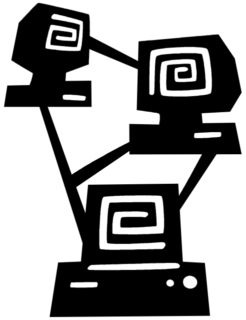
Must analyze this massive data:

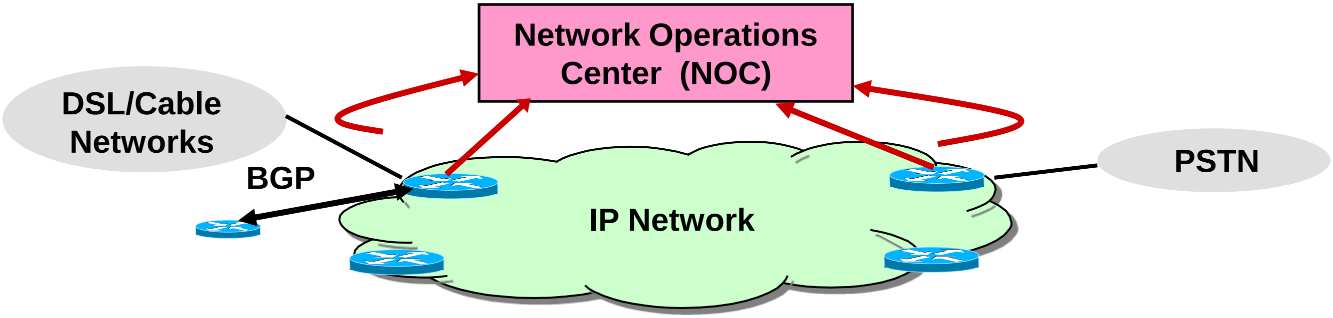
- Scientific research (monitor environment, species)
- System management (spot faults, drops, failures)
- Customer research (association rules, new offers)
- For revenue protection (phone fraud, service abuse)

Else, why even measure this data?



Example: Network Data



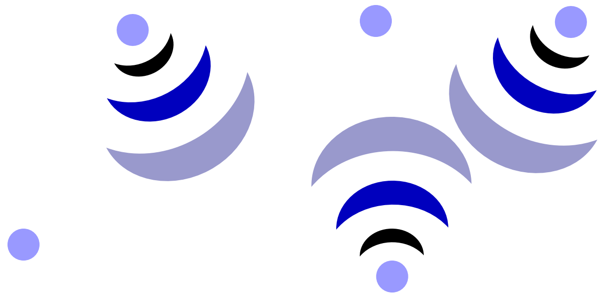


- Networks are sources of massive data: the metadata per hour per router is gigabytes
- Fundamental problem of data stream analysis: Too much information to store or transmit
- So process data as it passes each network device: one pass, small space—the *data stream* approach
- Approximate answers to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

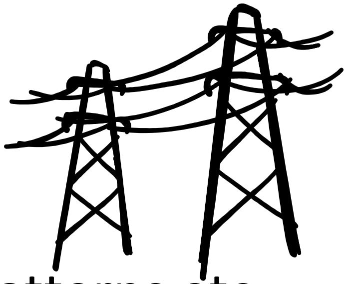
Streaming Data Questions

- Network managers ask questions requiring us to analyze and mine the data:
  - Find hosts with similar usage patterns (clusters)?
  - Which destinations or groups use most bandwidth?
  - Was there a change in traffic distribution overnight?
  - Build predictive models for future behavior?
- Complexity comes from scale of the distributed data
- Will introduce solutions for these and other problems

Other Streaming Applications



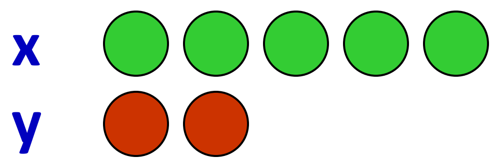
- Wireless monitors
  - Monitor habitat and environmental parameters
  - Track many objects, intrusions, trend analysis…



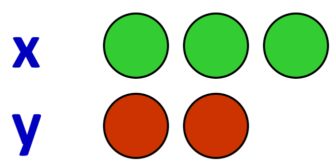
- Utility Companies
  - Monitor power grid, customer usage patterns etc.
  - Alerts and rapid response in case of problems

Data Stream Models

- We model data streams as sequences of simple tuples
- Problems hard due to scale and dimension of many streams
- Arrivals only streams:
  - Example: (x, 3), (y, 2), (x, 2) encodes the arrival of 3 copies of item x, 2 copies of y, then 2 copies of x.

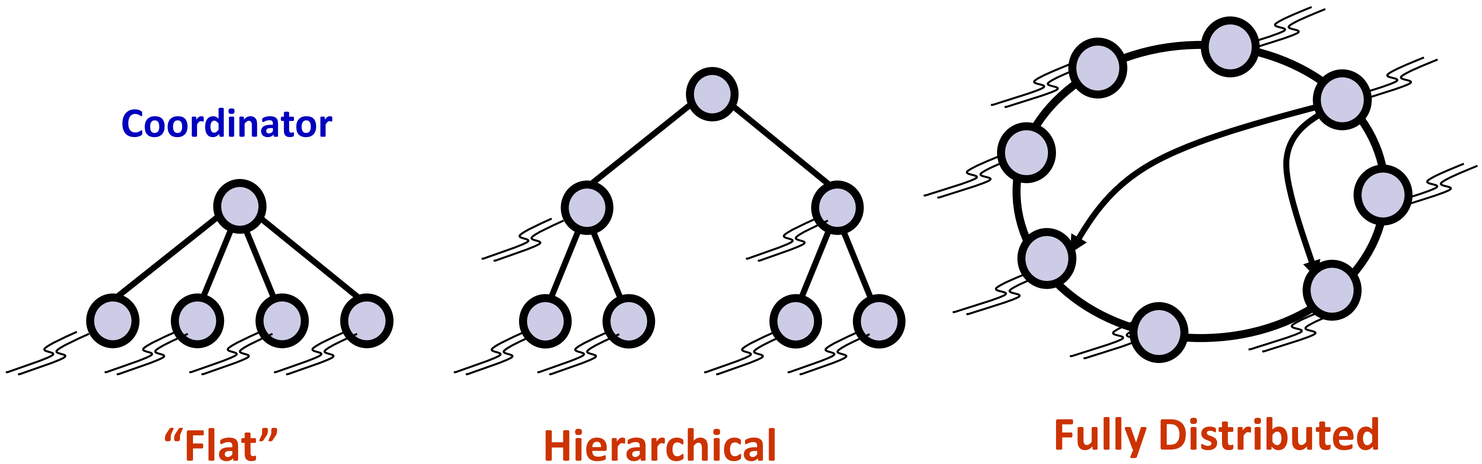


  - Could represent eg. packets on a network; power usage
- Arrivals and departures:
  - Example: (x, 3), (y,2), (x, -2) encodes final state of (x, 1), (y, 2).



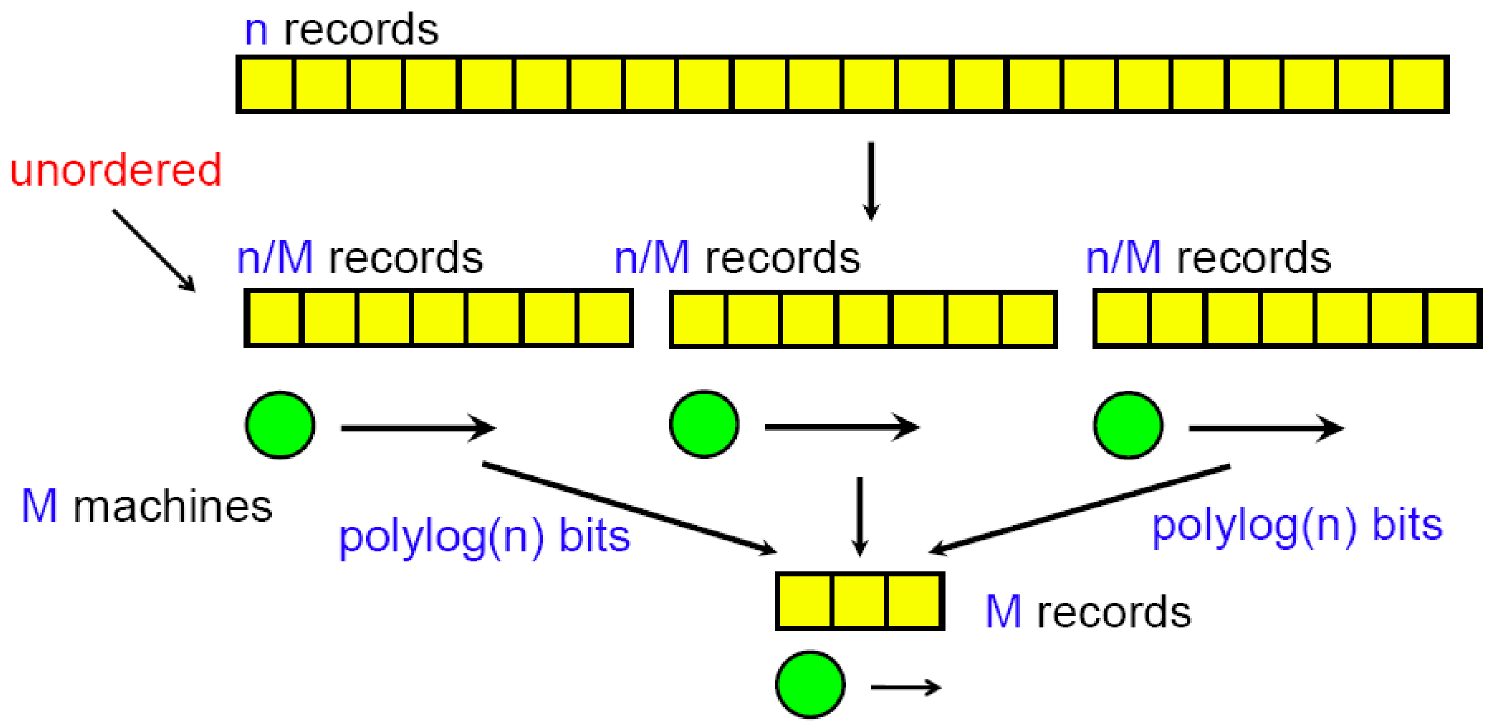
  - Can represent fluctuating quantities, or measure differences between two distributions

Models of Distribution



- One-shot computation [Gibbons, Tirthapura 01, Feldman et al. '08]
  - Build compact summaries of data that can be combined
- Gossip-based communication [Kempe, Dobra, Gehrke 03]
  - Opportunistically swap info with neighbors until convergence
- Continuous computation [C, 2011]
  - Track a (complex) function of distributed values

MUD Model



- Massive Unordered Data (MUD) model [Feldman et al. '08]
  - Special case of MapReduce/Hadoop processing
  - Theorem: Can simulate any deterministic streaming algs
  - Parameters: space used by each machine, message size, time

Approximation and Randomization

- Many things are hard to compute exactly over a stream
  - Is the count of all items the same in two different streams?
  - Requires linear space to compute exactly
- Approximation: find an answer correct within some factor
  - Find an answer that is within 10% of correct result
  - More generally, a $(1 \pm \varepsilon)$ factor approximation
- Randomization: allow a small probability of failure
  - Answer is correct, except with probability 1 in 10,000
  - More generally, success probability $(1-\delta)$
- Approximation **and** Randomization: (ε, δ)-approximations

First examples of streaming algorithms

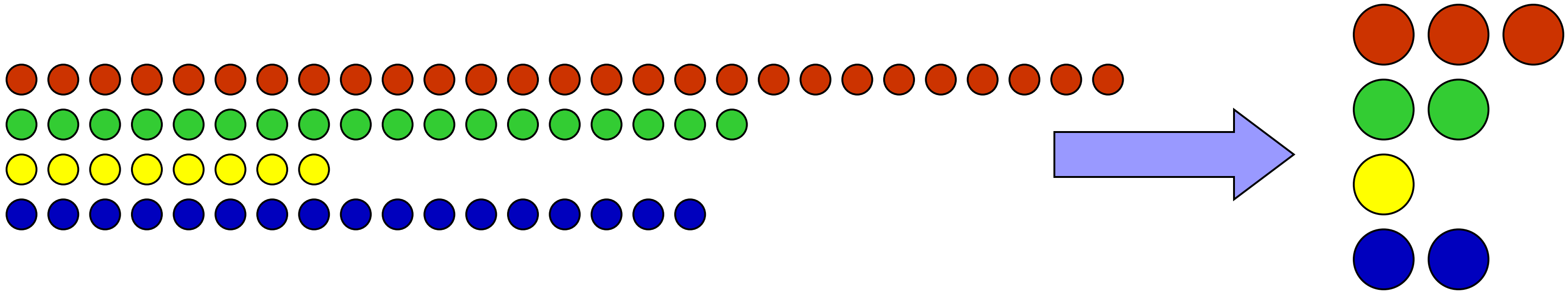
- How to draw a random sample?
- How to estimate the entropy of a distribution?
- How to efficiently find heavy hitters?
- How to test if two distributed streams are equal?
- How to compactly represent a set?

…and do all of this over distributed data?

Small Summaries

- A summary (approximately) allows answering such questions
- To earn the name, should be (very) small!
  - Can keep in fast storage
- Should be able to build, update and query efficiently
- Key methods for summaries:
  - Create an empty summary
  - Update with one new tuple: streaming processing
  - Merge summaries together: distributed processing
  - Query: may tolerate some approximation

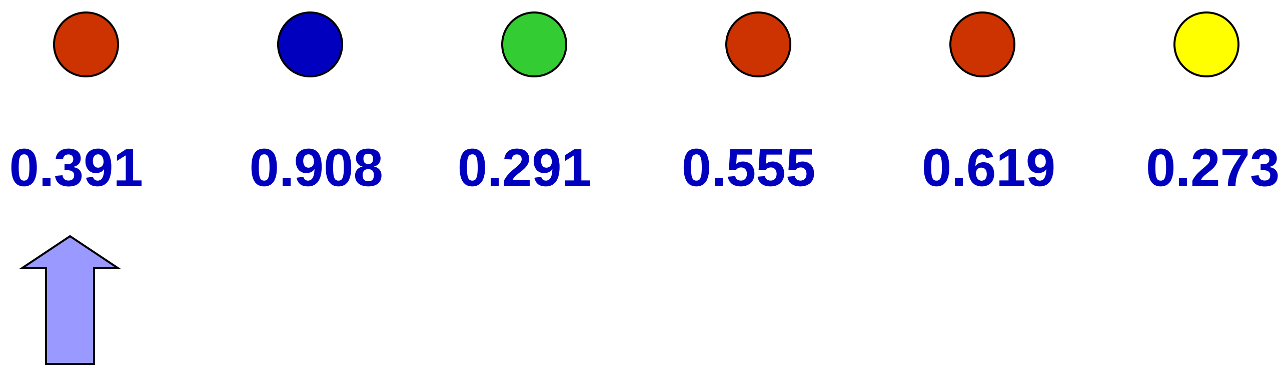
Sampling From a Data Stream



- Fundamental prob: sample m items uniformly from data
  – Useful: approximate costly computation on small sample
- Challenge: don't know how large total input is
  – So when/how often to sample?
- Several solutions, apply to different situations:
  – Reservoir sampling (dates from 1980s?)
  – Min-wise sampling (dates from 1990s?)

Min-wise Sampling

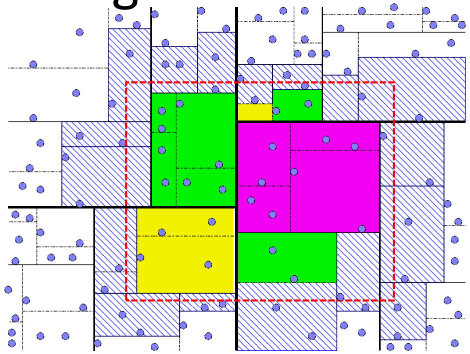
- For each item, pick a random fraction between 0 and 1
- Store item(s) with the smallest random tag [Nath et al.'04]



- Each item has same chance of least tag, so uniform
- Can run on multiple streams separately, then merge

Advanced Sampling

- Sampling widely used in practice: simple semantics
  - Can often apply exponential (Chernoff) error bounds
  - Sample of $O(1/\varepsilon^2)$ items gives ε additive error on predicate queries
- Much active research around sampling:
  - Sampling from unaggregated, weighted data?
  - Getting the most value from your sample
  - Sampling according to functions of weights
  - Sampling over distributed data with negative weights



Application of Sampling: Entropy

- Given a long sequence of characters

  $$S = \langle a_1, a_2, a_3 \ldots a_m \rangle \qquad \text{each } a_j \in \{1 \ldots n\}$$

- Let f_i = frequency of i in the sequence
- Compute the empirical entropy:

  $$H(S) = - \sum_i f_i/m \log f_i/m = - \sum_i p_i \log p_i$$

- Example: $S = \langle$ **a, b, a, b, c, a, d, a** $\rangle$
  - $p_a = 1/2$, $p_b = 1/4$, $p_c = 1/8$, $p_d = 1/8$
  - $H(S) = ½ + ¼ \times 2 + 1/8 \times 3 + 1/8 \times 3 = 7/4$
- Entropy promoted for anomaly detection in networks

Sampling Based Algorithm

- Simple estimator (the "AMS estimator"):
  - Randomly (min-wise) sample a position j in a single stream
  - Count how many times a_j appears subsequently = r
  - Output $X = -(r \log (r/m) – (r-1) \log(r-1)/m)$

- Claim: Estimator is unbiased – $E[X] = H(S)$
  - Proof: prob of picking $j = 1/m$, sum telescopes correctly
- Variance is not too large – $Var[X] = O(\log^2 m)$
  - Can be proven by bounding $|X| \leq \log m$

Analysis of Basic Estimator

- A general technique in data streams:
  - Repeat in parallel an unbiased estimator with bounded variance, take average of estimates to improve result
  - Use concentration bounds (next lecture) to guarantee accuracy
  - Number of repetitions depends on ratio $Var[X]/E^2[X]$
  - For entropy, this means space $O(\log^2 m / H^2(S))$

- Problem for entropy: when $H(S)$ is very small?
  - Space needed for an accurate approx goes as $1/H^2$!

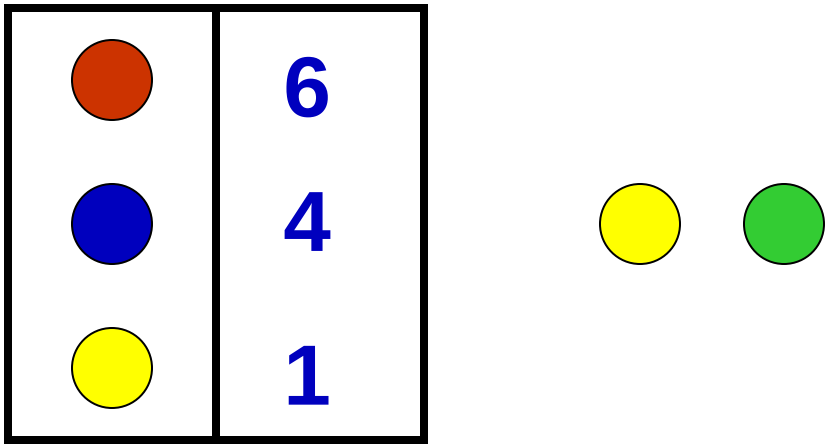
Outline of Improved Algorithm

- Observation: only way to get $H(S) = o(1)$ is to have only one character with p_i close to 1
  - **aaaaaaaaaaaaaaaaaaaaaaaaaaaaaabaaaaa**
- If we can identify this character, and make an estimator on stream without this token, can estimate $H(S)$
- How to identify and remove all in one pass?
- Keep a "back up" sample from the stream with a_j removed
  - Use the back-up sample to estimate H if a_j is most frequent item
- Full details and analysis in [Chakrabarti, C, McGregor 07]
  - Total space is $O(\varepsilon^{-2} \log m \log 1/\delta)$ for (ε,δ) approx

Distributing entropy computation

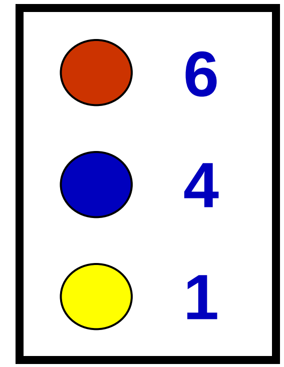
- Entropy summary allows update but not merge
  - How to set counts when summaries sample different items?
- Can simulate the above algorithm across distributed streams:
  - All observers keep in sync over what are current sampled items
  - Keep local counts of occurrences, can sum when required
- Adapt existing analysis to bound communication cost
  - Show that number of times sample changes is O(log n)
- Need new approach for more general models:
  - If we want to build a self-contained summary for entropy…
  - If we want to continuously monitor entropy…

Misra-Gries Summary (1982)



- Misra-Gries (MG) algorithm finds up to k items that occur more than 1/k fraction of the time in the input
- Update: Keep k different candidates in hand. For each item:
  - If item is monitored, increase its counter
  - Else, if < k items monitored, add new item with count 1
  - Else, decrease all counts by 1

Streaming MG analysis



- N = total weight of input
- M = sum of counters in data structure
- Error in any estimated count at most (N-M)/(k+1)
  - Estimated count a lower bound on true count
  - Each decrement spread over (k+1) items: 1 new one and k in MG
  - Equivalent to deleting (k+1) distinct items from stream
  - At most (N-M)/(k+1) decrement operations
  - Hence, can have “deleted” (N-M)/(k+1) copies of any item
  - So estimated counts have at most this much error

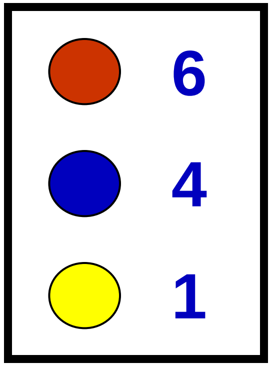
Merging two MG Summaries [ACHPWY '12]

- Merge algorithm:
  - Merge the counter sets in the obvious way
  - Take the $(k+1)$th largest counter $= C_{k+1}$, and subtract from all
  - Delete non-positive counters
  - Sum of remaining counters is M_{12}
- This keeps the same guarantee as Update:
  - Merge subtracts at least $(k+1)C_{k+1}$ from counter sums
  - So $(k+1)C_{k+1} \leq (M_1 + M_2 - M_{12})$
  - By induction, error is
    $((N_1-M_1) + (N_2-M_2) + (M_1+M_2-M_{12}))/(k+1)=((N_1+N_2) - M_{12})/(k+1)$

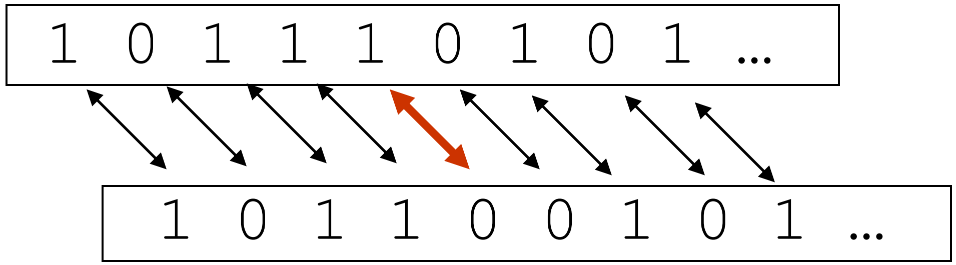
    (prior error) (from merge) (as claimed)

A Powerful Summary

- MG summary with update and merge is very powerful
  - Builds a compact summary of the frequency distribution
  - Can also multiply the summary by any scalar
  - Hence can take (positive) linear combinations: $\alpha x + \beta y$
  - Useful for building models of data
- Will later see sketches that allow arbitrary linear combinations



Fingerprints



- Test if two (distributed) binary streams are equal

  $d_=(x,y) = 0$ iff $x=y$, 1 otherwise

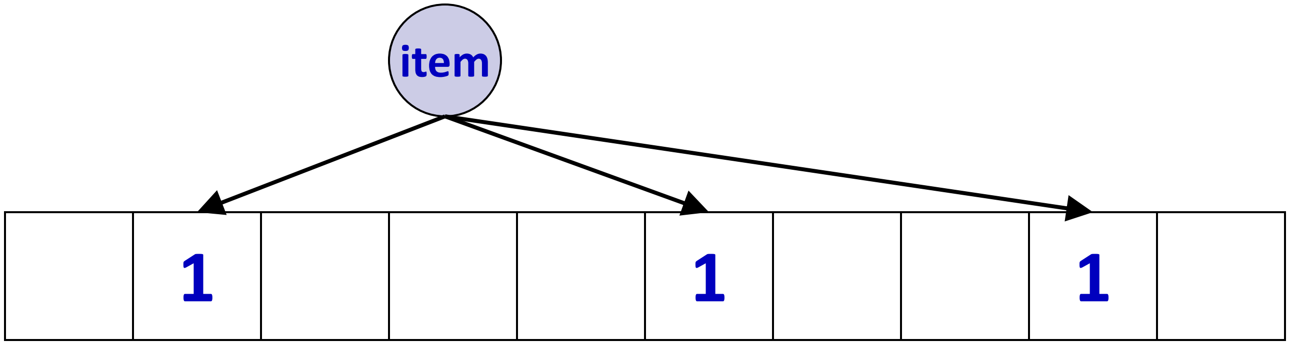
- To test in small space: pick a suitable hash function h
- Test $h(x)=h(y)$: small chance of false positive, no chance of false negative
- Compute $h(x)$, $h(y)$ incrementally as new bits arrive
  - How to choose the function $h()$?

