Streaming Graph Computations with a Helpful Advisor

Justin Thaler

Graham Cormode and

Michael Mitzenmacher

Thanks to Andrew McGregor

- For allowing borrowing and modification of slides from IITK Workshop on Algorithms for Processing Massive Data Sets.

Data Streaming Model

- Stream: m elements from universe of size n
  - e.g., $S=<x_1, x_2, \ldots, x_m> = 3,5,3,7,5,4,8,7,5,4,8,6,3,2, \ldots$

- Goal: Compute a function of stream, e.g., median, number of distinct elements, frequency moments, heavy hitters.

- Challenge:

  (i) Limited working memory, i.e., sublinear(n,m).

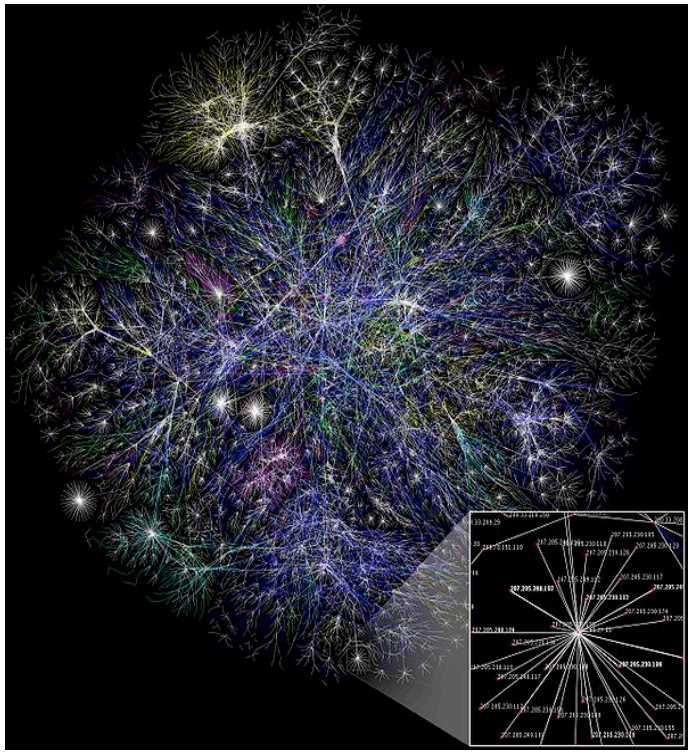
  (ii) Sequential access to adversarially ordered data.

  (iii) Process each update quickly.

Slide derived from [McGregor 10]

Graph Streams

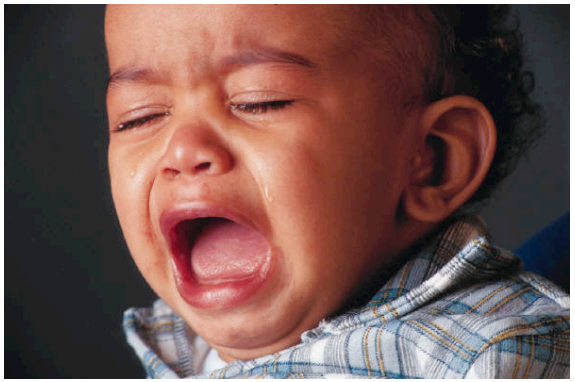
- $S = \langle x_1, x_2, \ldots, x_m \rangle$; $x_i \in [n] \times [n]$
- A defines a graph G on n vertices.
- Goal: compute properties of G.
- Challenge: subject to usual streaming constraints.



Snapshot of Internet Graph
Source: Wikipedia

Bad News

- Many graph problems are impossible in standard streaming model (require linear space or many passes over data).
- E.g. Connectivity, bipartiteness, counting triangles, diameter, perfect matching.
- What to do? Approximation (when applicable), or change the model.