Polynomial Fingerprints

- Pick $h(x) = \sum_{i=1}^{n} x_i r^i \bmod p$ for prime p, random $r \in \{1 \ldots p-1\}$
- Why?
- Flexible: $h(x)$ is linear function of x—easy to update and merge
- For accuracy, note that computation $\bmod p$ is over the field Z_p
  - Consider the polynomial in α, $\sum_{i=1}^{n} (x_i - y_i) \alpha^i = 0$
  - Polynomial of degree n over Z_p has at most n roots
- Probability that r happens to solve this polynomial is n/p
- So $\Pr[\, h(x) = h(y) \mid x \neq y \,] \leq n/p$
  - Pick $p = \text{poly}(n)$, fingerprints are $\log p = O(\log n)$ bits
- Fingerprints applied to small subsets of data to test equality
  - Will see several examples that use fingerprints as subroutine

Bloom Filters

- Bloom filters compactly encode set membership
  - k hash functions map items to bit vector k times
  - Set all k entries to **1** to indicate item is present
  - Can lookup items, store set of size n in O(n) bits



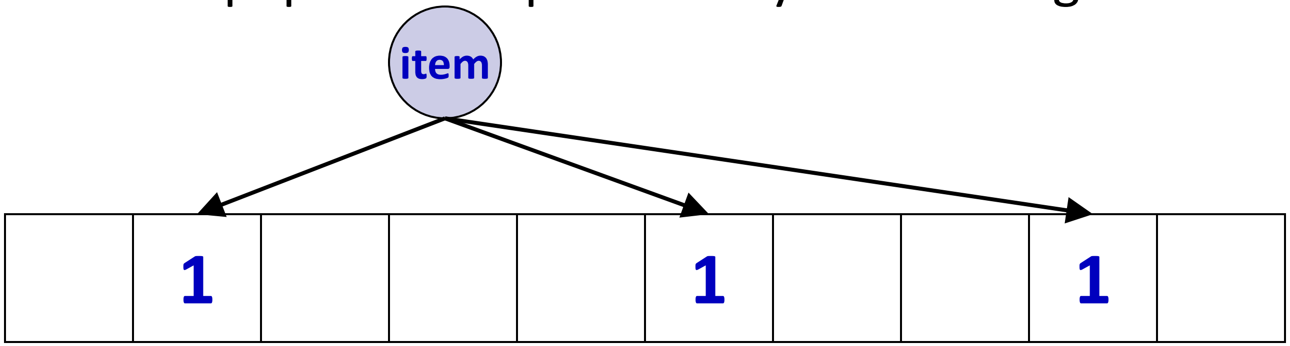
- Duplicate insertions do not change Bloom filters
- Can be merge by OR-ing vectors (of same size)

Bloom Filter analysis

- How to set k (number of hash functions), m (size of filter)?
- False positive: when all k locations for an item are set
  - If ρ fraction of cells are empty, false positive probability is $(1-\rho)^k$
- Consider probability of any cell being empty:
  - For n items, $\Pr[\text{ cell } j \text{ is empty }] = (1 - 1/m)^{kn} \approx \rho \approx \exp(-kn/m)$
  - False positive prob $= (1 - \rho)^k = \exp(k \ln(1 - \rho))$
    $= \exp(-m/n \ln(\rho) \ln(1-\rho))$
- For fixed n, m, by symmetry minimized at $\rho = ½$
  - Half cells are occupied, half are empty
  - Give $k = (m/n)\ln 2$, false positive rate is $½^k$
  - Choose $m = cn$ to get constant FP rate, e.g. $c=10$ gives < 1% FP

Bloom Filters Applications

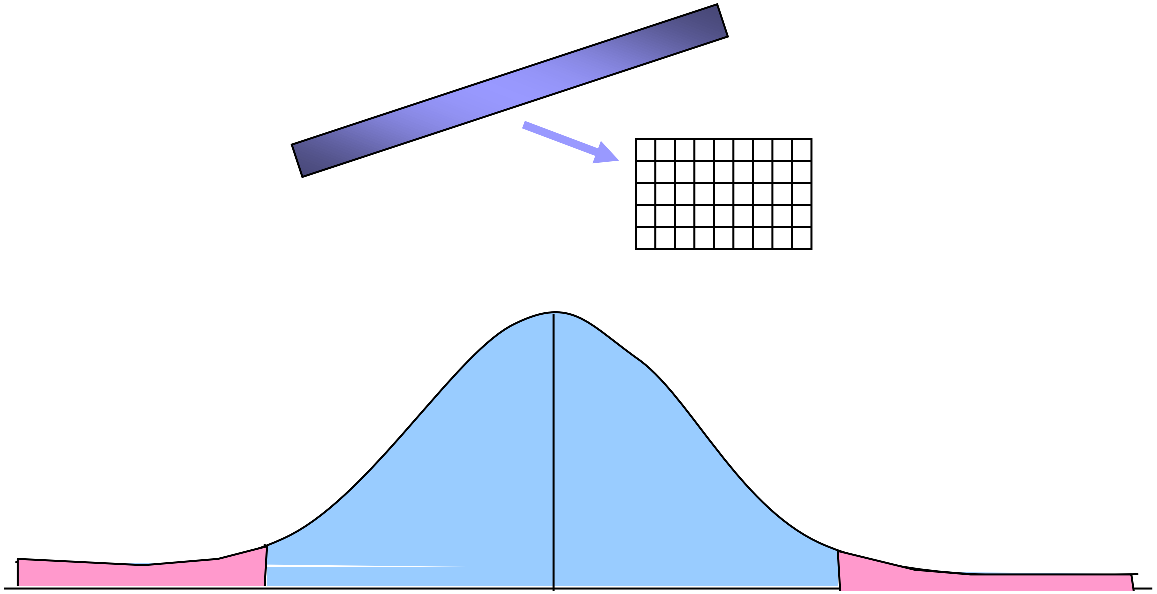
- Bloom Filters widely used in “big data” applications
  - Many problems require storing a large set of items
- Can generalize to allow deletions
  - Swap bits for counters: increment on insert, decrement on delete
  - If representing sets, small counters suffice: 4 bits per counter
  - If representing multisets, obtain sketches (next lecture)
- Bloom Filters are an active research area
  - Several papers on topic in every networking conference…



Coming Soon

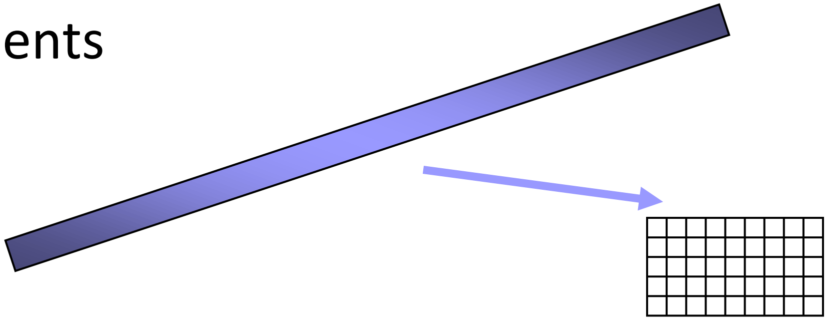
- Lecture 2: Sketches and Concentration Bounds
  - Randomized sketch data structures (Count-Min, AMS, F0)
  - Proved using Markov, Chebyshev, Chernoff inequalities
- Lecture 3: L_p sampling & Streaming verification
  - L_p sampling, with application to graph computations
  - Verifiable stream computations with interactive proofs
- Lecture 4: Distributed Continuous Monitoring & Lower Bounds
  - The distributed continuous model and algorithms
  - Lower bounds: what can't we do efficiently?

Sketch Data Structures and Concentration Bounds



Frequency Moments

- Intro to frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments



Frequency Distributions

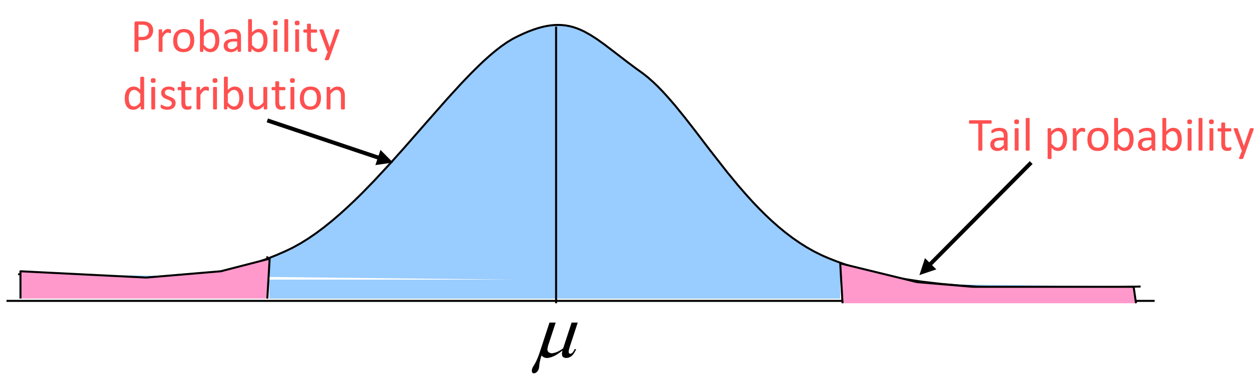
- Given set of items, let f_i be the number of occurrences of item i
- Many natural questions on f_i values:
  - Find those i's with large f_i values (heavy hitters)
  - Find the number of non-zero f_i values (count distinct)
  - Compute $F_k = \sum_i (f_i)^k$ – the k'th Frequency Moment
  - Compute $H = \sum_i (f_i/F_1) \log (F_1/f_i)$ – the (empirical) entropy
- "Space Complexity of the Frequency Moments"
  Alon, Matias, Szegedy in STOC 1996
  - Awarded Gödel prize in 2005
  - Set the pattern for many streaming algorithms to follow

Concentration Bounds

- Will provide randomized algorithms for these problems
- Each algorithm gives a (randomized) estimate of the answer
- Give confidence bounds on the final estimate X
  - Use probabilistic concentration bounds on random variables
- A concentration bound is typically of the form

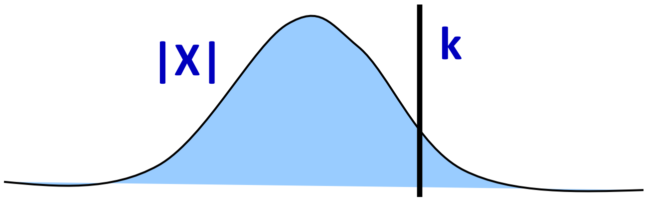
$$\Pr[\ |X - x| > \varepsilon y\ ] < \delta$$

  - At most probability δ of being more than εy away from x



Markov Inequality

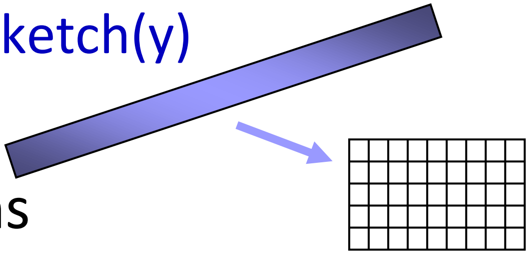
- Take *any* probability distribution X s.t. $\Pr[X < 0] = 0$
- Consider the event $X \geq k$ for some constant $k > 0$
- For any draw of X, $kI(X \geq k) \leq X$
  - Either $0 \leq X < k$, so $I(X \geq k) = 0$
  - Or $X \geq k$, lhs = k



- Take expectations of both sides: $k \Pr[X \geq k] \leq E[X]$
- Markov inequality: $\Pr[X \geq k] \leq E[X]/k$
  - Prob of random variable exceeding k times its expectation $< 1/k$
  - Relatively weak in this form, but still useful

Sketch Structures

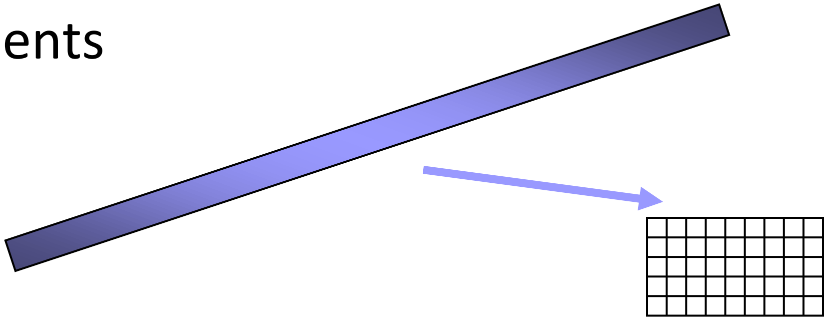
- Sketch is a class of summary that is a linear transform of input
  - Sketch(x) = Sx for some matrix S
  - Hence, Sketch(αx + βy) = α Sketch(x) + β Sketch(y)
  - Trivial to update and merge



- Often describe S in terms of hash functions
  - If hash functions are simple, sketch is fast
- Aim for limited independence hash functions h: $[n] \rightarrow [m]$
  - If $\Pr_{h\in H}[\ h(i_1)=j_1 \wedge h(i_2)=j_2 \wedge \ldots\ h(i_k)=j_k\ ] = m^{-k}$, then H is k-wise independent family ("h is k-wise independent")
  - k-wise independent hash functions take time, space O(k)

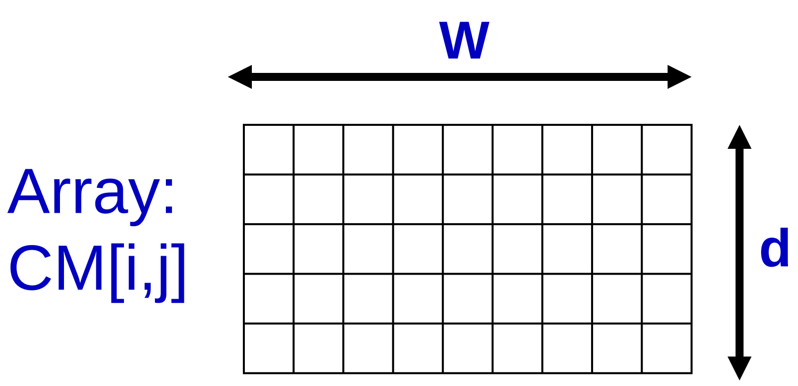
Frequency Moments

- Intro to frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments

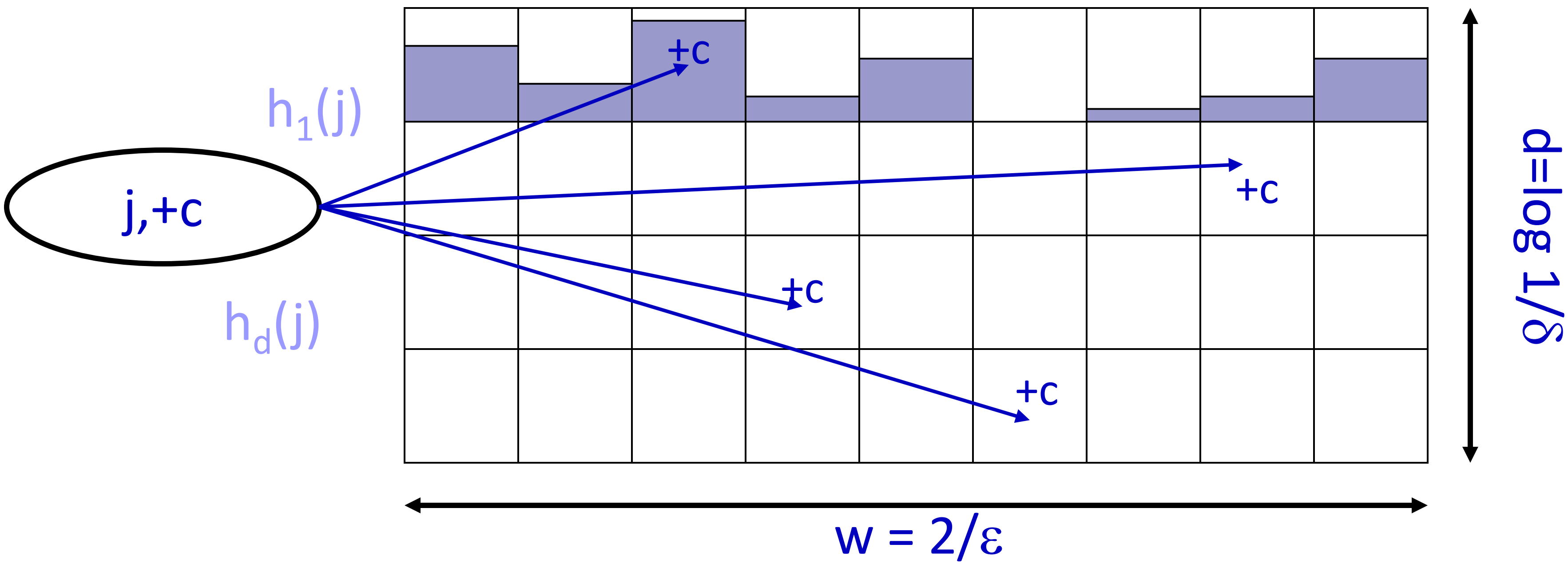


Count-Min Sketch

- Simple sketch idea relies primarily on Markov inequality
- Model input data as a vector x of dimension U
- Creates a small summary as an array of $w \times d$ in size
- Use d hash function to map vector entries to [1..w]
- Works on arrivals only and arrivals & departures streams



Count-Min Sketch Structure



- Each entry in vector x is mapped to one bucket per row.
- Merge two sketches by entry-wise summation
- Estimate $x[j]$ 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 $1-\delta$

[C, Muthukrishnan '04]

Approximation of Point Queries

Approximate point query $x'[j] = \min_k CM[k,h_k(j)]$

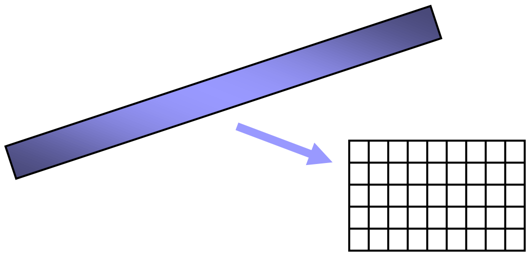
- Analysis: In k'th row, $CM[k,h_k(j)] = x[j] + X_{k,j}$
  - $X_{k,j} = \Sigma_i\, x[i]\, I(h_k(i) = h_k(j))$
  - $E[X_{k,j}] \quad = \Sigma_{i \neq j}\, x[i]*\Pr[h_k(i)=h_k(j)]$
    $\leq \Pr[h_k(i)=h_k(j)] * \Sigma_i\, x[i]$
    $= \varepsilon\, F_1/2$ – requires only pairwise independence of h
  - $\Pr[X_{k,j} \geq \varepsilon F_1] = \Pr[\, X_{k,j} \geq 2E[X_{k,j}]\,] \leq 1/2$ by Markov inequality
- So, $\Pr[x'[j] \geq x[j] + \varepsilon F_1] = \Pr[\forall\, k.\ X_{k,j} > \varepsilon F_1] \leq 1/2^{\log 1/\delta} = \delta$
- Final result: with certainty $x[j] \leq x'[j]$ and
  with probability at least $1-\delta$, $x'[j] < x[j] + \varepsilon F_1$

Applications of Count-Min to Heavy Hitters

- Count-Min sketch lets us estimate f_i for any i (up to εF_1)
- Heavy Hitters asks to find i such that f_i is large ($> \phi F_1$)
- Slow way: test every i after creating sketch
- Faster way: test every i after it is seen in the stream, and remember $1/\phi$ largest estimated values
- Alternate way:
  - Keep binary tree over input domain: each node is a subset
  - Keep sketches of all nodes at same level
  - Descend tree to find large frequencies, discard 'light' branches
  - Same structure estimates arbitrary range sums

Application to Large Scale Machine Learning

- In machine learning, often have very large feature space
  - Many objects, each with huge, sparse feature vectors
  - Slow and costly to work in the full feature space
- "Hash kernels": work with a sketch of the features
  - Works very well in practice!
- Similar analysis explains *why:*
  - Essentially, not too much noise on the important features

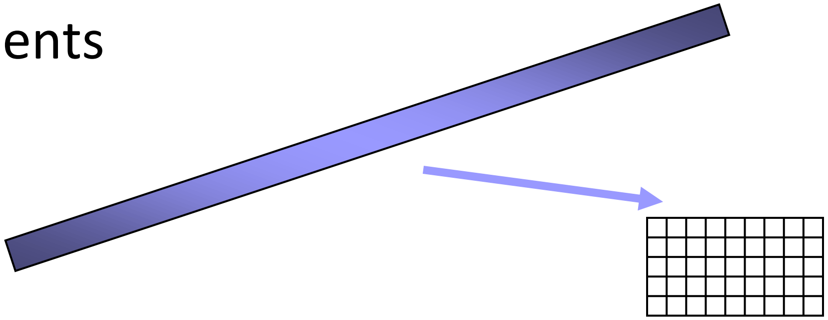


Count-Min Exercises

1. The median of a distribution is the item so that the sum of the frequencies of lexicographically smaller items is ½ F_1. Use CM sketch to find the (approximate) median.
2. Assume the input frequencies follow the Zipf distribution so that the i'th largest frequency is $\theta(i^{-z})$ for $z>1$. Show that CM sketch only needs to be size $\varepsilon^{-1/z}$ to give same guarantee
3. Suppose we have data where frequencies of items are allowed to be negative. Extend CM sketch analysis to estimate these frequencies (note, Markov argument no longer works directly)
4. How to efficiently find the large absolute frequencies when some are negative? Or in the difference of two streams?

Frequency Moments

- Intro to frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments

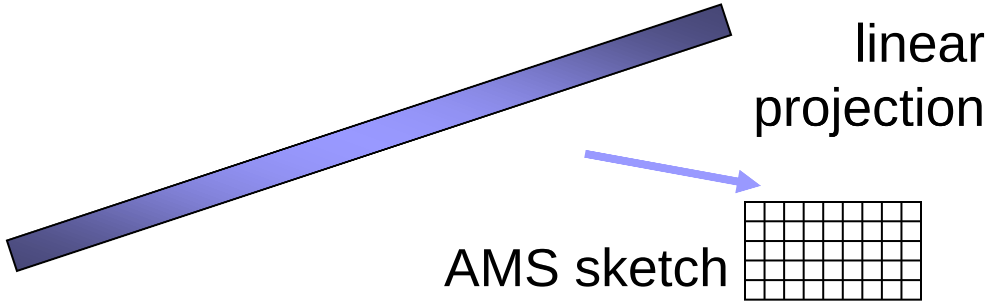


Chebyshev Inequality

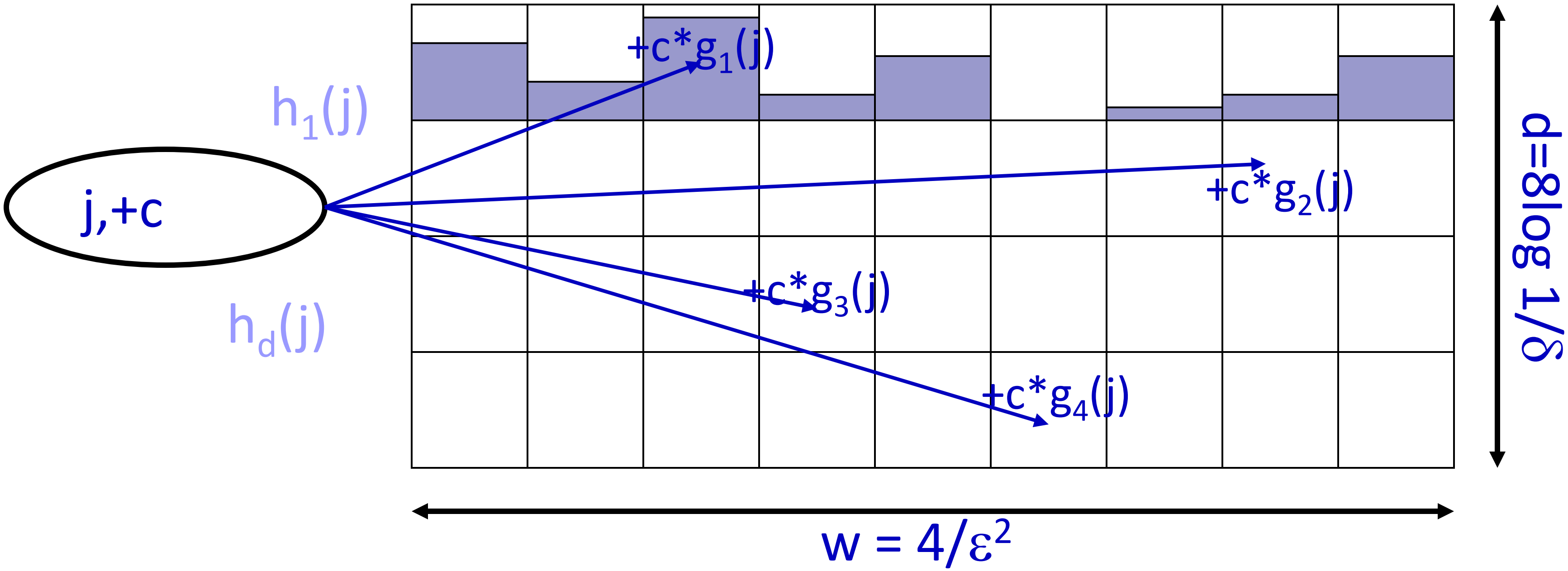
- Markov inequality is often quite weak
- But Markov inequality holds for any random variable
- Can apply to a random variable that is a function of X
- Set $Y = (X - E[X])^2$
- By Markov, $\Pr[\ Y > kE[Y]\ ] < 1/k$
  - $E[Y] = E[(X-E[X])^2] = Var[X]$
- Hence, $\Pr[\ |X - E[X]| > \sqrt{(k\ Var[X])}\ ] < 1/k$
- Chebyshev inequality: $\Pr[\ |X - E[X]| > k\ ] < Var[Y]/k^2$
  - If $Var[X] \leq \varepsilon^2\ E[X]^2$, then $\Pr[|X - E[X]| > \varepsilon\ E[X]\ ] = O(1)$

F_2 estimation

- AMS sketch (for Alon-Matias-Szegedy) proposed in 1996
  - Allows estimation of F_2 (second frequency moment)
  - Used at the heart of many streaming and non-streaming applications: achieves dimensionality reduction
- Here, describe AMS sketch by generalizing CM sketch.
- Uses extra hash functions $g_1 \ldots g_{\log 1/\delta}$ $\{1\ldots U\} \rightarrow \{+1,-1\}$
- Now, given update $(j,+c)$, set $CM[k,h_k(i)] \mathrel{+}= c * g_k(j)$



F_2 analysis



- Estimate $F_2 = \text{median}_k \sum_i CM[k,i]^2$
- Each row's result is $\sum_i g(i)^2 x[i]^2 + \sum_{h(i)=h(j)} 2\, g(i)\, g(j)\, x[i]\, x[j]$
- But $g(i)^2 = -1^2 = +1^2 = 1$, and $\sum_i x[i]^2 = F_2$
- $g(i)g(j)$ has 1/2 chance of +1 or –1 : expectation is 0 …

F_2 Variance

- Expectation of row estimate $R_k = \sum_i CM[k,i]^2$ is exactly F_2
- Variance of row k, $Var[R_k]$, is an expectation:
  - $Var[R_k] = E[(\sum_{\text{buckets } b} (CM[k,b])^2 - F_2)^2]$
  - Good exercise in algebra: expand this sum and simplify
  - Many terms are zero in expectation because of terms like $g(a)g(b)g(c)g(d)$ (degree at most 4)
  - Requires that hash function g is *four-wise independent*: it behaves uniformly over subsets of size four or smaller
    - Such hash functions are easy to construct

F_2 Variance

- Terms with odd powers of $g(a)$ are zero in expectation
  - $g(a)g(b)g^2(c)$, $g(a)g(b)g(c)g(d)$, $g(a)g^3(b)$
- Leaves

$$
\begin{aligned}
Var[R_k] \le \textstyle\sum_i g^4(i)\, x[i]^4 \\
&+ 2 \textstyle\sum_{j\neq i} g^2(i)\, g^2(j)\, x[i]^2\, x[j]^2 \\
&+ 4 \textstyle\sum_{h(i)=h(j)} g^2(i)\, g^2(j)\, x[i]^2\, x[j]^2 \\
&- (x[i]^4 + \textstyle\sum_{j\neq i} 2x[i]^2\, x[j]^2) \\
&\le F_2^2/w
\end{aligned}
$$

- Row variance can finally be bounded by F_2^2/w
  - Chebyshev for $w=4/\varepsilon^2$ gives probability ¼ of failure:

$$\Pr[\ |R_k - F_2| > \varepsilon^2 F_2\ ] \le ¼$$

  - How to amplify this to small δ probability of failure?
  - Rescaling w has cost linear in $1/\delta$

Tail Inequalities for Sums

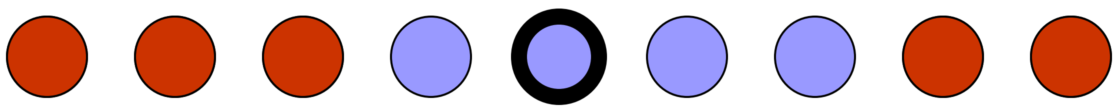
- We achieve stronger bounds on tail probabilities for the sum of *independent Bernoulli trials* via the Chernoff Bound:
  - Let $X_1, \ldots, X_m$ be independent Bernoulli trials s.t. $\Pr[X_i=1] = p$ ($\Pr[X_i=0] = 1-p$).
  - Let $X = \sum_{i=1}^{m} X_i$,and $\mu = mp$ be the expectation of X.
  - Then, for $\varepsilon>0$, Chernoff bound states:

$$\Pr[\ |X - \mu| \geq \varepsilon\mu] \leq 2 \exp(-\tfrac{1}{2}\, \mu\varepsilon^2)$$

  - Proved by applying Markov inequality to $Y = \exp(X_1 \cdot X_2 \cdot \ldots \cdot X_m)$

Applying Chernoff Bound

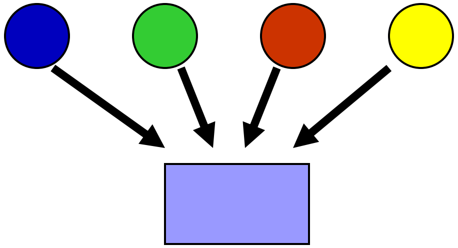
- Each row gives an estimate that is within ε relative error with probability $p' > 3/4$
- Take d repetitions and find the median. Why the median?



  - Because bad estimates are either too small or too large
  - Good estimates form a contiguous group “in the middle”
  - At least d/2 estimates must be bad for median to be bad
- Apply Chernoff bound to d independent estimates, $p=1/4$
  - Pr[More than d/2 bad estimates] < $2\exp(-d/8)$
  - So we set $d = \Theta(\ln 1/\delta)$ to give δ probability of failure
- Same outline used many times in data streams

Aside on Independence

- Full independence is expensive in a streaming setting
  - If hash functions are fully independent over n items, then we need $\Omega(n)$ space to store their description
  - Pairwise and four-wise independent hash functions can be described in a constant number of words
  - Pairwise hashing: $\Pr_{\text{over random choice of } h}[h(i) = h(j)] = 1/(\text{range}(h))$
- AMS sketch uses a careful mix of limited and full independence
  - Each hash function is four-wise independent over all n items
  - Each repetition is fully independent of all others – but there are only $O(\log 1/\delta)$ repetitions.

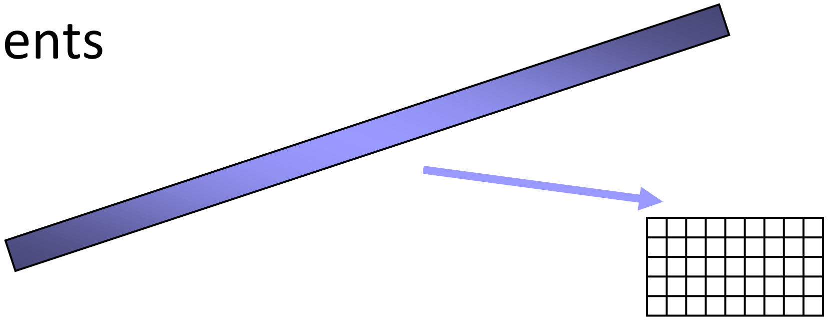


AMS Sketch Exercises

1. Let x and y be binary streams of length n.
   The Hamming distance $H(x,y) = |\{i \mid x[i] \neq y[i]\}|$
   Show how to use AMS sketches to approximate $H(x,y)$
2. Extend for strings drawn from an arbitrary alphabet
3. The inner product of two strings x, y is $x \cdot y = \sum_{i=1}^{n} x[i]*y[i]$
   Use AMS sketches to estimate $x \cdot y$
   - Hint: try computing the inner product of the sketches.
     Show the estimator is unbiased (correct in expectation)
   - What form does the error in the approximation take?
   - Use Count-Min Sketches for same problem, compare the errors.
   - Is it possible to build a $(1\pm\varepsilon)$ approximation of $x \cdot y$?

Frequency Moments

- Introduction to Frequency Moments and Sketches
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments

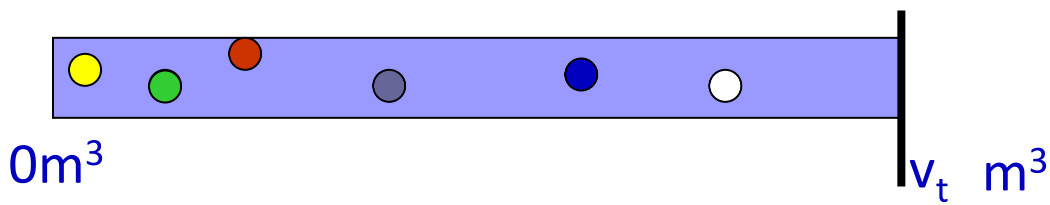


F_0 Estimation

- F_0 is the number of distinct items in the stream
  - a fundamental quantity with many applications
- Early algorithms by Flajolet and Martin [1983] gave nice hashing-based solution
  - analysis assumed fully independent hash functions
- Will describe a generalized version of the FM algorithm due to Bar-Yossef et. al with only pairwise indendence

F_0 Algorithm

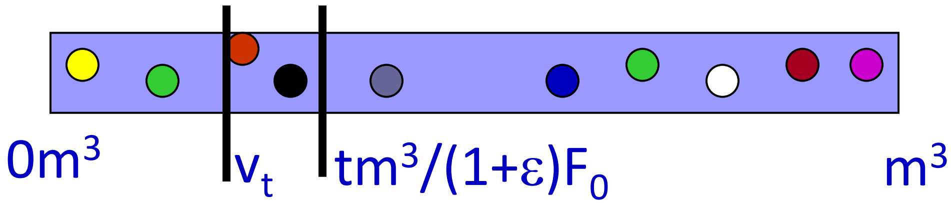
- Let m be the domain of stream elements
  - Each item in data is from [1…m]
- Pick a random (pairwise) hash function $h: [m] \to [m^3]$
  - With probability at least $1-1/m$, no collisions under h



- For each stream item i, compute h(i), and track the t distinct items achieving the smallest values of h(i)
  - Note: if same i is seen many times, h(i) is same
  - Let v_t = t'th smallest (distinct) value of h(i) seen
- If $F_0 < t$, give exact answer, else estimate $F'_0 = tm^3/v_t$
  - $v_t/m^3 \approx$ fraction of hash domain occupied by t smallest

Analysis of F_0 algorithm

- Suppose $F'_0 = tm^3/v_t > (1+\varepsilon)\ F_0$ [estimate is too high]



- So for input = set $S \in 2^{[m]}$, we have
  - $|\{ s \in S \mid h(s) < tm^3/(1+\varepsilon)F_0 \}| > t$
  - Because $\varepsilon < 1$, we have $tm^3/(1+\varepsilon)F_0 \leq (1-\varepsilon/2)tm^3/F_0$
  - $\Pr[\ h(s) < (1-\varepsilon/2)tm^3/F_0] \approx 1/m^3 * (1-\varepsilon/2)tm^3/F_0 = (1-\varepsilon/2)t/F_0$

  - (this analysis outline hides some rounding issues)

Chebyshev Analysis

- Let Y be number of items hashing to under $tm^3/(1+\varepsilon)F_0$
  - $E[Y] = F_0 * \Pr[\, h(s) < tm^3/(1+\varepsilon)F_0] = (1-\varepsilon/2)t$
  - For each item i, variance of the event = $p(1-p) < p$
  - $\mathrm{Var}[Y] = \sum_{s \in S} \mathrm{Var}[\, h(s) < tm^3/(1+\varepsilon)F_0] < (1-\varepsilon/2)t$
    - We sum variances because of pairwise independence

- Now apply Chebyshev inequality:
  - $$\begin{aligned}\Pr[\, Y > t \,] &\leq \Pr[\, |Y - E[Y]| > \varepsilon t/2] \\ &\leq 4\mathrm{Var}[Y]/\varepsilon^2 t^2 \\ &< 4t/(\varepsilon^2 t^2)\end{aligned}$$
  - Set $t=20/\varepsilon^2$ to make this $\mathrm{Prob} \leq 1/5$

Completing the analysis

- We have shown

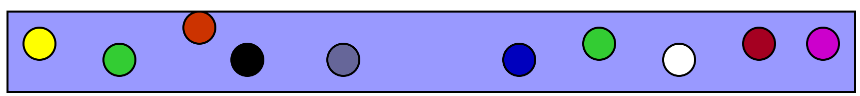
  $\Pr[\ F'_0 > (1+\varepsilon)\ F_0\ ] < 1/5$

- Can show $\Pr[\ F'_0 < (1-\varepsilon)\ F_0\ ] < 1/5$ similarly
  - too few items hash below a certain value
- So $\Pr[\ (1-\varepsilon)\ F_0 \leq F'_0 \leq (1+\varepsilon)F_0] > 3/5$ [Good estimate]

- Amplify this probability: repeat $O(\log 1/\delta)$ times in parallel with different choices of hash function h
  - Take the median of the estimates, analysis as before

F_0 Issues

- Space cost:
  - Store t hash values, so $O(1/\varepsilon^2 \log m)$ bits
  - Can improve to $O(1/\varepsilon^2 + \log m)$ with additional tricks



- Time cost:
  - Find if hash value $h(i) < v_t$
  - Update v_t and list of t smallest if $h(i)$ not already present
  - Total time $O(\log 1/\varepsilon + \log m)$ worst case

F_0 applications

- Many cases where we want to track number of distinct items
- Can also estimate size of subpopulations
  - E.g. "How many distinct Firefox users visited my network?"
  - Compute fraction of the t satisfying the predicate
  - Error is (additive) εF_0
- Compare F_0 to Bloom Filter
  - Bloom Filter: $\Omega(n)$ space to test membership of set of n items
  - F_0 estimation: $O(1/\varepsilon^2)$ space to approximate size of set

Range Efficiency

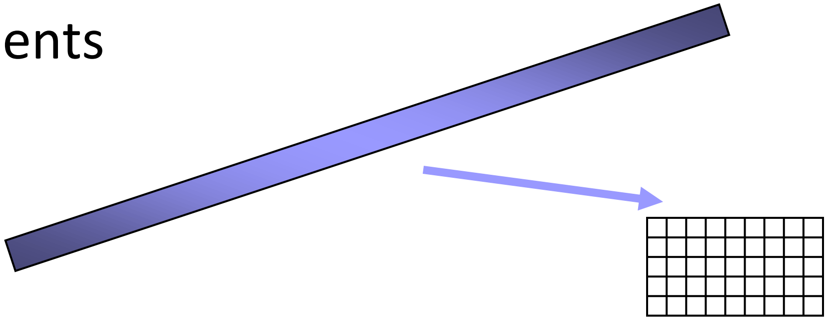
- Sometimes input is specified as a stream of ranges [a,b]
  - [a,b] means insert all items (a, a+1, a+2 … b)
  - Trivial solution: just insert each item in the range
- Range efficient F_0 [Pavan, Tirthapura 05]
  - Start with an alg for F_0 based on pairwise hash functions
  - Key problem: track which items hash into a certain range
  - Dives into hash fns to divide and conquer for ranges
- Range efficient F_2 [Calderbank et al. 05, Rusu,Dobra 06]
  - Start with sketches for F_2 which sum hash values
  - Design new hash functions so that range sums are fast

F_0 Exercises

- Suppose the stream consists of a sequence of insertions and deletions.
  Design an algorithm to approximate F_0 of the current set.
  – What happens when some frequencies are negative?
- Give an algorithm to find F_0 of the most recent W arrivals
- Use F_0 algorithms to approximate Max-dominance: given a stream of pairs (i,x(i)), approximate $\sum_i \max_{(i,\, x(i))} x(i)$

Frequency Moments

- Intro to frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments



Higher Frequency Moments

- F_k for $k>2$. Use a sampling trick [Alon et al 96]:
  - Uniformly pick an item from the stream length $1\ldots n$
  - Set r = how many times that item appears subsequently
  - Set estimate $F'_k = n(r^k - (r-1)^k)$

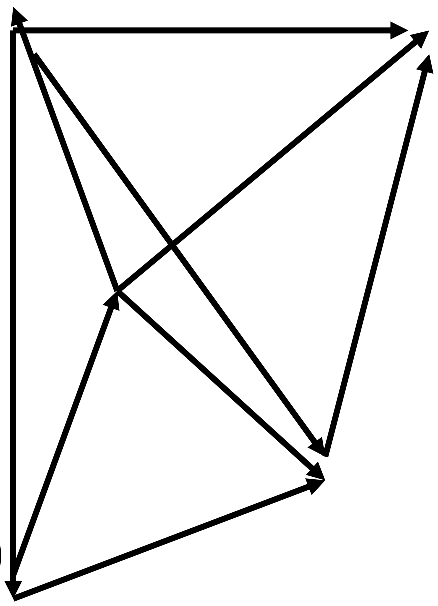
- $E[F'_k]=1/n*n*[\ f_1^k - (f_1-1)^k + (f_1-1)^k - (f_1-2)^k + \ldots + 1^k-0^k]+\ldots$
  $= f_1^k + f_2^k + \ldots = F_k$
- $Var[F'_k] \leq 1/n*n^2*[(f_1^k-(f_1-1)^k)^2 + \ldots]$
  - Use various bounds to bound the variance by $k\ m^{1-1/k}\ F_k^2$
  - Repeat $k\ m^{1-1/k}$ times in parallel to reduce variance
- Total space needed is $O(k\ m^{1-1/k})$ machine words
  - Not a sketch: does not distribute easily

Improvements

- [Coppersmith and Kumar '04]: Generalize the F_2 approach
  - E.g. For F_3, set $p=1/\sqrt{m}$, and hash items onto $\{1-1/p, -1/p\}$ with probability $\{1/p, 1-1/p\}$ respectively.
  - Compute cube of sum of the hash values of the stream
  - Correct in expectation, bound variance $\leq O(\sqrt{m}F_3^2)$
- [Indyk, Woodruff '05, Bhuvangiri et al. '06]: Optimal solutions by extracting different frequencies
  - Use hashing to sample items and f_i's, combine to build estimator
  - Cost is $O(m^{1-2/k}\text{ poly-log}(m,n,1/\varepsilon))$ space
- [Andoni Krauthgamer Onak 11]: Precision sampling via sketches
  - Recover frequencies at different precision to get optimal bounds

Combined Frequency Moments

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

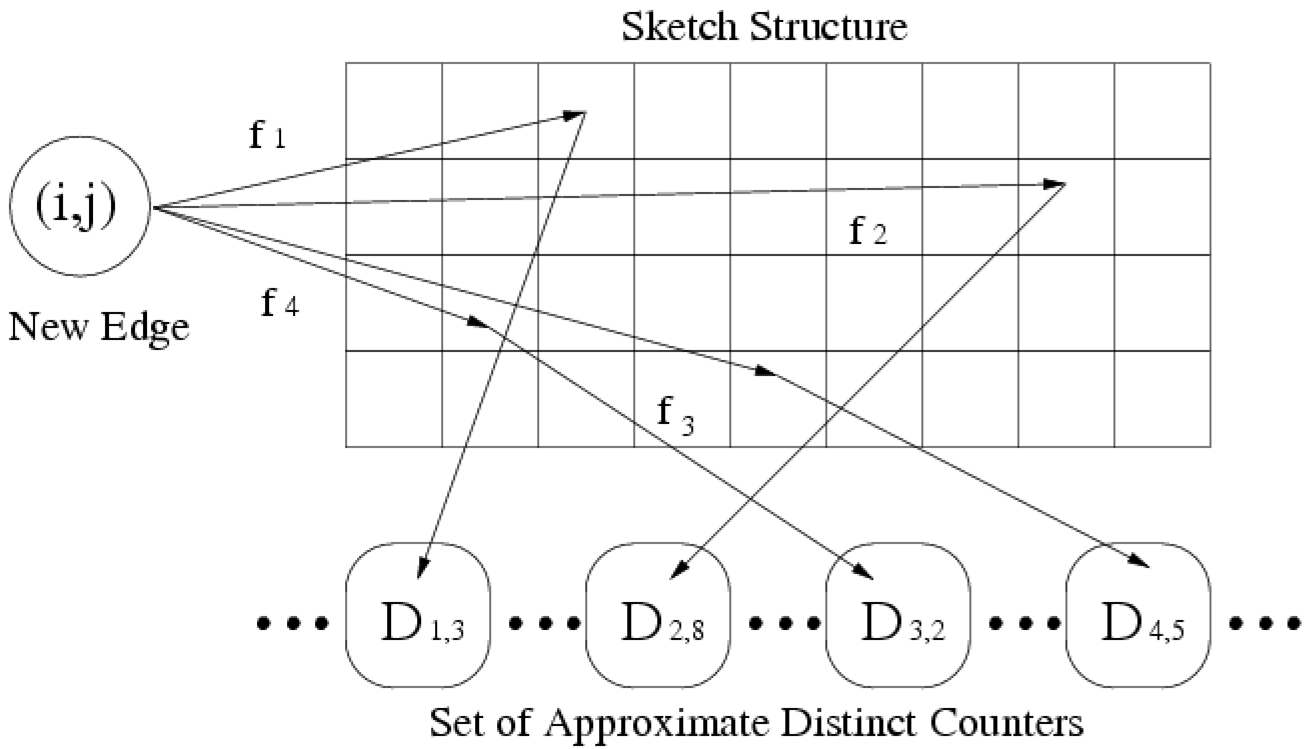
- Want to focus on number of distinct communication pairs, not size of communication
- So want to compute moments of F_0 values...

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)

F_∞ (F_0) using CM-FM

- Find i's such that $d_i > \phi \sum_i d_i$
  Finds the people that talk to many others
- Count-Min sketch only uses additions, so can apply:



Accuracy for $F_\infty(F_0)$

- Focus on point query accuracy: estimate d_i.
- Can prove estimate has only small bias in expectation
  - Analysis is similar to original CM sketch analysis, but now have to take account of F_0 estimation of counts
- Gives an bound of $O(1/\varepsilon^3 \text{ poly-log}(n))$ space:
  - The product of the size of the sketches

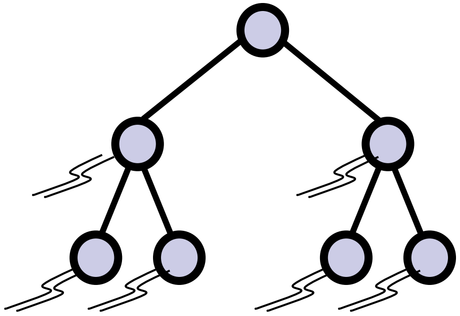
- Other combinations of functions require fresh analysis, eg. $F_2(F_0)$, $F_2(F_2)$ etc.

Exercises / Problems

1. (Research problem) What can be computed for other combinations of frequency moments, e.g. F_2 of F_2 values, etc.?
2. The F_2 algorithm uses the fact that +1/-1 values square to preserve F_2 but are 0 in expectation. Why won't it work to estimate F_4 with $g \rightarrow \{-1, +1, -i, +i\}$?
3. (Research problem) Read, understand and **simplify** analysis for optimal F_k estimation algorithms
4. Take the sampling F_k algorithm and combine it with F_0 estimators to approximate F_k of node degrees
5. Why can't we use the sketch approach for F_2 of node degrees? Show there the analysis breaks down

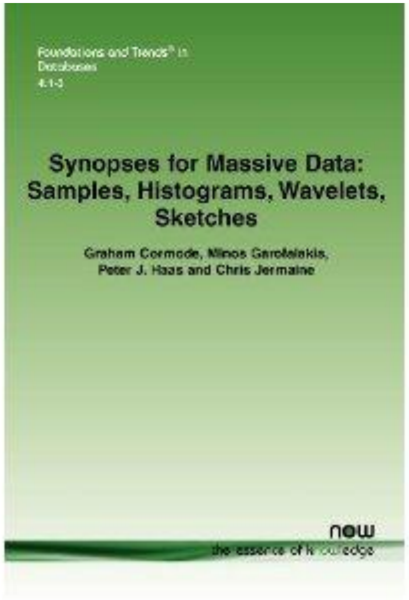
Distributed Issues

- Sketches like these are easy to compute in distributed setting
  - CM Sketch, AMS sketch: merge by adding sketches
  - F_0 sketch: can merge and take bottom-k set of items
  - Combined sketches also merge naturally
- Stronger property: these sketches are data-order independent
- Other questions remain:
  - *When* to merge?
  - *How* to deal with failure/loss?

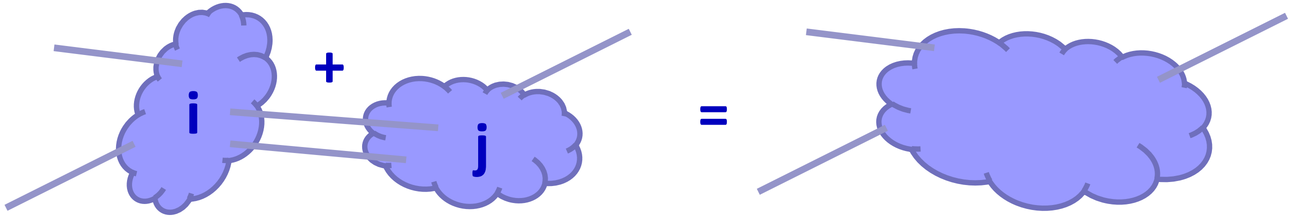


Sketching Resources

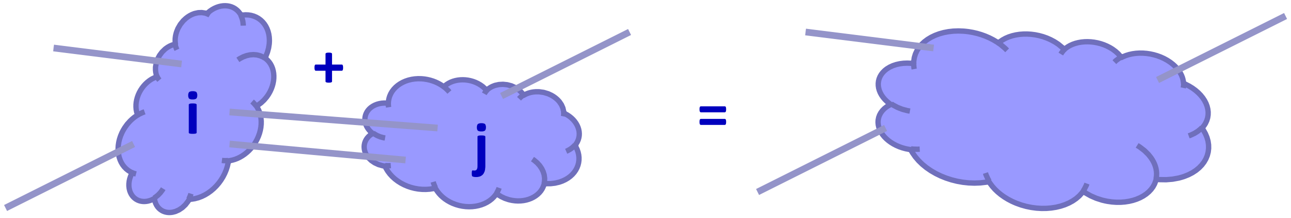
- Sample implementations on web
  - Ad hoc, of varying quality
- Technical descriptions
  - Original papers
  - Surveys, comparisons
- (Partial) wikis and book chapters
  - Wiki: **`sites.google.com/site/countminsketch/`**
  - “Sketch Techniques for Approximate Query Processing”
    `dimacs.rutgers.edu/~graham/pubs/papers/sk.pdf`



L_p Sampling and Stream Verification



L_p Sampling:
Sampling from Sketches



Sampling from Sketches

- Given distributed streams with positive and negative weights
- Want to sample based on the overall frequency distribution
  - Sample from support set of n possible items
  - Sample proportional to (absolute) weights
  - Sample proportional to some function of weights
- How to do this sampling effectively?
- Recent approach: L_p sampling

L_p Sampling

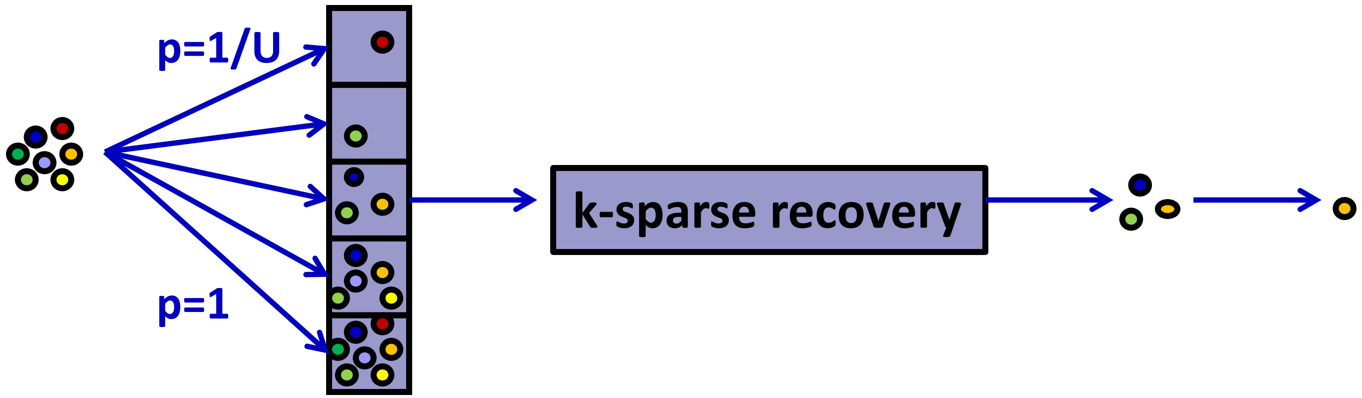
- L_p sampling: use sketches to sample i w/prob $(1\pm\varepsilon)\ f_i^p/\|f\|_p^p$
- "Efficient" solutions developed of size $O(\varepsilon^{-2} \log^2 n)$
  - [Monemizadeh, Woodruff 10] [Jowhari, Saglam, Tardos 11]
- L_0 sampling enables novel "graph sketching" techniques
  - Sketches for connectivity, sparsifiers [Ahn, Guha, McGregor 12]
- L_2 sampling allows optimal estimation of frequency moments

- Challenge: improve space efficiency of L_p sampling
  - Empirically or analytically

L_0 Sampling

- L_0 sampling: sample with prob $(1\pm\varepsilon)\ f_i^0/F_0$
  - i.e., sample (near) uniformly from items with non-zero frequency
- General approach: [Frahling, Indyk, Sohler 05, C., Muthu, Rozenbaum 05]
  - Sub-sample all items (present or not) with probability p
  - Generate a sub-sampled vector of frequencies f_p
  - Feed f_p to a *k-sparse recovery* data structure
    - Allows reconstruction of f_p if $F_0 < k$
  - If f_p is k-sparse, sample from reconstructed vector
  - Repeat in parallel for exponentially shrinking values of p

Sampling Process



- Exponential set of probabilities, p=1, ½, ¼, 1/8, 1/16… 1/U
  - Let $N = F_0 = |\{ i : f_i \neq 0\}|$
  - Want there to be a level where k-sparse recovery will succeed
  - At level p, expected number of items selected S is Np
  - Pick level p so that $k/3 < Np \leq 2k/3$
- Chernoff bound: with probability exponential in k, $1 \leq S \leq k$
  - Pick $k = O(\log 1/\delta)$ to get $1-\delta$ probability

k-Sparse Recovery

- Given vector x with at most k non-zeros, recover x via sketching
  - A core problem in compressed sensing/compressive sampling
- First approach: Use Count-Min sketch of x
  - Probe all U items, find those with non-zero estimated frequency
  - Slow recovery: takes O(U) time
- Faster approach: also keep sum of item identifiers in each cell
  - Sum/count will reveal item id
  - Avoid false positives: keep fingerprint of items in each cell
- Can keep a sketch of size O(k log U) to recover up to k items

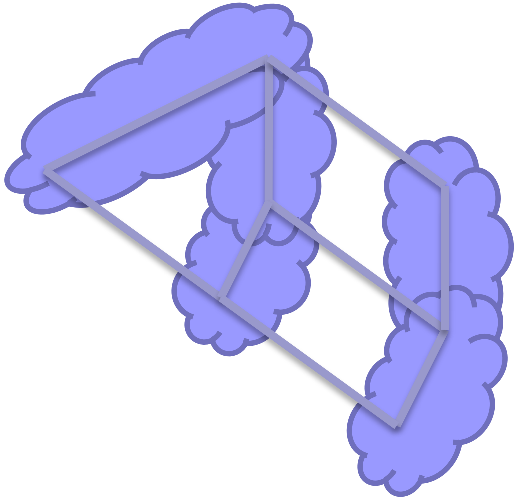
Sum, $\sum_{i : h(i)=j} i$
Count, $\sum_{i : h(i)=j} x_i$
Fingerprint, $\sum_{i : h(i)=j} x_i r^i$

Uniformity

- Also need to argue sample is uniform
  - Failure to recover could bias the process
- Pr[i would be picked if k=n] = $1/F_0$ by symmetry
- Pr[i is picked] = Pr[i would be picked if k=n $\wedge$ S$\leq$ k]
  $\geq (1-\delta)/F_0$
- So $(1-\delta)/N \leq$ Pr[i is picked] $\leq 1/N$
- Sufficiently uniform (pick $\delta = \varepsilon$)

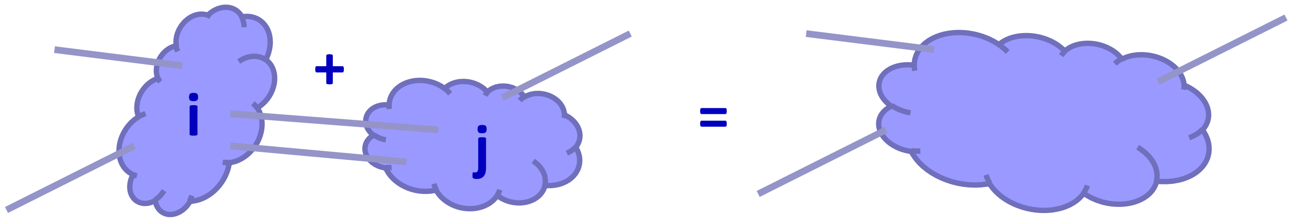
Application: Graph Sketching

- Given L_0 sampler, use to sketch (undirected) graph properties
- Connectivity: want to test if there is a path between all pairs
- Basic alg: repeatedly contract edges between components
- Use L_0 sampling to provide edges on vector of adjacencies
- Problem: as components grow, sampling most likely to produce internal links



Graph Sketching

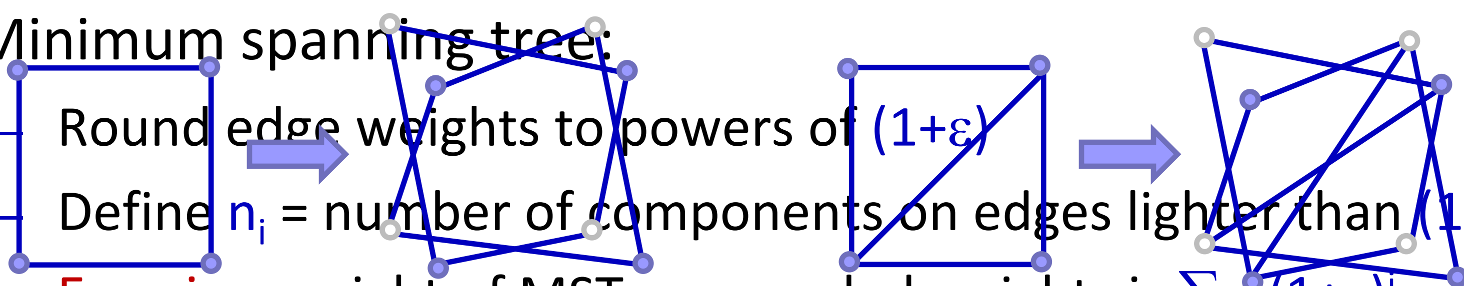
- Idea: use clever encoding of edges
- Encode edge (i,j) as ((i,j),+1) for node i<j, as ((i,j),-1) for node j>i
- When node i and node j get merged, sum their L_0 sketches
  - Contribution of edge (i,j) exactly cancels out



- Only non-internal edges remain in the L_0 sketches
- Use independent sketches for each iteration of the algorithm
  - Only need O(log n) rounds with high probability
- Result: O(poly-log n) space per node for connectivity

Other Graph Results via sketching

- K-connectivity via connectivity
  - Use connectivity result to find and remove a spanning forest
  - Repeat k times to generate k spanning forests $F_1, F_2, \ldots F_k$
  - Theorem: G is k-connected if $\cup_{i=1}^{k} F_i$ is k-connected
- Bipartiteness via connectivity:
  - Compute c = number of connected components in G
  - Generate G' over $V \cup V'$ so $(u,v) \in E \Rightarrow (u, v') \in E', (u', v) \in E'$
  - If G is bipartite, G' has $2c$ components, else it has c components
- Minimum spanning tree:
  - Round edge weights to powers of $(1+\varepsilon)$
  - Define n_i = number of components on edges lighter than $(1+\varepsilon)^i$
  - Exercise: weight of MST on rounded weights is $\sum_i \varepsilon(1+\varepsilon)^i n_i$



Application: F_k via L_2 Sampling

- Recall, $F_k = \sum_i f_i^k$
- L_2 sampling lets us sample f_i with probability f_i^2/F_2
  - Also estimate sampled f_i with relative error ε
- Estimator: $X = F_2 f_i^{k-2}$ (with estimates of F_2, f_i)
  - Expectation: $E[X] = F_2 \sum_i f_i^{k-2} \cdot f_i^2 / F_2 = F_k$
  - Variance: $Var[X] \leq E[X^2] = \sum_i f_i^2/F_2 (F_2 f_i^{k-2})^2 = F_2 F_{2k-2}$

Rewriting the Variance

- Want to express variance of $F_2\, F_{2k-2}$ in terms of F_k
- Hölder's inequality: $\langle x, y\rangle \leq \|x\|_p\, \|y\|_q$ for $1 \leq p, q$ with $1/p+1/q=1$
  - Generalizes Cauchy-Shwarz inequality, where $p=q=2$.
- So pick $p=k/(k-2)$ and $q = k/2$ for $k > 2$. Then

$$\langle 1^n, (f_i)^2\rangle \leq \|1^n\|_{k/(k-2)}\, \|(f_i)^2\|_{k/2}$$

$$F_2 \leq n^{(k-2)/k}\, F_k^{2/k} \qquad (1)$$

- Also, since $\|x\|_{p+a} \leq \|x\|_p$ for any $p \geq 1$, $a > 0$
  - Thus $\|x\|_{2k-2} \leq \|x\|_k$ for $k \geq 2$
  - So $F_{2k-2} = \|f\|_{2k-2}^{2k-2} \leq \|f\|_k^{2k-2} = F_k^{2-2/k}$ (2)
- Multiply (1) * (2) : $F_2\, F_{2k-2} \leq n^{1-2/k}\, F_k^2$
  - So variance is bounded by $n^{1-2/k}\, F_k^2$

F_k Estimation

- For $k \geq 3$, we can estimate F_k via L_2 sampling:
  - Variance of our estimate is $O(F_k^2\, n^{1-2/k})$
  - Take mean of $n^{1-2/k}\varepsilon^{-2}$ repetitions to reduce variance
  - Apply Chebyshev inequality: constant prob of good estimate
  - Chernoff bounds: $O(\log 1/\delta)$ repetitions reduces prob to δ
- How to instantiate this?
  - Design method for approximate L_2 sampling via sketches
  - Show that this gives relative error approximation of f_i
  - Use approximate value of F_2 from sketch
  - Complicates the analysis, but bound stays similar

L_2 Sampling Outline

- For each i, draw u_i uniformly in the range $0\ldots1$
  – From vector of frequencies f, derive g so $g_i = f_i/u_i$
  – Sketch g_i vector (can do this over distributed streams)
- Sample: return (i, f_i) if there is unique i with $g_i^2 > t=F_2/\varepsilon$ threshold
  – $$\begin{aligned}\Pr[\, g_i^2 > t \wedge \forall\, j \neq i : g_j^2 < t] &= \Pr[g_i^2 > t] \prod_{j\neq i} \Pr[g_j^2 < t] \\ &= \Pr[u_i < \varepsilon f_i^2/F_2] \prod_{j\neq i} \Pr[u_j > \varepsilon f_j^2/F_2] \\ &= (\varepsilon f_i^2/F_2)\prod_{j\neq i} (1 - \varepsilon f_j^2/F_2) \\ &\approx \varepsilon f_i^2/F_2\end{aligned}$$
- Probability of returning anything is not so big: $\sum_i \varepsilon\, f_i^2/F_2 = \varepsilon$
  – Repeat $O(1/\varepsilon \log 1/\delta)$ times to improve chance of sampling

L_2 sampling continued

- Given (estimated) g_i s.t. $g_i^2 \geq F_2/\varepsilon$, estimate $f_i = u_i\ g_i$
- Sketch size $O(\varepsilon^{-1} \log n)$ means estimate of f_i^2 has error $(\varepsilon f_i^2 + u_i^2)$
  - With high prob, no $u_i < 1/\text{poly}(n)$, and so $F_2(g) = O(F_2(f) \log n)$
  - Since estimated $f_i^2/u_i^2 \geq F_2/\varepsilon$, $u_i^2 \leq \varepsilon f_i^2/F_2$
- Estimating f_i^2 with error εf_i^2 sufficient for estimating F_k

- Many details skipped…

L_p Sampling exercises

1. Work through details of sketch-based L_2 sampling for F_k estimation. Can the details be simplified?
2. (Research) The bound for F_k estimation is optimal in n (depends on $n^{1-2/k}$), but not ε (ε^{-4}). Can this be improved?
3. (Research) The graph sampling result only requires an arbitrary sample to be drawn. What is the best sketch for this?
4. Design a graph sketch so that given a set S we can approximate cut(S), the number of edges in $E \cap (S \times (V \setminus S))$

Stream Verification:

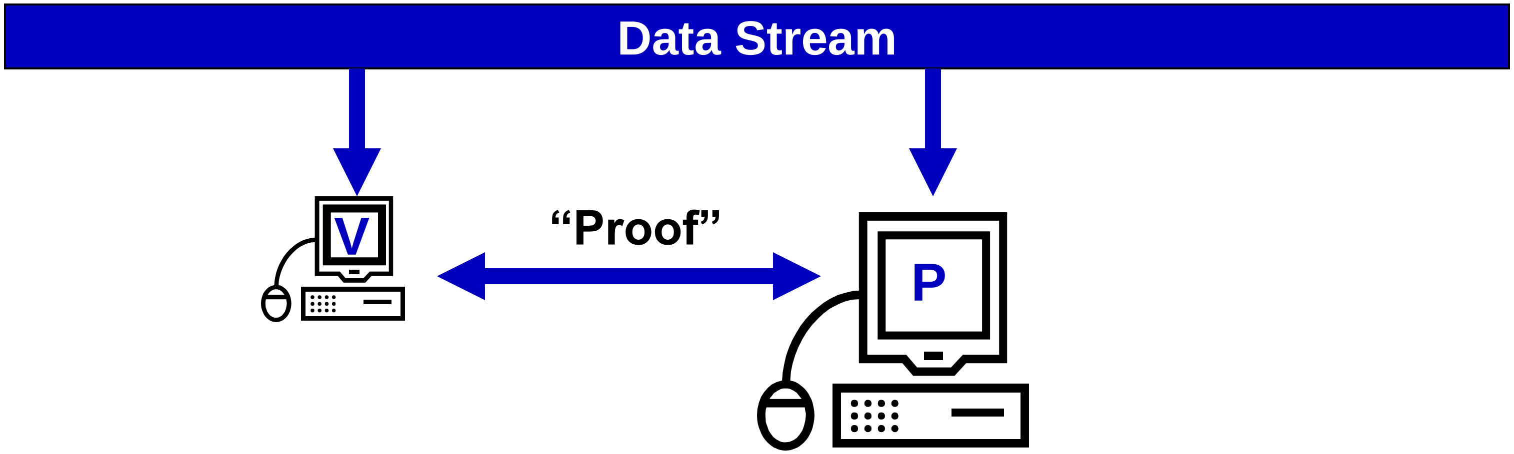
Checking the cloud with a stream

Outsourced Computation

- Current trend to 'outsource' computation
  - Cloud computing: Amazon EC2, Microsoft Azure…
  - Hardware support: multicore systems, graphics cards
- We provide data to a third party, they return an answer
- How can we be sure that the computation is correct?
  - Duplicate the whole computation ourselves?
  - Find some ad hoc sanity checks on the answer?
- This talk: construct protocols to prove the correctness
  - Protocols must be very low cost for the data owner (streaming)
  - Amount of information transmitted should not be too large

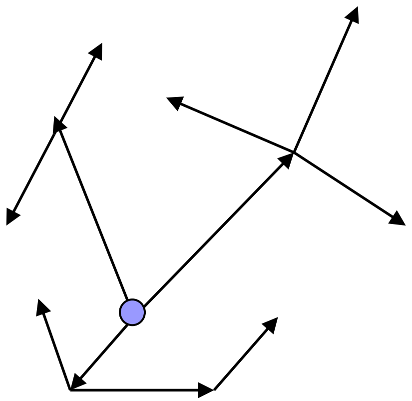
Streaming Proofs

- Objective: prove integrity of the computed solution
  - Not concerned with security: third party sees unencrypted data
- Prover provides "proof" of the correct answer
  - Ensure that "verifier" has very low probability of being fooled
  - Related to communication complexity Arthur-Merlin model, and Arithmetization, with additional streaming constraints



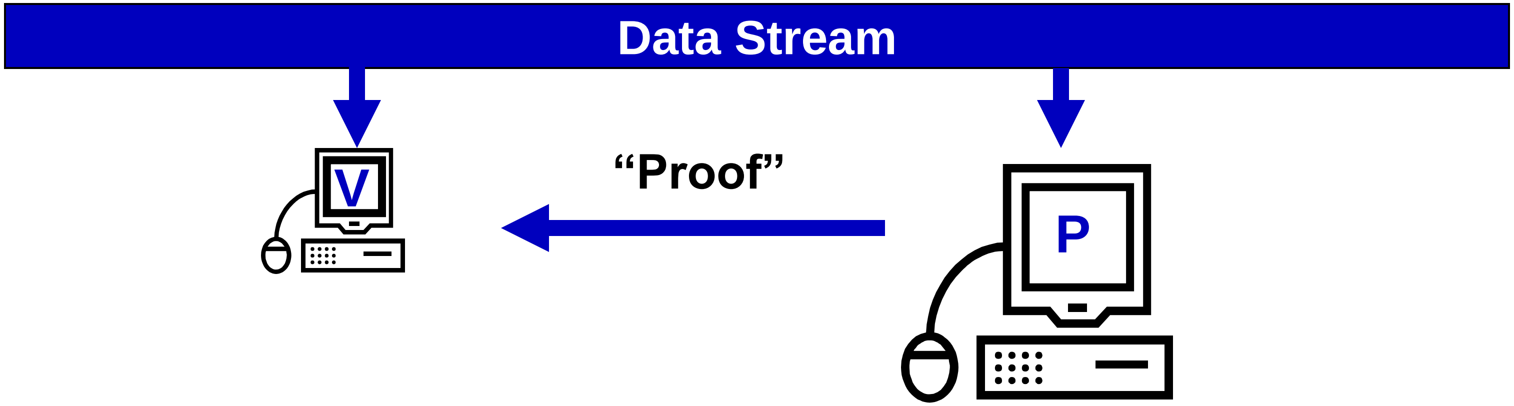
Problem Setting

- Data is large, so is not stored in full by the verifier
  - (Distributed) streaming model: verifier sees data in one pass
- Consider large graph data
  - Verifier sees graph edge by edge in no particular order
- Want to solve graph problems
  - Bipartite? Connected?
  - Max flow/min cut
  - Shortest s-t paths
  - Matchings and MST
  - Counting triangles



One Round Model

- One-round model [Chakrabarti, C, McGregor 09]
  - Define protocol with help function h over input length N
  - Maximum length of h over all inputs defines *help cost*, H
  - Verifier has V bits of memory to work in
  - Verifier uses randomness so that:
    - For all help strings, $\Pr[\text{output} \neq f(x)] \leq \delta$
    - Exists a help string so that $\Pr[\text{output} = f(x)] \geq 1-\delta$
  - H = 0, V = N is trivial; but H = N, V = polylog N is not



Basic Tool: Fingerprints

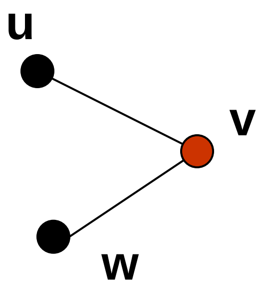
- Fingerprints allow us to test if two vectors (matrices, multisets) are equal, with high probability

- Pick a prime p, and a random $\alpha \in [p]$
  - Given vector x of dimension q
  - Compute $f(x) = \sum_i x_i \alpha^i \bmod p$
  - $\Pr[\, f(x) = f(y) \mid x \neq y \,] \leq q/p$

- Linearity: f can be computed incrementally as x is observed
  - E.g. if x is a stream of edges, can fingerprint multiset of nodes

Warm Up: Bipartiteness

- Prove a graph is bipartite: exhibit a bipartition
  - List each edge with different labels on each node
- Ensure consistent labels via fingerprinting
  - Fingerprint the (multi)set (node, label, degree)
  - Compare to fingerprint of claimed labeling
- Prove a graph is not bipartite: exhibit an odd cycle
  - List all other edges, so fingerprints can be compared
- Either way, size of proof is O(|E|) = O(m)
  - Verifier only needs to store O(1) fingerprints of O(log n) bits
- Similar arguments for other problems, via (lots of) fingerprints



(u, B, 1)
(v, R, 2)
(w, B, 1)

(u, B, v, R)
(v, R, w, B)

General Simulation Argument

- General simulation argument:
  Given a (deterministic) algorithm in the RAM model that solves a problem in time $t(m,n)$, there is a protocol with proof size $O(m + t(m,n))$, using $O(1)$ fingerprints
  - Main idea: use memory checking techniques
  - Verifier runs the algorithm, proof provides result of each memory access
  - Verifier uses fingerprints of reads and writes to memory to ensure consistency
    - Every memory access is a read followed by a write

Applications of Simulation Argument

- Immediately provides:
  - $O(m + n \log n)$ size proof for single-source shortest paths
  - $O(n^3)$ size proof for all-pairs shortest paths
- Minimum spanning tree has an $O(n)$ sized proof
  - Based on linear time algorithms to verify that a tree is minimal
- Limitation: small space but large proof
  - Can we reduce proof sizes by using slightly more space?

Streaming ILP problem

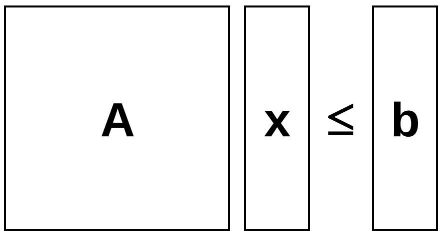
- Stream defines (non-zero) entries of matrix A, vectors b and c
  - Updates "add k to entry (i,j) of A" in arbitrary order
  - Goal: prove x satisfies $\min \{c^T x \mid Ax \leq b\}$

- Use primal-duality and matrix-vector multiplication:
  1. Provide primal-feasible solution x
  2. For each row i of A:

     List x_i, c_i, b_i to find $c_i\, x_i$

     List non-zeros A_i and corresponding entries of x to find $A_i x$

     Verifier uses fingerprints ensure consistency
  3. Repeat for dual-feasible solution y s.t. $A^T y \leq c$
  4. Accept if $c^T x = b^T y$

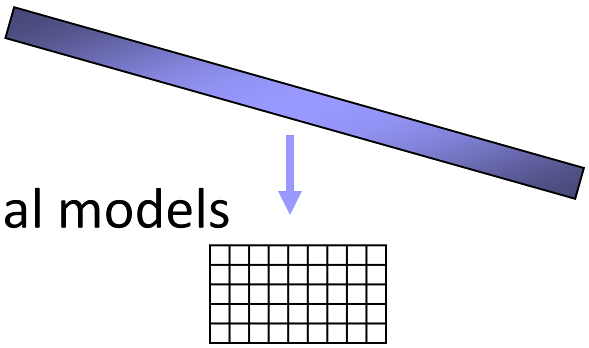


Application to Graph Streams

- Applies to Totally Unimodular Integer Programs (TUM IPs)
  - Optimality when primal=dual for LP relaxation
- Gives protocols for (flow) problems formulated as TUM IPs:
  - Max-flow, min-cut
  - Minimum-weigh bipartite matching
  - Shortest s-t path
- Size of proof = $|A|$ = size of constraints = $|E|$
  - Verifier only remembers a constant number of fingerprints
- Can show lower bound of n^2 on (HV) product
  - Via reduction to canonical "hard" problem of **INDEX**
- Can we increase space to decrease proof size?

Inner Product Computation

- Given vectors a, b, defined in the stream, want to compute $a \cdot b$
- Inner product appears in many problems
  - Core computation in data streams
  - Requires $\Omega(N)$ space to compute in traditional models



- Results: for h,v s.t. $(hv) > N$, there exists a protocol with proof size $O(h \log m)$, and space $O(v \log m)$ to compute inner product
  - Lower bounds: $HV = \Omega(N)$ necessary for exact computation

Inner Product Protocol

- Map [N] to $h \times v$ array
- Interpolate entries in array as polynomials $a(x,y)$, $b(x,y)$
- Verifier picks random r, evaluates $a(r, j)$ and $b(r,j)$ for $j \in [v]$
- Helper sends $s(x) = \sum_{j \in [v]} a(x, j)b(x,j)$ (degree h)
  - Verifier checks $s(r) = \sum_{j \in [v]} a(r,j)b(r,j)$
  - Output $a \cdot b = \sum_{i \in [h]} s(i)$ if test passed
- Probability of failure small if evaluated over large enough field
  - A “Low Degree Extension” / arithmetization technique

3	7	1	2	0	8	5	9	1	1	1	0

3	7	1	2
0	8	5	9
1	1	1	0

Streaming Computation

- Must evaluate $a(r,j)$ incrementally as $a()$ is defined by stream
- Structure of polynomial means updates to (w,z) cause

$$a(r,j) \leftarrow a(r,j) + p_{w,z}(r,j)$$

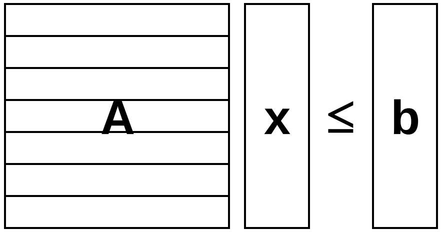
where $p_{w,z}(x,y) = \prod_{i \in [h]\backslash\{w\}} (x-i)(w-i)^{-1} \cdot \prod_{j \in [v]\backslash\{z\}} (y-j)(z-j)^{-1}$

  – p is a Lagrange polynomial corresponding to an impulse at (w,z)

- Can be computed quickly, using appropriate precomputed look-up tables
- Evaluation is linear: can be computed over distributed data

Matrix-Vector Computation

- For Linear Programming, need to verify matrix-vector product
  - Equivalent to multiple vector-vector products
  - Use the fact that one vector is held constant



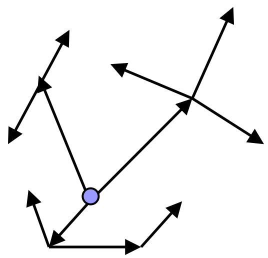
- Central idea: “simulate” the multiple vector-vector protocols
  - Don’t explicitly store the evaluations of $a_i(r,j)$ for all i
  - Instead, keep a fingerprint of the evaluations
  - Evaluate the polynomials $s_i(x)$ at r, fingerprint the result vector
- Scale and add fingerprints of $a_i(r,j)$
  - By linearity, $f(\sum_j a_i(r,j)\, b(r,j)) = \sum_j b(r,j)\, f(a_i(r,j))$
  - Accept if the two fingerprints match—
    Implies whp that *all* the tests $s_i(r) = \sum_j a_i(r,j)\, b(r,j)$ passed

Matrix-Vector Bounds

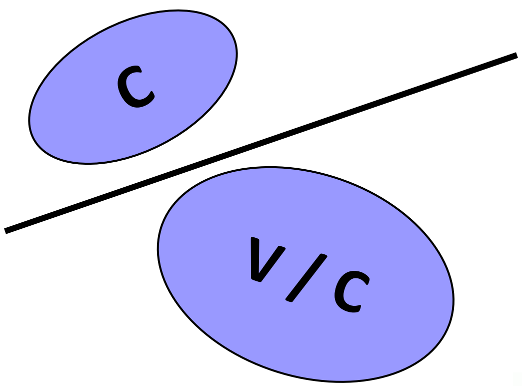
- "Tradeoff" to verify $Ax=b$, for $n \times n$ matrix A
  - Size of proof = $O(n^{1+\alpha})$ for any $0<\alpha<1$
  - Space of verifier = $O(n^{1-\alpha})$
- Applies to a variety of problems:
  - Protocols for dense LPs
  - Connectivity
  - Max bipartite matching

Connectivity Tradeoff

- If graph is connected, can easily verify a spanning tree T
- Challenge: show that $T \subseteq E$!
  - Outline: show $\langle T, E \rangle = |n|$
  - Inner product: $n^{1+\alpha}$ proof, $n^{1-\alpha}$ space

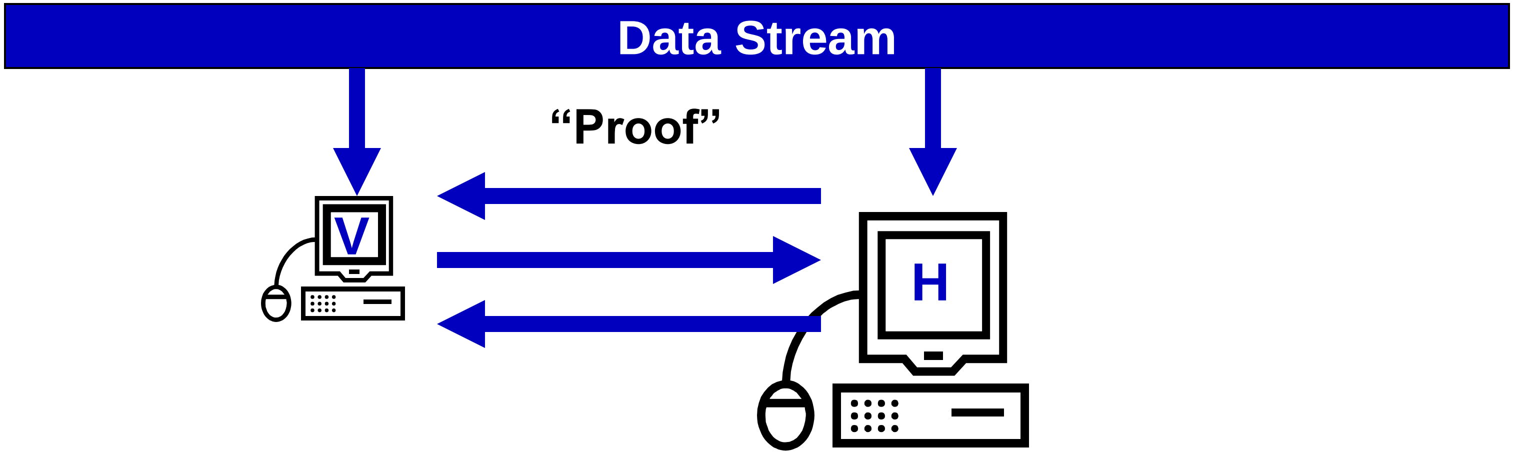


- If graph is not connected, prover lists connected component C
- Challenge: verify that no edges leave C: $E \cap (C \times (V / C)) = 0$
  - Use matrix-vector protocol to compute EC
  - Check that support set of EC is C
  - Cost bounded: $n^{1+\alpha}$ proof, $n^{1-\alpha}$ space



Multi-Round Protocols

- Advantage of one-round protocols: Prover can provide proof without direct interaction (e.g. publish + go offline)
- Disadvantage: Resources still polynomial in input size
- Multi-round protocol can improve exponentially [C,Thaler,Yi 12]:
  - Prover and Verifier follow communication protocol
  - H now denotes upper bound on total communication
  - V is verifier's space, study tradeoff between H and V as before



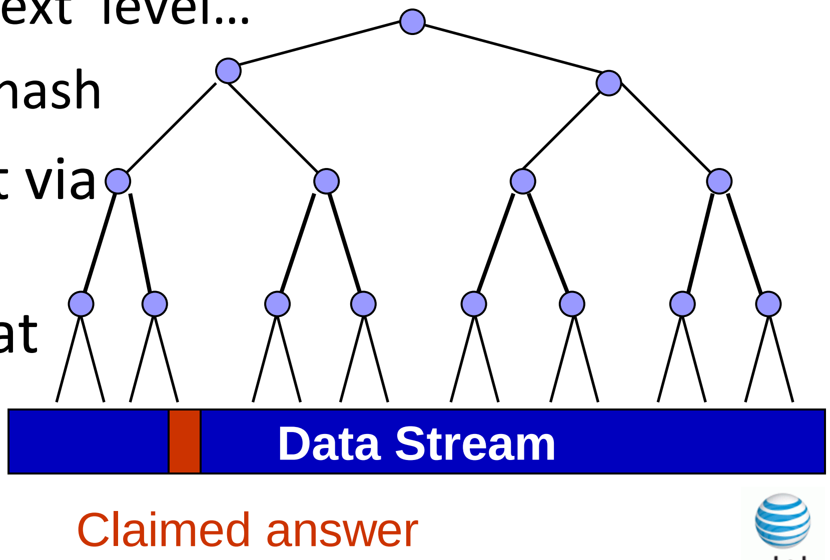
General Theorems

- Universal Arguments works in this model [Kilian 92]
  - Implies computationally sound polylog n sized proof with polylog n space for all of NP
- "Interactive Proofs for Muggles" [Goldwasser et al 08]
  - Implies statistically sound polylog n size proof with polylog n space for all of NP
- In both cases, verifier computes LDE of input data

- Challenge: these protocols are potentially unwieldy (e.g. Universal Arguments depends on building a PCP)
  - Can we find cheaper solutions for certain problems?

Multi-Round Index Protocol

- Basic idea: V keeps hash of whole stream, use helper to help check hash of stream containing claimed answer
  - Verifier imposes a binary tree, and a (secret) hash for each level
  - Round 1: Prover sends answer, and its sibling
    Verifier sends hash for leaf level
  - Round 2: Prover sends hash of answer's parent's sibling
    Verifier sends hash for next level…
  - Round log N: Verifier checks root hash
- Correctness: Prover can only cheat via hash collisions—but doesn't know hash function until too late to cheat
  - Small chance over log N levels



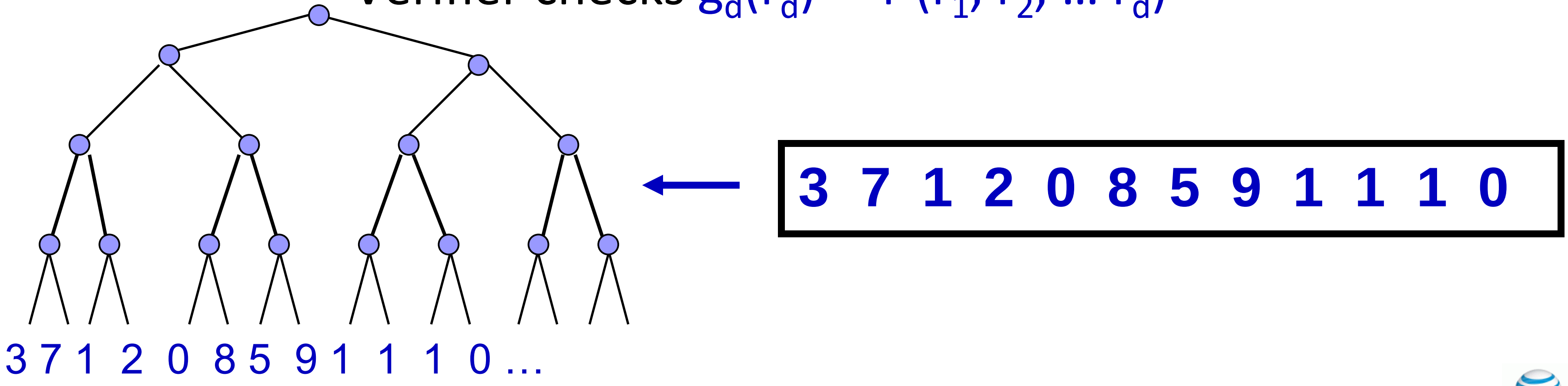
Multi-Round Index Protocol

- Challenge: Verifier must compute hash of root in small space
- h(root) $= h_{\log N}(h_{\log N-1}(\text{left half}), h_{\log N-1}(\text{right half}))$
  $= h_{\log N}(h_{\log N} \ldots h_2 (h_1 (x_1, x_2) \ldots .)))$
- Solution: appropriate choice of each hash function
  - $h_i(x, y) = x + r_i\, y \bmod p$ gives sufficient security ($1/p \log N$ error)
  - Then $h(\text{root}) = \sum_i (w_i \prod_{j=1}^{\log N} r_j^{\text{bit}(j,i)})$ where $\text{bit}(j,i)$ = i'th bit of j
  - So each update requires only log N field multiplications
- Final bounds: $O(\log^2 N)$ communication, $O(\log^2 N)$ space

Multi-Round Frequency Moments

Now index data using $\{0,1\}^d$ in $d = \log N$ dimensional space

- Verifier picks one $(r_1 \ldots r_d) \in [p]^d$, and evaluates $f^k(r_1, r_2, \ldots r_d)$
- Round 1: Prover sends $g_1(x_1)=\sum_{x_2 \ldots x_d} f^k(x_1, x_2 \ldots x_d)$, V sends r_1
- Round i: Prover sends $g_i(x_i) = \sum_{x_{i+1} \ldots x_d} f^k(r_1, r_2 \ldots r_{i-1}, x_i, x_{i+1} \ldots x_d)$
  Verifier checks $g_{i-1}(r_{i-1}) = g_i(0) + g_i(1)$, sends r_i
- Round d: Prover sends $g_d(x_d) = f^k(r_1, \ldots r_{d-1}, x_d)$
  Verifier checks $g_d(r_d) = f^k(r_1, r_2, \ldots r_d)$



Multi-Round Frequency Moments

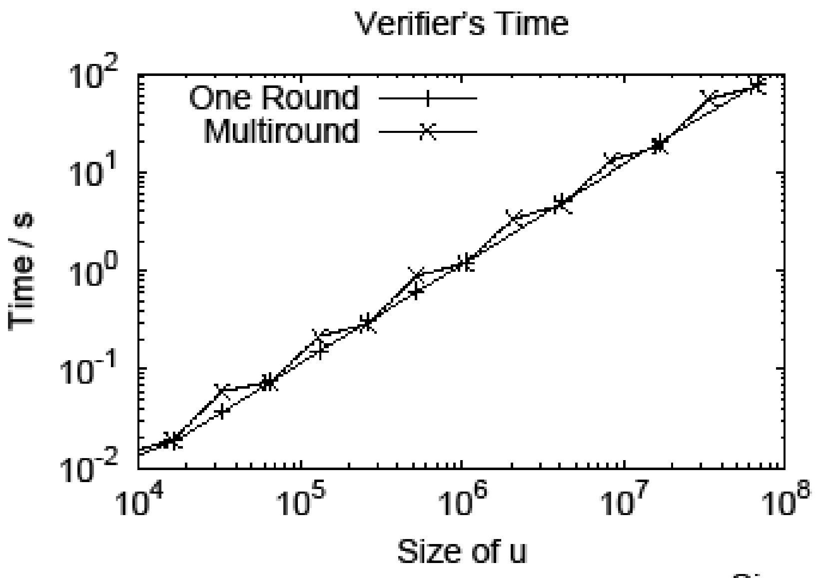
- Correctness: helper can't cheat last round without knowing r_d
- Then can't cheat round i without knowing r_i…
  - Similar to protocols from "traditional" Interactive Proofs
- Inductive proof, conditioned on each later round succeeding

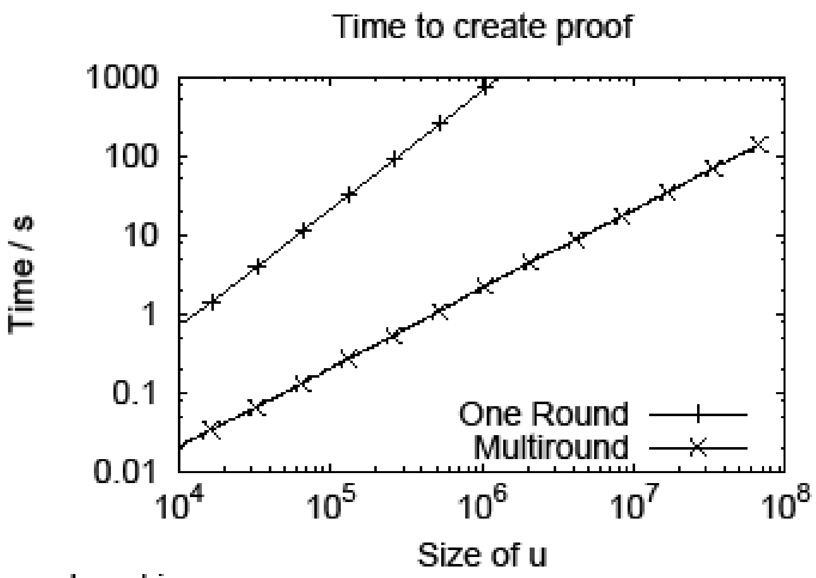
- Bounds: $O(k^2 \log N)$ total communication, $O(k \log N)$ space
- V's incremental computation possible in small space, via

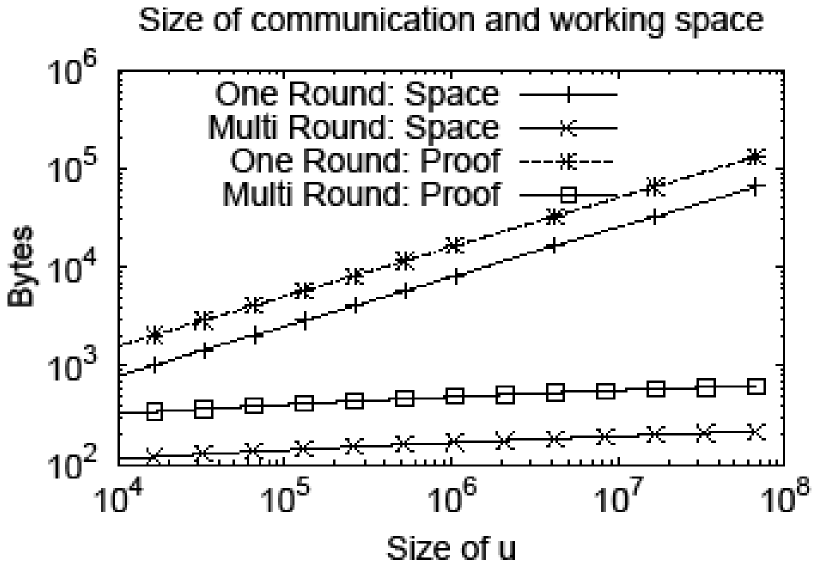
$$\prod_{j=1}^{d} (r_j + \mathrm{bit}(j,i)(1-2r_j))$$

- Intermediate polynomials relatively cheap for helper to find

Experimental Results



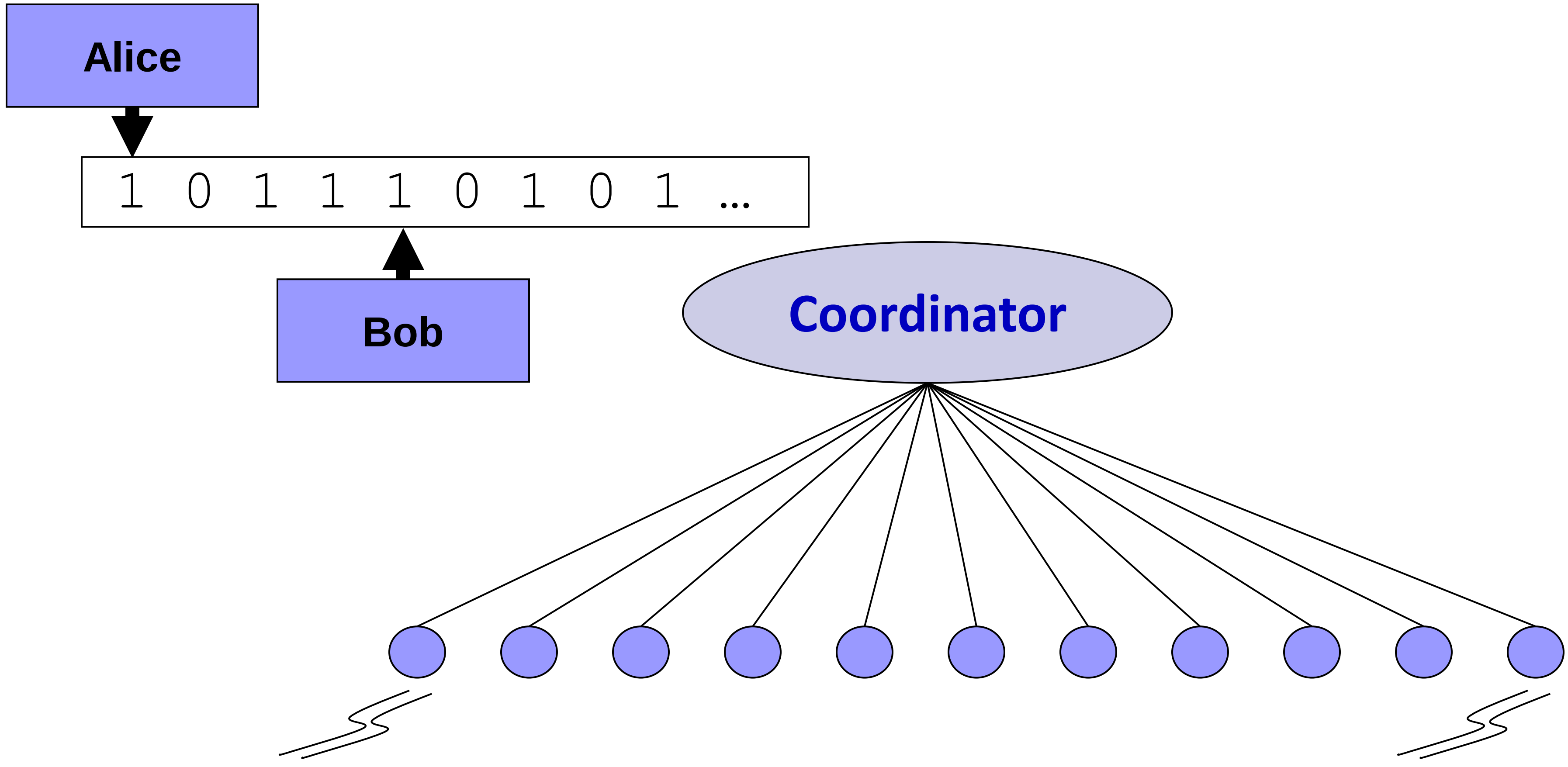




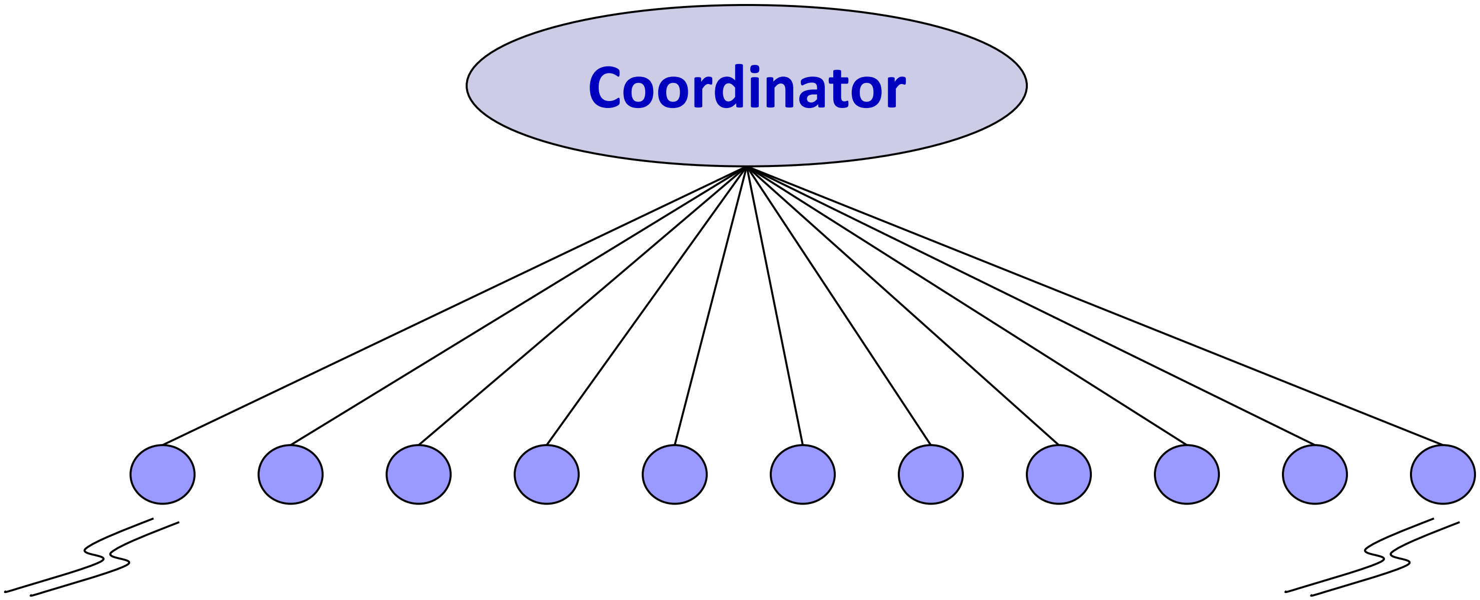
Extensions

- Distributed/parallel versions of these protocols
  - Can compute proofs in MapReduce, multicore and GPU models
- Lower bounds for multi-round versions of the protocols
  - May need new communication complexity models
- Use these protocols
  - Protocols seem practical, but are they compelling?
  - For what problems are protocols most needed?

Continuous Distributed Monitoring and Lower Bounds



Continuous Distributed Monitoring



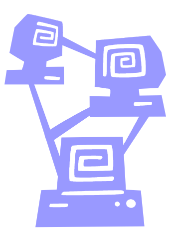
Distributed Monitoring

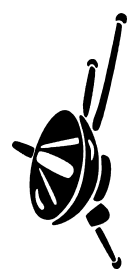
There are many scenarios where we need to track events:

- Network health monitoring within a large ISP
- Collecting and monitoring environmental data with sensors
- Observing usage and abuse of distributed data centers

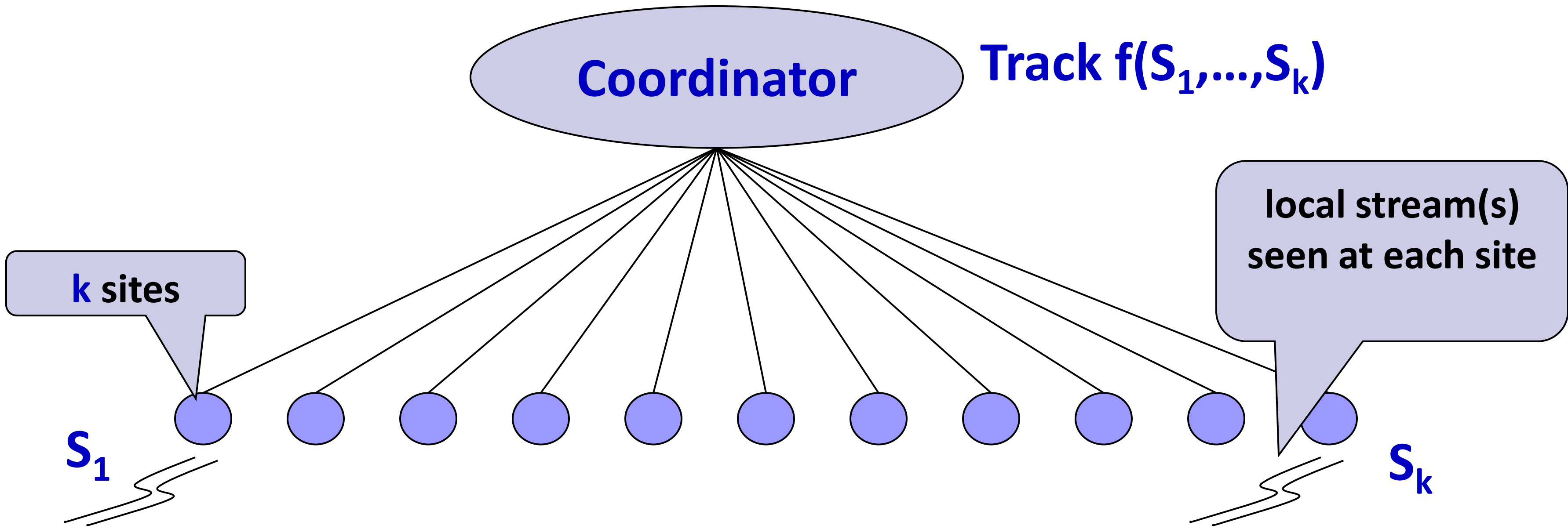
All can be abstracted as a collection of observers who want to collaborate to compute a function of their observations

From this we generate the Continuous Distributed Model





Continuous Distributed Model



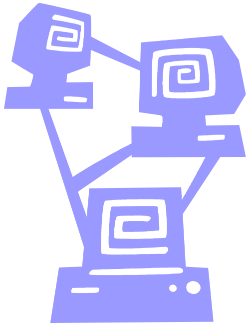
- Site-site communication only changes things by factor 2
- **Goal**: Coordinator *continuously tracks* (global) function of streams
  - Achieve communication poly(k,1/ε,log n)
  - Also bound space used by each site, time to process each update

Challenges

- Monitoring is Continuous…
  - Real-time tracking, rather than one-shot query/response
- …Distributed…
  - Each remote site only observes part of the global stream(s)
  - Communication constraints*:* must minimize monitoring burden
- …Streaming…
  - Each site sees a high-speed local data stream and can be resource (CPU/memory) constrained
- …Holistic…
  - Challenge is to monitor the complete global data distribution
  - Simple aggregates (e.g., aggregate traffic) are easier

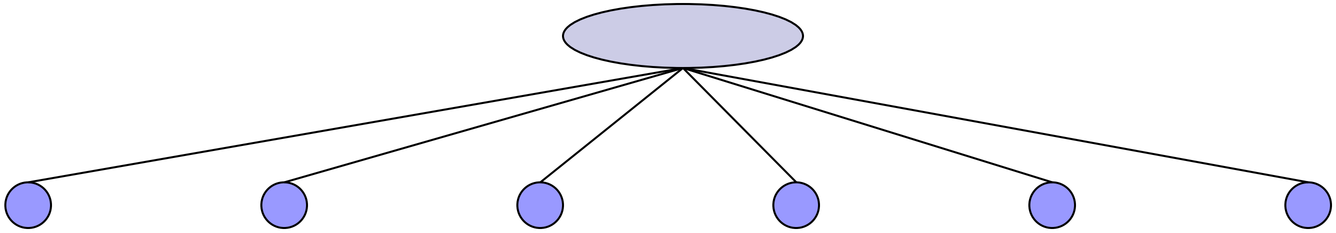
Baseline Approach

- Sometimes periodic polling suffices for simple tasks
  - E.g., SNMP polls total traffic at coarse granularity
- Still need to deal with holistic nature of aggregates
- Must balance polling frequency against communication
  - Very frequent polling causes high communication, excess battery use in wireless devices
  - Infrequent polling means delays in observing events
- Need techniques to reduce communication while guaranteeing rapid response to events



Variations in the model

- Multiple streams define the input A
- Given function f, several types of problem to study:
  - Threshold Monitoring: identify when $f(A) > \tau$
    Possibly tolerate some approximation based on $\varepsilon\tau$
  - Value Monitoring: always report accurate approximation of f(A)
  - Set Monitoring: f(A) is a set, always provide a “close” set
- Direct communication between sites and the coordinator
  - Other network structures possible (e.g., hierarchical)

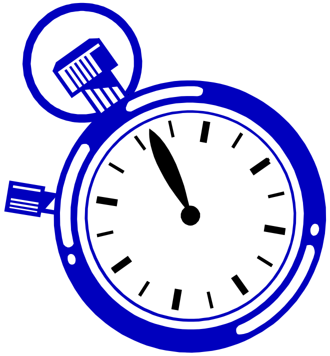


Outline

1. The Continuous Distributed Model
2. **How to count to 10**
3. The geometric approach
4. A sample of sampling
5. Prior work and future directions

The Countdown Problem

- A first abstract problem that has many applications
- Each observer sees events
- Want to alert when a total of τ events have been seen
  - Report when more than 10,000 vehicles have passed sensors
  - Identify the 1,000,000th customer at a chain of stores
- Trivial solution: send 1 bit for each event, coordinator counts
  - $O(\tau)$ communication
  - Can we do better?



A First Approach

- One of k sites must see τ/k events before threshold is met
- So each site counts events, sends message when τ/k are seen
- Coordinator collects current count n_i from each site
  - Compute new threshold $\tau' = \tau - \sum_{i=1}^{k} n_i$
  - Repeat procedure for τ' until $\tau' < k$, then count all events
- Analysis: $\tau > \tau'/(1-1/k) > \tau''/(1-1/k)^2 > \ldots$
  - Number of thresholds $= \log(\tau/k) / \log(1/(1-1/k)) = O(k \log(\tau/k))$
  - Total communication: $O(k^2 \log(\tau/k))$ [each update costs $O(k)$]
- Can we do better?

A Quadratic Improvement

- Observation: $O(k)$ communication per update is wasteful
- Try to wait for more updates before collecting
- Protocol operates over $\log(\tau/k)$ rounds [C.,Muthukrishnan, Yi 08]
  - In round j, each site waits to receive $\tau/(2^j k)$ events
  - Subtract this amount from local count n_i, and alert coordinator
  - Coordinator awaits k messages in round j, then starts round $j+1$
  - Coordinator informs all sites at end of each round
- Analysis: k messages in each round, $\log(\tau/k)$ rounds
  - Total communication is $O(k \log(\tau/k))$
  - Correct, since total count can't exceed τ until final round

Approximate variation

- Sometimes, we can tolerate approximation
- Only need to know if threshold τ is reached approximately
- So we can allow some bounded uncertainty:
  - Do not report when count < $(1-\varepsilon)\ \tau$
  - Definitely report when count > τ
  - In between, do not care
- Previous protocol adapts immediately:
  - Just wait until distance to threshold reaches $\varepsilon\tau$
  - Cost of the protocol reduces to $O(k \log 1/\varepsilon)$ (independent of τ)

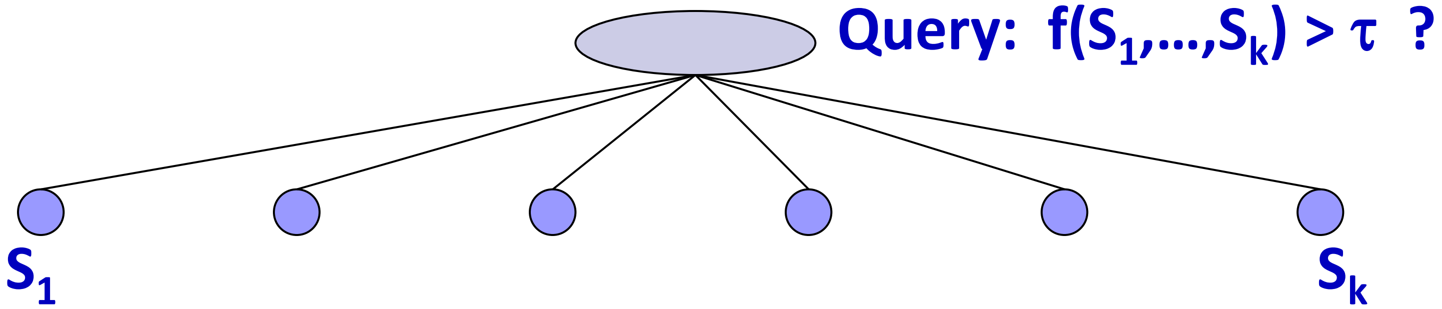
Extension: Randomized Solution

- Cost is high when k grows very large
- Randomization reduces this dependency, with parameter ε
- Now, each site waits to see $O(\varepsilon^2\tau/k)$ events
  - Roll a die: report with probability $1/k$, otherwise stay silent
  - Coordinator waits to receive $O(1/\varepsilon^2)$ reports, then terminates
- Analysis: in expectation, coordinator stops after $\tau(1-\varepsilon/2)$ events
  - With Chernoff bounds, show that it stops before τ events
  - And does not stop before $\tau(1-\varepsilon)$ events
- Gives a randomized, approximate solution: uncertainty of $\varepsilon\tau$

Outline

1. The Continuous Distributed Model
2. How to count to 10
3. **The geometric approach**
4. A sample of sampling
5. Prior work and future directions

General Non-linear Functions



- For general, non-linear f(), the problem becomes a lot harder!
  - E.g., information gain over global data distribution
- Non-trivial to decompose the global threshold into "safe" local site constraints
  - E.g., consider $N=(N_1+N_2)/2$ and $f(N) = 6N - N^2 > 1$
    Tricky to break into thresholds for $f(N_1)$ and $f(N_2)$

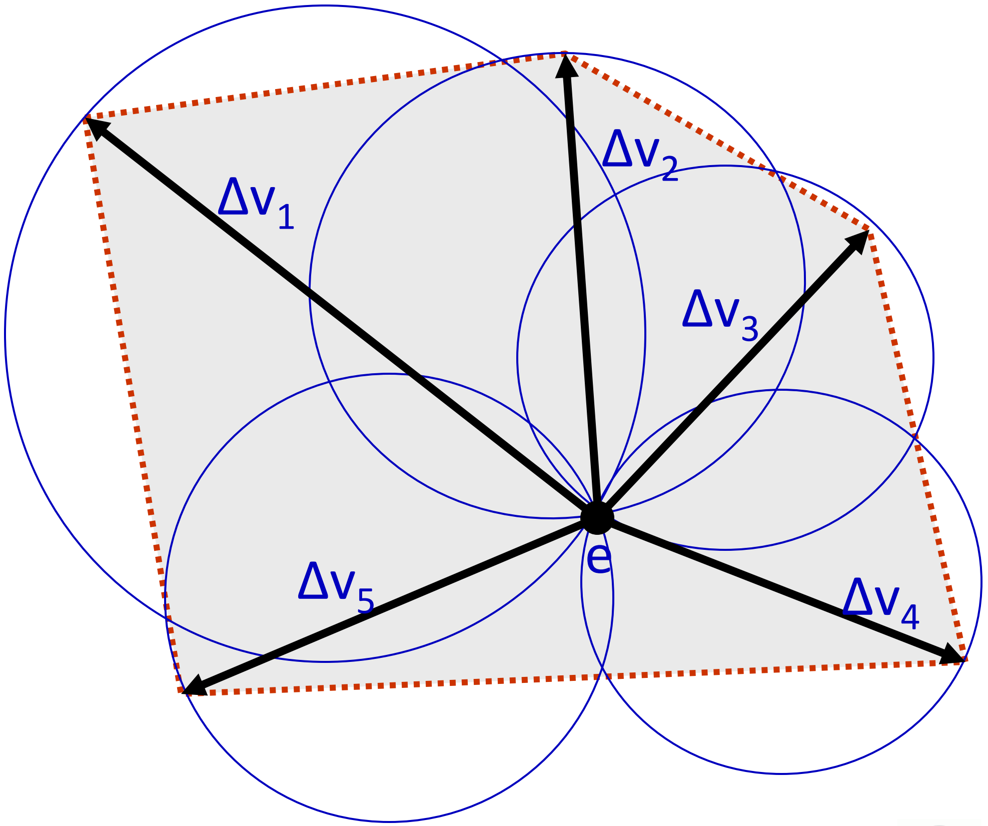
The Geometric Approach

- A general purpose geometric approach [Scharfman et al.'06]
  - Each site tracks a local statistics *vector* v_i (e.g., data distribution)
- Global condition is $f(v) > \tau$, where $v = \sum_i \lambda_i v_i \quad (\sum_i \lambda_i = 1)$
  - v = convex combination of local statistics vectors
- All sites share estimate $e = \sum_i \lambda_i v_i'$ of v

  based on latest update v_i' from site i
- Each site i tracks its drift from its most recent update $\Delta v_i = v_i - v_i'$

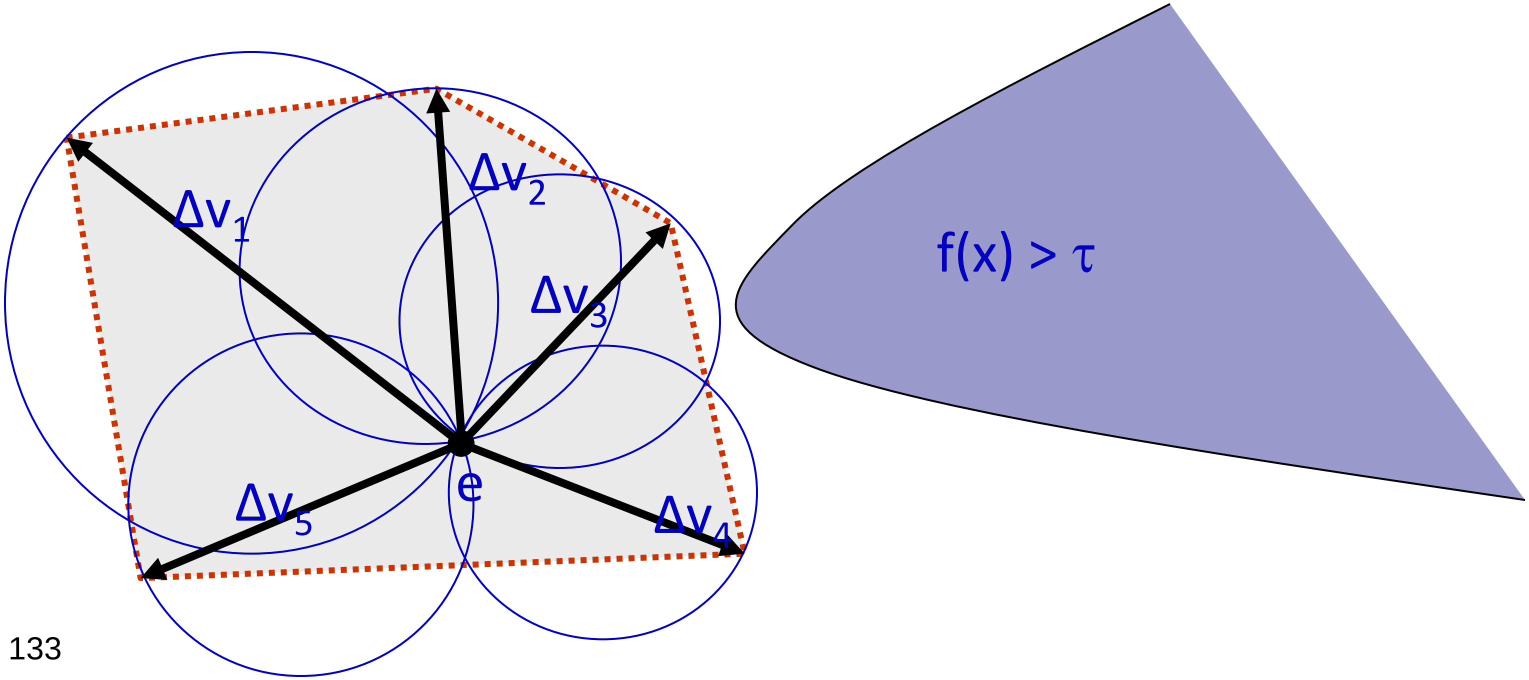
Covering the convex hull

- Key observation: $v = \sum_i \lambda_i \cdot (e + \Delta v_i)$
  (a convex combination of "translated" local drifts)
- v lies in the convex hull of the $(e+\Delta v_i)$ vectors
- Convex hull is completely covered by spheres with radii $\|\Delta v_i/2\|_2$ centered at $e+\Delta v_i/2$
- Each such sphere can be constructed independently



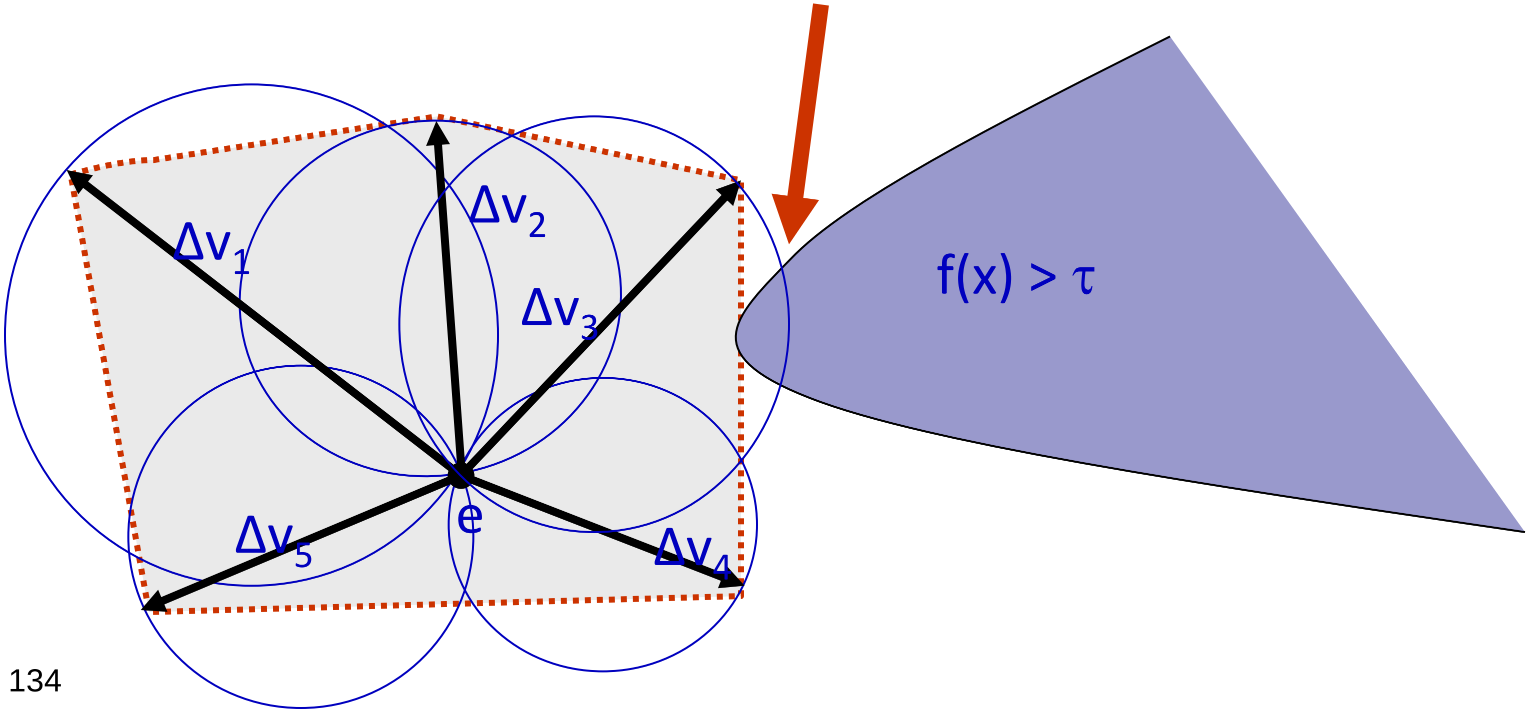
Monochromatic Regions

- Monochromatic Region: For all points x in the region $f(x)$ is on the same side of the threshold ($f(x) > \tau$ or $f(x) \leq \tau$)
- Each site independently checks its sphere is monochromatic
  - Find max and min for f() in local sphere region (may be costly)
  - Broadcast updated value of v_i if not monochrome



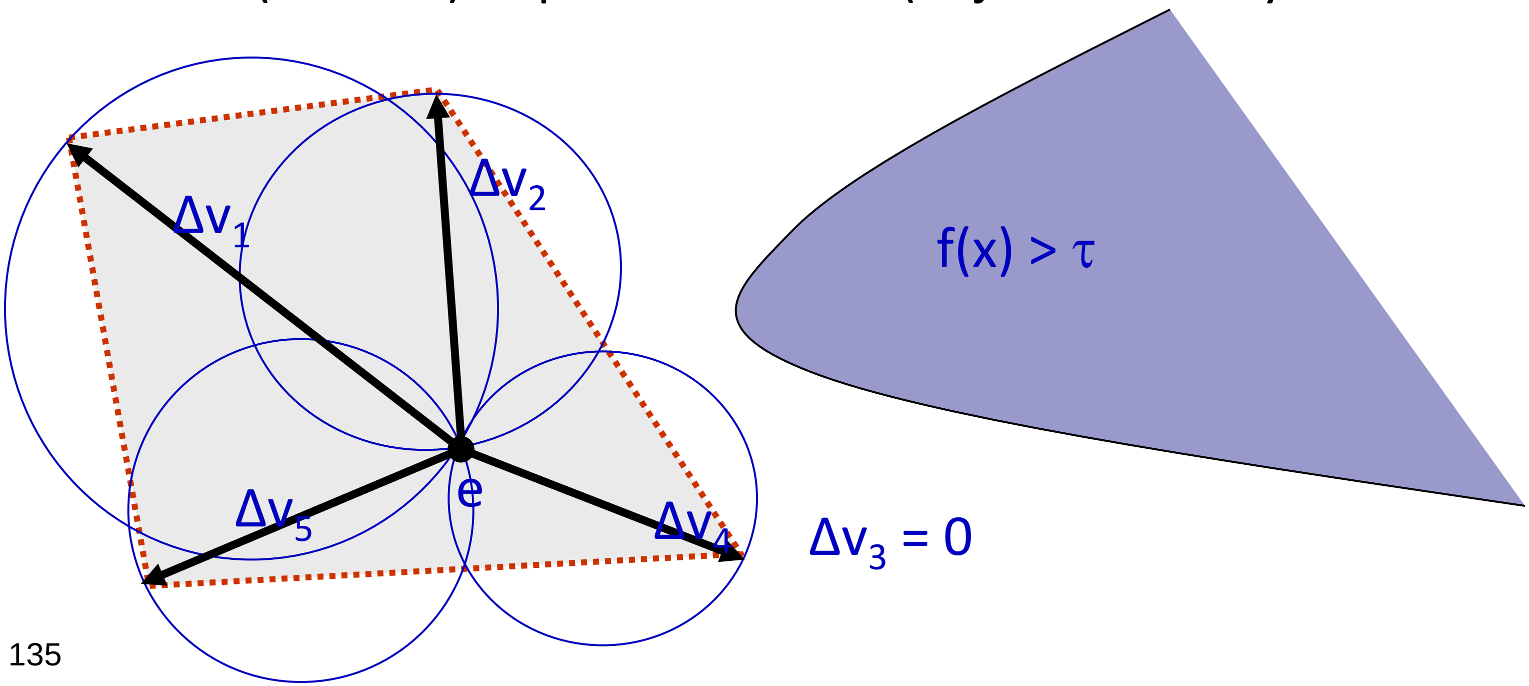
Restoring Monochomicity

- After broadcast, $\|\Delta v_i\|_2 = 0 \Rightarrow$ Sphere at i is monochromatic



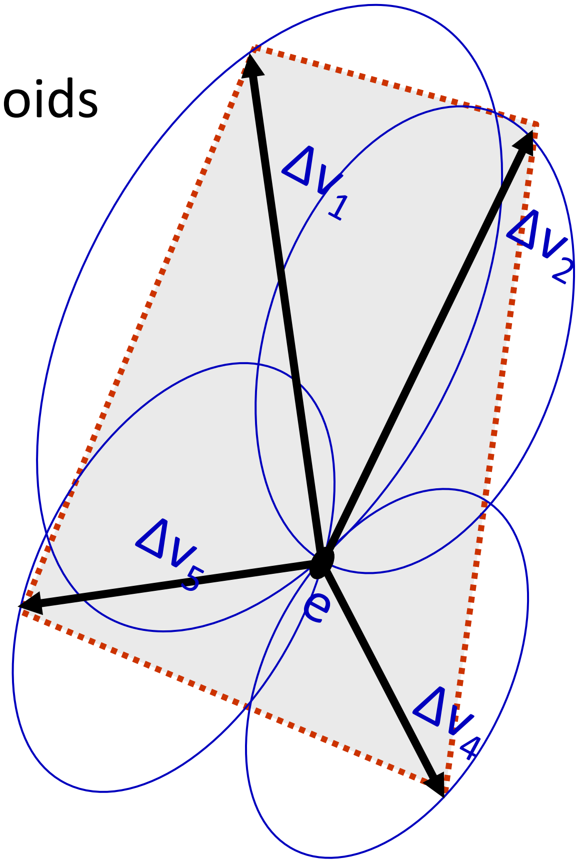
Restoring Monochomicity

- After broadcast, $\|\Delta v_i\|_2 = 0 \Rightarrow$ Sphere at i is monochromatic
  - Global estimate e is updated, which may cause more site update broadcasts
- Coordinator case*:* Can allocate local slack vectors to sites to enable "localized" resolutions
  - Drift (=radius) depends on slack (adjusted locally for subsets)



Extension: Transforms and Shifts

- Subsequent extensions further reduce cost [Scharfman et al. 10]
  - Same analysis of correctness holds when spheres are allowed to be ellipsoids
  - Additional offset vectors can be used to increase radius when close to threshold values
  - Combining these observations allows additional cost savings



Outline

1. The Continuous Distributed Model
2. How to count to 10
3. The geometric approach
4. **A sample of sampling**
5. Prior work and future directions

Drawing a Sample

- A basic 'set monitoring' problem is to draw a uniform sample
- Given inputs of total size N, draw a sample of size s
  - Uniform over all subsets of size s
- Overall approach:
  - Define a general sampling technique amenable to distribution
  - Bound the cost
  - Extend to sliding windows

Binary Bernoulli Sampling

- Always sample with probability $p = 2^{-i}$
- Randomly pick i bits, each of which is 0/1 with probability ½
- Select item if all i random bits are 0
- (Conceptually) store the random bits for each item
  - Can easily pick more random bits if the sampling rate decreases

Sampling Protocol

- Protocol based on [C., Muthukrishnan, Yi, Zhang 10]
- In round i, each site samples with $p = 2^{-i}$
  - Sampled items are sent to the coordinator
  - Coordinator picks one more random bit
  - End round i when coordinator has s items with $(i+1)$ zeros
  - Coordinator informs each site that a new round has started
  - Coordinator picks extra random bits for items in its sample

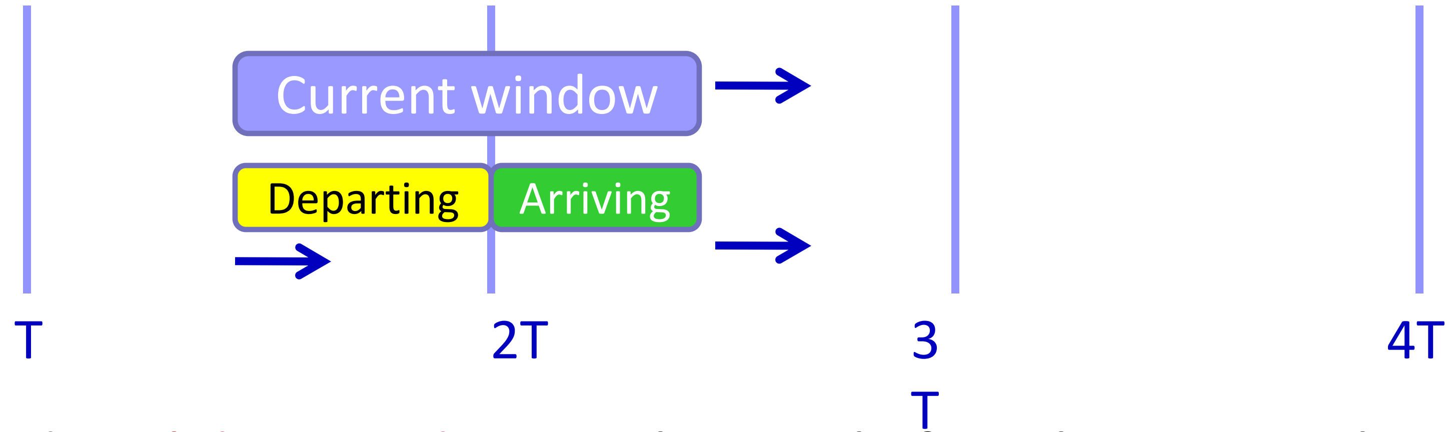
Protocol Costs

- Correctness: coordinator always has (at least) s items
  - Sampled with the same probability p
  - Can subsample to reach exactly s items
- Cost: each round is expected to send $O(s)$ items total
  - Can bound this with high probability via Chernoff bounds
  - Number of rounds is similarly bounded as $O(\log N)$
  - Communication cost is $O((k+s) \log N)$
- Lower bound on communication cost of $\Omega(k + s \log N)$
  - At least this many items are expected to appear in the sample
  - $O(k \log_{k/s} N + s \log N)$ upper bound by adjusting probabilities

Simplified Protocol

- Can simplify the protocol further [Tirthapura, Woodruff 11]:
  - Site j generates random tag u_i, sends if u_i < local threshold p_j
  - Coordinator receives sampled item, tests u_i against local p value
  - Set p so there are at most s items with tag less than p
  - Inform site of current p value, which updates p_j
- Prove correctness by matching to previous algorithm
  - Show simplified protocol never sends more than round-based

Extension: Sliding Window



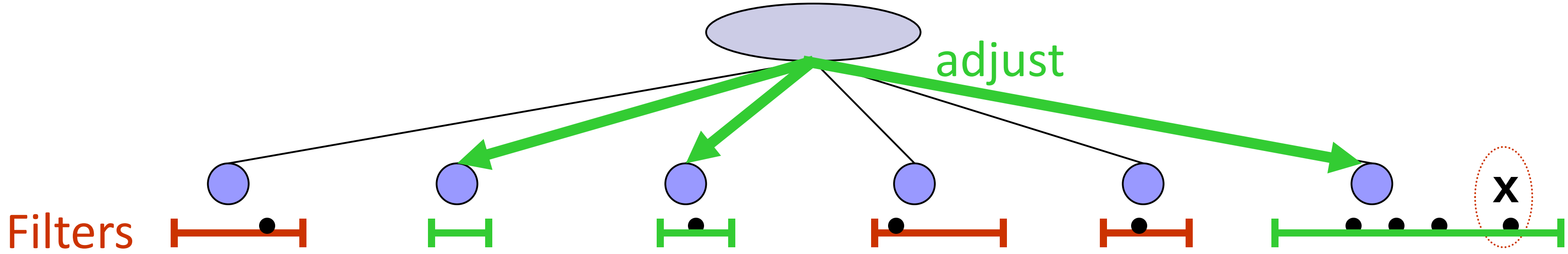
- Extend to sliding windows: only sample from last T arrivals
- Key insight: can break window into 'arriving' and 'departing'
  - Use multiple instances of Countdown protocol to track expiries
- Cost of such a protocol is $O(ks \log (W/s))$
  - Near-matching $\Omega(ks \log(W/ks))$ lower bound

Outline

1. The Continuous Distributed Model
2. How to count to 10
3. The geometric approach
4. A sample of sampling
5. **Prior work and future directions**

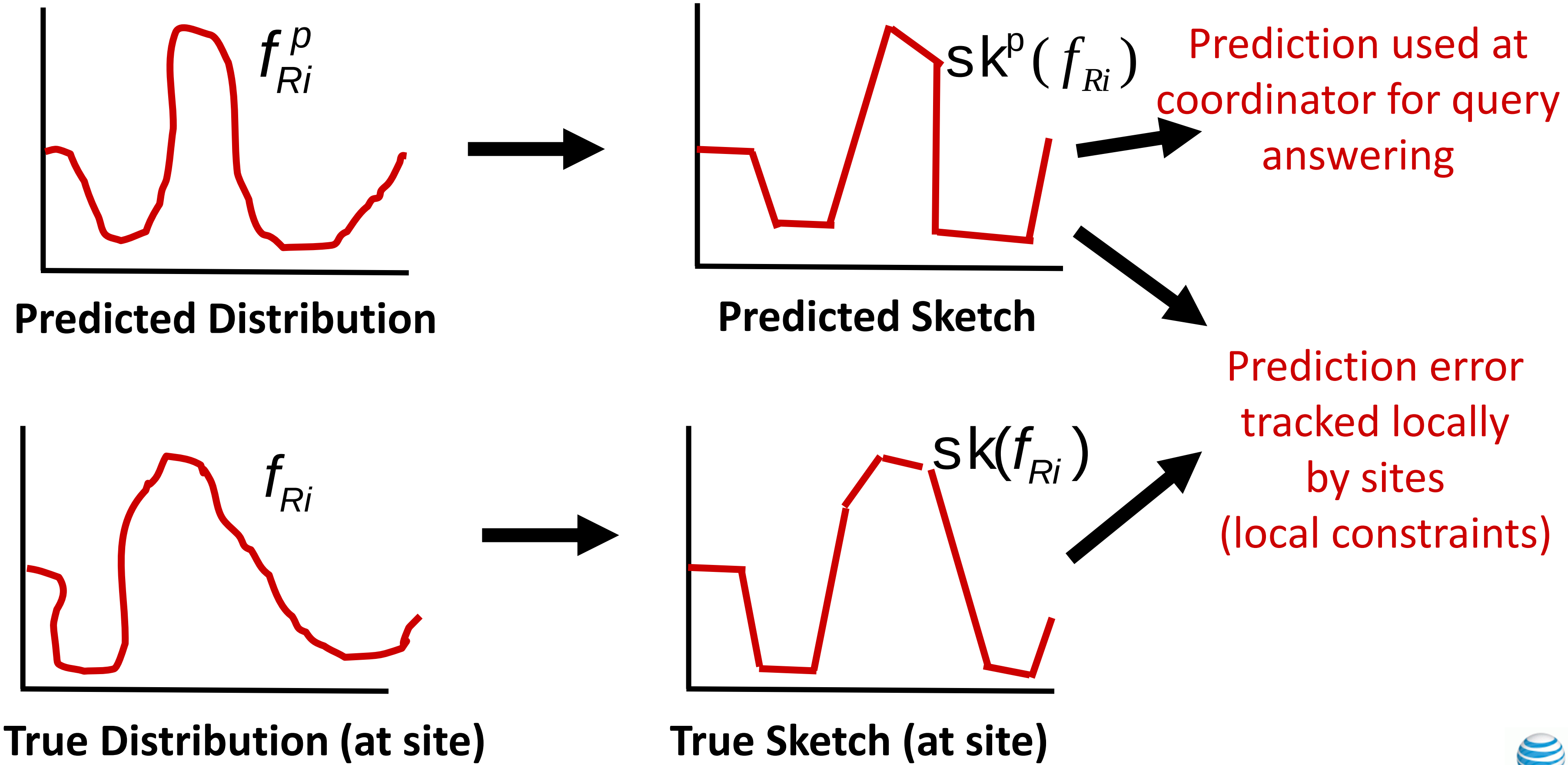
Early Work

- Continuous distributed monitoring arose in several places:
  - Networks: Reactive monitoring [Dilman Raz 01]
  - Databases: Distributed triggers [Jain et al. 04]
- Initial work on tracking multiple values
  - "Adaptive Filters" [Olston Jiang Widom 03]
  - Distributed top-k [Babcock Olston 03]



Prediction Models

- Prediction further reduces cost [C, Garofalakis, Muthukrishnan, Rastogi 05]
  - Combined with approximate (sketch) representations

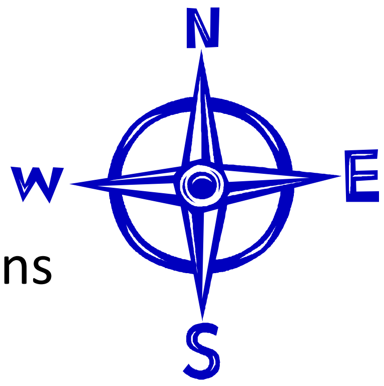


Problems in Distributed Monitoring

- Much interest in these problems in TCS and Database areas
- Many specific functions of (global) data distribution studied:
  - Set expressions [Das Ganguly Garofalakis Rastogi 04]
  - Quantiles and heavy hitters [C, Garofalakis, Muthukrishnan, Rastogi 05]
  - Number of distinct elements [C., Muthukrishnan, Zhuang 06]
  - Conditional Entropy [Arackaparambil, Bratus, Brody, Shubina 10]
  - Spectral properties of data matrix [Huang et al. 06]
  - Anomaly detection in networks [Huang et al. 07]
- Track functions only over sliding window of recent events
  - Samples [C, Muthukrishnan, Yi, Zhang 10]
  - Counts and frequencies [Chan Lam Lee Ting 10]

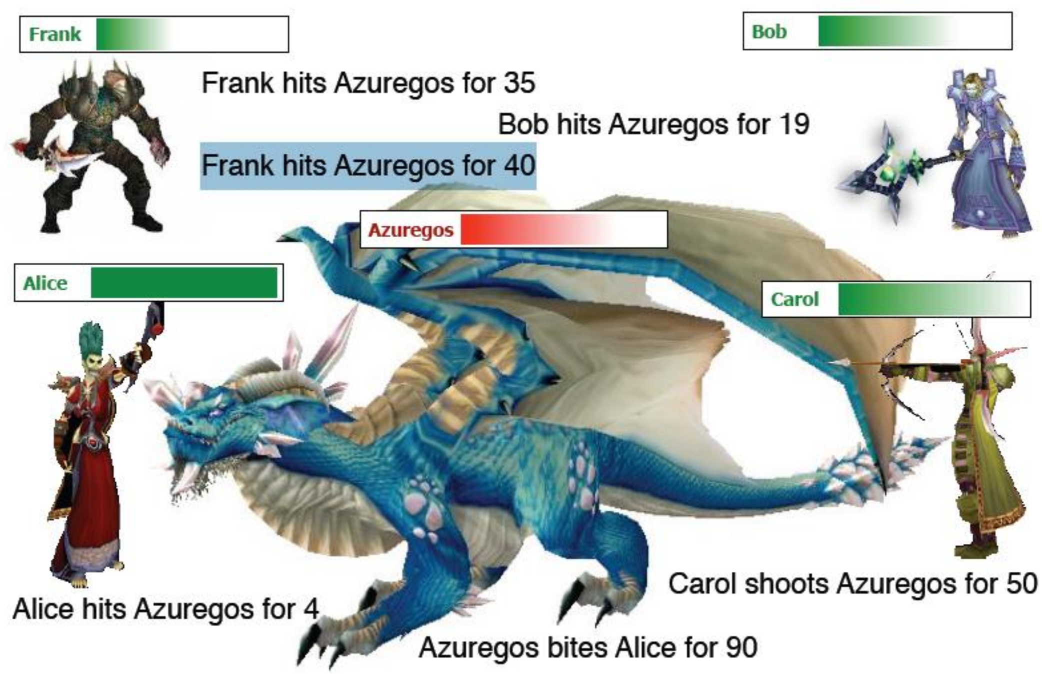
Other Work

- Many open problems remain in this area
  - Improve bounds for previously studied problems
  - Provide bounds for other important problems
  - Give general schemes for larger classes of functions
- Much ongoing work
  - See EU-support LIFT project, `lift-eu.org`
- Two specific open problems:
  - Develop systems and tools for continuous distributed monitoring
  - Provide a deeper theory for continuous distributed monitoring



Monitoring Systems

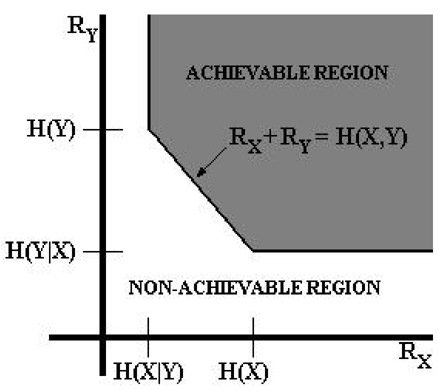
- Much theory developed, but less progress on deployment
- Some empirical study in the lab, with recorded data
- Still applications abound: Online Games [Heffner, Malecha 09]
  - Need to monitor many varying stats and bound communication
- Several
  - Buil ... ems
  - Evol ... buted DBMSs?)
- Several
  - Wha ... specific?
  - Wha ... onitoring?

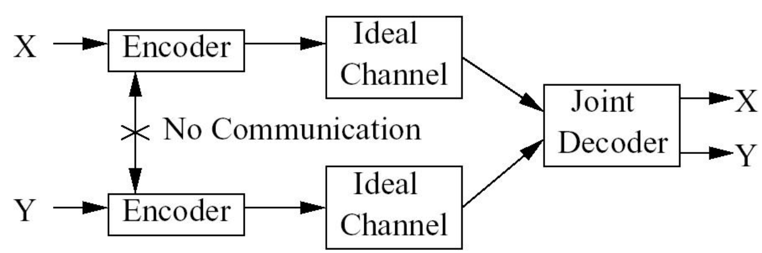


Theoretical Foundations

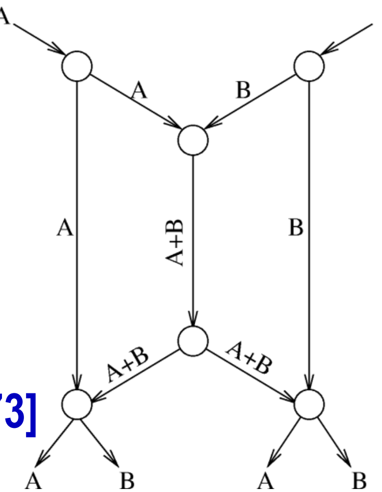
"Communication complexity" studies lower bounds of distributed one-shot computations

- Gives lower bounds for various problems, e.g., `count distinct` (via reduction to abstract problems)
- Need new theory for continuous computations
  - Based on info. theory and models of how streams evolve?
  - Link to distributed source coding or network coding?





Slepian-Wolf theorem [Slepian Wolf 1973]



http://www.networkcoding.info/

https://buffy.eecs.berkeley.edu/PHP/resabs/resabs.php?f_year=2005&f_submit=chapgrp&f_chapter=1

Distributed Monitoring Summary

- Continuous distributed monitoring is a natural model
- Captures many real world applications
- Much non-trivial work in this model
- Much work remains to do!
- Longer survey:
  `dimacs.rutgers.edu/~graham/pubs/papers/cdsurvey.pdf`

References (1)

[Babcock, Olston 03] B. Babcock and C. Olston. Distributed top-k monitoring. In ACM SIGMOD Intl. Conf. Management of Data, 2003.

[Chan Lam Lee Ting 10] H.-L. Chan, T.-W. Lam, L.-K. Lee, and H.-F. Ting. Continuous monitoring of distributed data streams over a time-based sliding window. In Symp. Theoretical Aspects of Computer Science, 2010.

[Cormode, Garofalakis '05] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In Proceedings of the International Conference on Very Large Data Bases, 2005.

[Cormode Garofalakis, Muthukrishnan Rastogi 05] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2005.

[C., Muthukrishnan, Zhuang 06] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate resilient aggregates on data streams. In IEEE Intl. Conf. Data Engineering, 2006.

[Cormode, Muthukrishnan, Yi 08] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed, functional monitoring. In ACM-SIAM Symp. Discrete Algorithms, 2008.

References (2)

[Cormode, Muthukrishnan, Yi, Zhang, 10] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang. Optimal sampling from distributed streams. In ACM Principles of Database Systems, 2010.

[Das Ganguly Garofalakis Rastogi 04] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed Set-Expression Cardinality Estimation. In Proceedings of VLDB, 2004.

[Dilman, Raz 01] M. Dilman, D. Raz. Efficient Reactive Monitoring. In IEEE Infocom, 2001.

[Heffner, Malecha 09] K. Heffner and G. Malecha. Design and implementation of generalized functional monitoring. www.people.fas.harvard.edu/~gmalecha/proj/funkymon.pdf, 2009.

[Huang et al. 06] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft. Distributed PCA and Network Anomaly Detection. In NIPS, 2006.

[Huang et al. 07] L. Huang, M. N. Garofalakis, A. D. Joseph, and N. Taft. Communication-efficient tracking of distributed cumulative triggers. In ICDCS, 2007.

[Jain et al. 04] A. Jain, J.M.Hellerstein, S. Ratnasamy, D. Wetherall. A Wakeup Call for Internet Monitoring Systems: The Case for Distributed Triggers. In Proceedings of HotNets-III, 2004.

[Kerlapura et al. 06] R. Kerlapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In ACM SIGMOD, 2006.

References (3)

[Olston, Jiang, Widom 03] C. Olston, J. Jiang, J. Widom. Adaptive Filters for Continuous Queries over Distributed Data Streams. In ACM SIGMOD, 2003.

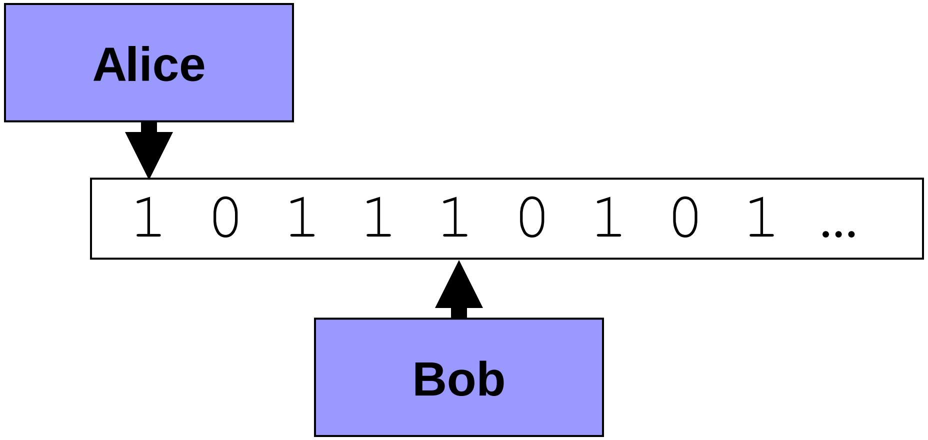
[Sharfman et al. 06] I. Sharfman, A. Schuster, D. Keren: A geometric approach to monitoring threshold functions over distributed data streams. SIGMOD Conference 2006: 301-312

[Sharfman et al. 10] I. Sharfman, A. Schuster, and D. Keren. Shape-sensitive geometric monitoring. In ACM Principles of Database Systems, 2010.

[Slepian, Wolf 73] D. Slepian, J. Wolf. Noiseless coding of correlated information sources. IEEE Transactions on Information Theory, 19(4):471-480, July 1973.

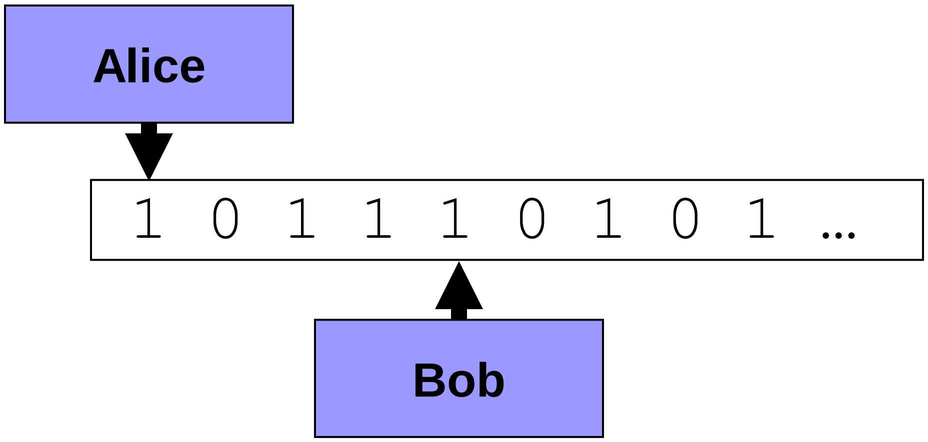
Distributed Streaming

Lower Bounds



Streaming Lower Bounds

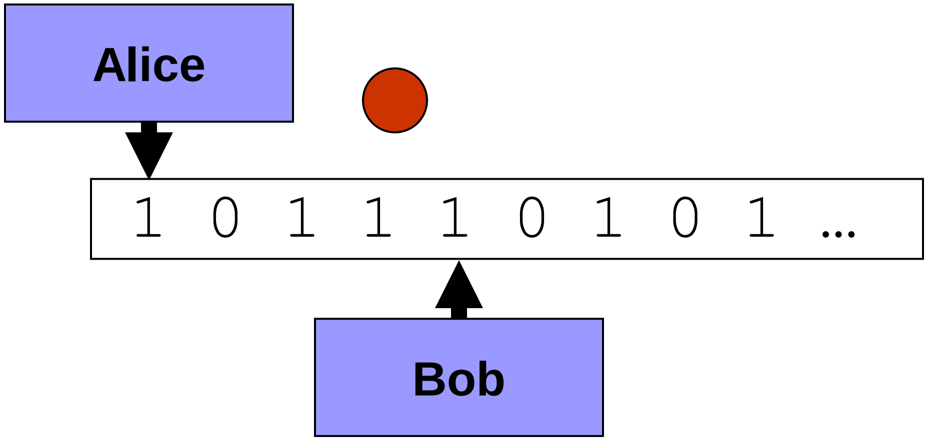
- Lower bounds for data streams
  - Communication and information complexity bounds
  - Simple reductions
  - Hardness of **`Gap-Hamming`** problem
  - Reductions to **`Gap-Hamming`**



Lower Bounds

- So far, have seen many examples of things we can do in streaming models
- What about things we can't do?
- What's the best we could achieve for things we can do?
- Will show lower bounds for (distributed) data streams based on communication complexity

Streaming As Communication

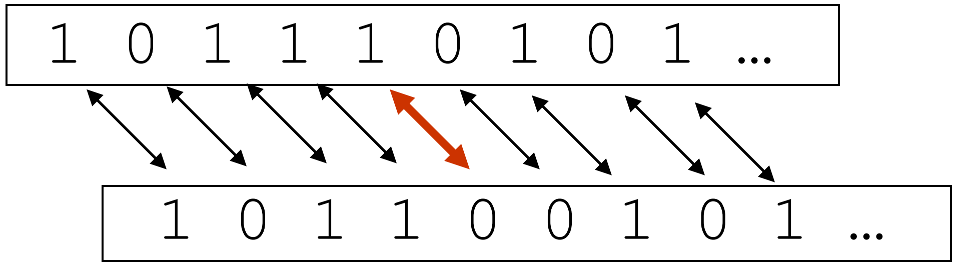


- Imagine Alice processing a single stream
- Then take the whole working memory, and send to Bob
- Bob continues processing the remainder of the stream

Streaming As Communication

- Suppose Alice's part of the stream corresponds to string x, and Bob's part corresponds to string y...
- ...and that computing the function on the stream corresponds to computing $f(x,y)$...
- ...then if $f(x,y)$ has communication complexity $\Omega(g(n))$, then the streaming computation has a *space lower bound* of $\Omega(g(n))$
- Proof by contradiction:
  If there was an algorithm with better space usage, we could run it on x, then send the memory contents as a message, and hence solve the communication problem

Deterministic Equality Testing



- Alice has string x, Bob has string y, want to test if x=y
- Consider a deterministic (one-round, one-way) protocol that sends a message of length $m < n$
- There are 2^m possible messages, so some strings must generate the same message: this would cause error
- So a deterministic message (sketch) must be $\Omega(n)$ bits
  - In contrast, we saw a randomized sketch of size $O(\log n)$

Hard Communication Problems

- **INDEX**: x is a binary string of length n
  y is an index in $[n]$
  Goal: output $x[y]$
  Result: (one-way) (randomized) communication complexity of **INDEX** is $\Omega(n)$ bits

- **DISJ**: x and y are both length n binary strings
  Goal: Output 1 if $\exists i$: $x[i]=y[i]=1$, else 0
  Result: (multi-round) (randomized) communication complexity of **DISJ** (disjointness) is $\Omega(n)$ bits

Hardness of INDEX

- Show hardness of **`INDEX`** via Information Complexity argument
  - Makes extensive use of Information Theory
- Entropy of random variable X: $H(X) = - \sum_x \Pr[X=x] \lg \Pr[X=x]$
  - (Expected) information (in bits) gained by learning value of X
  - If X takes on at most N values, $H(X) \leq \lg N$
- Conditional Entropy of X given Y: $H(X|Y) = \sum_y \Pr[y] H[X|Y=y]$
  - (Expected) information (bits) gained by learning value of X given Y
- Mutual Information: $I(X : Y) = I(Y : X) = H(X) – H(X \mid Y)$
  - Information (in bits) shared by X and Y
  - If X, Y are independent, $I(X : Y) = 0$ and $I(XY : Z) \geq I(X : Z) + I(Y : Z)$

Information Cost

- Use Information Theoretic properties to lower bound communication complexity
- Suppose Alice and Bob have random inputs X and Y
- Let M be the (random) message sent by Alice in protocol P
- The cost of (one-way) protocol P is $\text{cost}(P) = \max |M|$
  - Worst-case size of message (in bits) sent in the protocol
- Define information cost as $\text{icost}(P) = I(M : X)$
  - The information conveyed about X in M
  - $\text{icost}(P) = I(M : X) = H(M) - H(M \mid X) \leq H(M) \leq \text{cost}(P)$

Information Cost of INDEX

- Give Alice random input X = n uniform random bits
- Given protocol P for **INDEX**, Alice sends message $M(X)$
- Give Bob input i. He should output X_i
- $\text{icost}(P) = I(X_1 X_2 \ldots X_n : M)$
  $\geq I(X_1 : M) + I(X_2 : M) + \ldots + I(X_n : M)$
- Now consider the mutual information of X_i and M
  - Have reduced the problem to n instances of a simpler problem

Fano's Inequality

- When forming estimate X' from X given (message) M, where X, X' have k possible values, let E denote $X \neq X'$. We have:

  $$H(E) + \cancel{Pr[E] \log(k-1)} \geq H(X \mid M)$$

  where $H(E) = -Pr[E] \lg Pr[E] - (1-Pr[E]) \lg(1-Pr[E])$
- Here, $k=2$, so we get $I(X : M) = H(X) - H(X \mid M) \geq H(X) - H(E)$
  - $H(X) = 1$. If $Pr[E]=\delta$, we have $H(E) < ½$ for $\delta<0.1$
  - Hence $I(X_i : M) > ½$
- Thus $cost(P) \geq icost(P) > ½ n$ if P succeeds w/prob $1-\delta$
  - Protocols for **`INDEX`** must send $\Omega(n)$ bits

Outline for DISJOINTNESS hardness

- Hardness for **DISJ** follows a similar outline
- Reduce to n instances of the problem "**AND**"
  - "**AND**" problem: test whether $X_i = Y_i = 1$
- Show that the information cost of **DISJ** protocol is sufficient to solve all n instances of **AND**
- Show that the information cost of each instance is $\Omega(1)$
- Proves that communication cost of **DISJ** is $\Omega(1)$
  - Even allowing multiple rounds of communication

at&t
Your world. Delivered.

Simple Reduction to Disjointness

x: 1 0 1 1 0 1 $\longrightarrow$ 1, 3, 4, 6

y: 0 0 0 1 1 0 $\longrightarrow$ 4, 5

- F_∞: output the highest frequency in a stream
- Input: the two strings x and y from disjointness instance
- Stream: if x[i]=1, then put i in stream; then same for y
  - A streaming reduction (compare to polynomial-time reductions)
- Analysis: if $F_\infty=2$, then intersection; if $F_\infty \leq 1$, then disjoint.
- Conclusion: Giving exact answer to F_∞ requires $\Omega(N)$ bits
  - Even approximating up to 50% relative error is hard
  - Even with randomization: **DISJ** bound allows randomness

Simple Reduction to Index

x: 1 0 1 1 0 1 $\longrightarrow$ 1, 3, 4, 6

y: 5 $\longrightarrow$ 5

- F_0: output the number of items in the stream
- Input: the strings x and index y from **INDEX**
- Stream: if x[i]=1, put i in stream; then put y in stream
- Analysis: if $(1-\varepsilon)F'_0(x \cup y) > (1+\varepsilon)F'_0(x)$ then x[y]=1, else it is 0
- Conclusion: Approximating F_0 for $\varepsilon < 1/N$ requires $\Omega(N)$ bits
  - Implies that space to approximate must be $\Omega(1/\varepsilon)$
  - Bound allows randomization

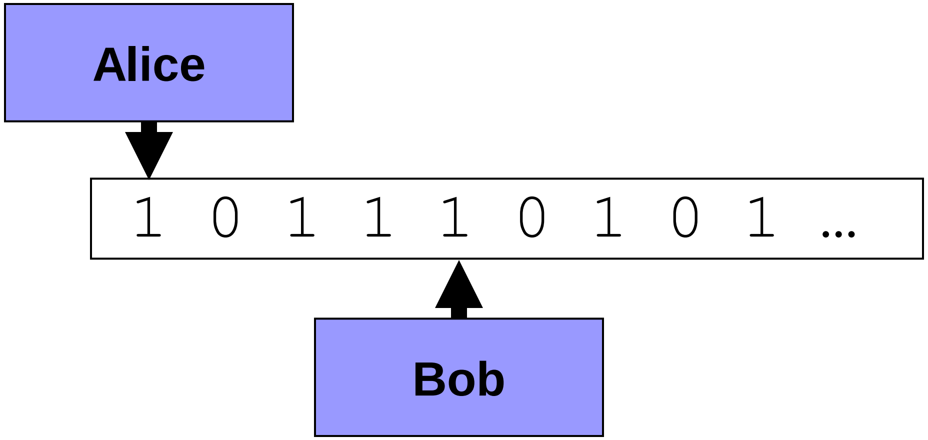
Hardness Reduction Exercises

Use reductions to **DISJ** or **INDEX** to show the hardness of:

1. Frequent items: find all items in the stream whose frequency $> \phi N$, for some ϕ.
2. Sliding window: given a stream of binary (0/1) values, compute the sum of the last N values
   – Can this be approximated instead?
3. Min-dominance: given a stream of pairs $(i,x(i))$, approximate $\sum_i \min_{(i,\, x(i))} x(i)$
4. Rank sum: Given a stream of (x,y) pairs and query (p,q) specified after stream, approximate $|\{(x,y)|\ x<p,\ y<q\}|$

Streaming Lower Bounds

- Lower bounds for data streams
  - Communication complexity bounds
  - Simple reductions
  - Hardness of **`Gap-Hamming`** problem
  - Reductions to **`Gap-Hamming`**



Gap Hamming

Gap-Hamming communication problem:

- Alice holds $x \in \{0,1\}^N$, Bob holds $y \in \{0,1\}^N$
- Promise: $Ham(x,y)$ is either $\leq N/2 - \sqrt{N}$ or $\geq N/2 + \sqrt{N}$
- Which is the case?
- Model: one message from Alice to Bob
- Sketching upper bound: need relative error $\varepsilon = \sqrt{N}/F_2 = 1/\sqrt{N}$
  - Gives space $O(1/\varepsilon^2) = O(N)$

Requires $\Omega(N)$ bits of one-way randomized communication

[Indyk, Woodruff'03, Woodruff'04, Jayram, Kumar, Sivakumar '07]

Hardness of Gap Hamming

- Reduction starts with an instance of **INDEX**
  - Map string x to u by $1 \to +1$, $0 \to -1$ (i.e. $u[i] = 2x[i] - 1$)
  - Assume both Alice and Bob have access to public random strings r_j, where each bit of r_j is iid $\{-1, +1\}$
  - Assume w.l.o.g. that length of string n is odd (important!)
  - Alice computes $a_j = \text{sign}(r_j \cdot u)$
  - Bob computes $b_j = \text{sign}(r_j[y])$
- Repeat N times with different random strings, and consider the Hamming distance of $a_1 \ldots a_N$ with $b_1 \ldots b_N$
  - Argue if we solve **Gap-Hamming** on (a, b), we solve **INDEX**

Probability of a Hamming Error

- Consider the pair $a_j = \text{sign}(r_j \cdot u)$, $b_j = \text{sign}(r_j[y])$
- Let $w = \sum_{i \neq y} u[i]\, r_j[i]$
  - w is a sum of $(n-1)$ values distributed iid uniform $\{-1,+1\}$
- Case 1: $w \neq 0$. So $|w| \geq 2$, since $(n-1)$ is even
  - so $\text{sign}(a_j) = \text{sign}(w)$, independent of $x[y]$
  - Then $\Pr[a_j \neq b_j] = \Pr[\text{sign}(w) \neq \text{sign}(r_j[y])] = ½$
- Case 2: $w = 0$.

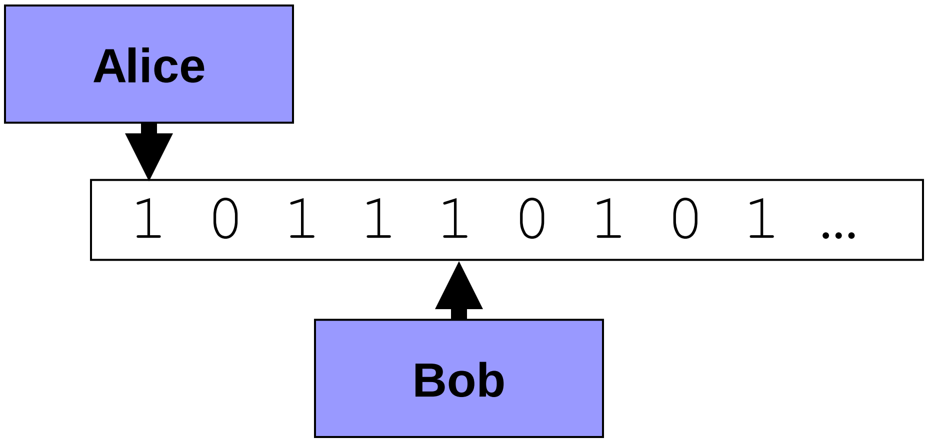
  So $a_j = \text{sign}(r_j \cdot u) = \text{sign}(w + u[y]r_j[y]) = \text{sign}(u[y]r_j[y])$
  - Then $\Pr[a_j \neq b_j] = \Pr[\text{sign}(u[y]r_j[y]) = \text{sign}(r_j[y])]$
  - This probability is 1 is $u[y]=+1$, 0 if $u[y]=-1$
  - Completely biased by the answer to **INDEX**

Finishing the Reduction

- So what is $Pr[w=0]$?
  - w is sum of $(n-1)$ iid uniform $\{-1,+1\}$ values
  - Exercise: $Pr[w=0] = 2^{-n}(n \text{ choose } n/2) = c/\sqrt{n}$, for some constant c
- Do some probability manipulation:
  - $Pr[a_j = b_j] = ½ + c/2\sqrt{n}$ if $x[y]=1$
  - $Pr[a_j = b_j] = ½ - c/2\sqrt{n}$ if $x[y]=0$
- Amplify this bias by making strings of length $N=4n/c^2$
  - Apply Chernoff bound on N instances
  - With prob>$2/3$, either $Ham(a,b)>N/2 + \sqrt{N}$ or $Ham(a,b)<N/2 - \sqrt{N}$
- If we could solve **Gap-Hamming**, could solve **INDEX**
  - Therefore, need $\Omega(N) = \Omega(n)$ bits for **Gap-Hamming**

Streaming Lower Bounds

- Lower bounds for data streams
  - Communication complexity bounds
  - Simple reductions
  - Hardness of **`Gap-Hamming`** problem
  - Reductions to **`Gap-Hamming`**



Lower Bound for Entropy

`Gap-Hamming` instance—Alice: $x \in \{0,1\}^N$, Bob: $y \in \{0,1\}^N$

Entropy estimation algorithm **`A`**

- Alice runs **`A`** on $\mathrm{enc}(x) = \langle(1,x_1), (2,x_2), \ldots, (N,x_N)\rangle$
- Alice sends over memory contents to Bob
- Bob continues **`A`** on $\mathrm{enc}(y) = \langle(1,y_1), (2,y_2), \ldots, (N,y_N)\rangle$

Alice	0	1	0	0	1	1
	(1,0)	(2,1)	(3,0)	(4,0)	(5,1)	(6,1)
Bob	(1,1)	(2,1)	(3,0)	(4,0)	(5,1)	(6,0)
	1	1	0	0	1	0

Lower Bound for Entropy

- Observe: there are
  - $2\text{Ham}(x,y)$ tokens with frequency 1 each
  - $N-\text{Ham}(x,y)$ tokens with frequency 2 each
- So (after algebra), $H(S) = \log N + \text{Ham}(x,y)/N = \log N + ½ \pm 1/\sqrt{N}$
- If we separate two cases, size of Alice's memory contents = $\Omega(N)$
  Set $\varepsilon = 1/(\sqrt{(N)} \log N)$ to show bound of $\Omega(\varepsilon/\log 1/\varepsilon)^{-2})$

	0	1	0	0	1	1
Alice	(1,0)	(2,1)	(3,0)	(4,0)	(5,1)	(6,1)
	(1,1)	(2,1)	(3,0)	(4,0)	(5,1)	(6,0)
Bob	1	1	0	0	1	0

Lower Bound for F_0

- Same encoding works for F_0 (Distinct Elements)
  - $2\mathrm{Ham}(x,y)$ tokens with frequency 1 each
  - $N-\mathrm{Ham}(x,y)$ tokens with frequency 2 each
- $F_0(S) = N + \mathrm{Ham}(x,y)$
- Either $\mathrm{Ham}(x,y)>N/2 + \sqrt{N}$ or $\mathrm{Ham}(x,y)<N/2 - \sqrt{N}$
  - If we could approximate F_0 with $\varepsilon < 1/\sqrt{N}$, could separate
  - But space bound $= \Omega(N) = \Omega(\varepsilon^{-2})$ bits
- Dependence on ε for F_0 is tight

- Similar arguments show $\Omega(\varepsilon^{-2})$ bounds for F_k
  - Proof assumes k (and hence 2^k) are constants

Lower Bounds Exercises

1. Formally argue the space lower bound for F_2 via Gap-Hamming
2. Argue space lower bounds for F_k via Gap-Hamming
3. (Research problem) Extend lower bounds for the case when the order of the stream is random or near-random

Other Streaming Directions

Many fundamentals have been studied, not time to cover here:

- Different streaming data types
  - Massive Matrices, Permutations, Graph Data, Geometric Data
- Different streaming processing models
  - Sliding Windows, Exponential and other decay, Random order streams, Skewed streams
- Different streaming scenarios
  - Gossip computations, sensor network computations, MapReduce computations
- Different streaming applications
  - Advanced mining algorithms, large scale machine learning