Modified Models

- Semi-Streaming: Use O(n polylog(n)) space. Big improvement for dense graphs.
  - "Sweet spot" for graph algorithms: Connectivity, bipartiteness, etc.
- W-Stream: Algorithm can write temporary streams to aid computation. [Demetrescu et al. 06]
- Random-order, Sort-stream [Aggarwal et al. 04].

Modified Models II: Outsourcing

- Stream Punctuation [Tucker et al. 05], Proof Infused Streams [Li et al. 07], Stream Outsourcing [Yi et al. 08], Best-Order Model [Das Sarma et al. 09] (is a special case of our model)

Modified Models II: Outsourcing

- Stream Punctuation [Tucker et al. 05], Proof Infused Streams [Li et al. 07], Stream Outsourcing [Yi et al. 08], Best-Order Model [Das Sarma et al. 09] (is a special case of our model)

- [Chakrabarti et al. 09] Online Annotation Model: Give streaming algorithm access to powerful *helper* H who can annotate the stream.
  - Main motivation: Commercial cloud computing services such as Amazon EC2. Helper is untrusted.
  - Also, Volunteer Computing (SETI@home. Great Internet Mersenne Prime Search, etc.)
  - Weak peripheral devices.

Online Annotation Model

- **Problem**: Given stream S, want to compute f(S):

$S=<x_1, x_2, x_3, x_4, x_5, x_6, \ldots, x_m>$

Online Annotation Model

- **Problem**: Given stream S, want to compute f(S):

$S=<x_1, x_2, x_3, x_4, x_5, x_6, \ldots, x_m>$

- Helper H: augments stream with h-word annotation:

$(S,a)=<x_1, x_2, x_3, x_4, x_5, x_6, \ldots, x_m, a_1, a_2, \ldots, a_h>$

Online Annotation Model

- **Problem**: Given stream S, want to compute f(S):

$S=<x_1, x_2, x_3, x_4, x_5, x_6, \ldots, x_m>$

- Helper H: augments stream with h-word annotation:

$(S,a)=<x_1, x_2, x_3, x_4, x_5, x_6, \ldots, x_m, a_1, a_2, \ldots, a_h>$

- Verifier V: using v words of space and random string r, run verification algorithm to compute g(S,a,r) such that for all a either:

  a)$Pr_r[g(S,a,r) = f(S)]=1$ (we say a is valid for S) or

  b) $Pr_r[g(S,a,r) = \perp] \geq 1-\delta$ (we say a is δ-invalid for S)

  c) And at least one a is valid for S.

Note: this model differs slightly from [Chakrabarti et al. 09].

Online Annotation Model

- Two costs: words of annotation h and working memory v.
  - We refer to (h, v)-protocols.
  - Primarily interested in minimizing v.
  - But strive for optimal tradeoffs between h and v.
  - Proves more challenging for graph streams than numerical streams. Algebraic structure seems critical.

Fingerprinting

- Need a way to test multiset equality (e.g. to see if two streams have the same frequency distribution).
  - But need to do so in a streaming fashion.
  - We often use this to make sure H is "consistent".

Fingerprinting

- Need a way to test multiset equality (e.g. to see if two streams have the same frequency distribution).
  - But need to do so in a streaming fashion.
  - We often use this to make sure H is "consistent".
- Solution: fingerprints.
  - Hash functions that can be computed by a streaming verifier.
  - If S≠ S' as frequency distributions, then $f(S) \neq f(S')$ w.h.p.

Fingerprinting

- Need a way to test multiset equality (e.g. to see if two streams have the same frequency distribution).
  - But need to do so in a streaming fashion.
  - We often use this to make sure H is "consistent".
- Solution: fingerprints.
  - Hash functions that can be computed by a streaming verifier.
  - If $S \neq S'$ as frequency distributions, then $f(S) \neq f(S')$ w.h.p.
- We choose a fingerprint function f that is linear! $f(S \circ S') = f(S) + f(S')$ where $\circ$ denotes concatenation. Will need this for matrix-vector multiplication.

Two Approaches To Designing Protocols

1. Prove matching upper and lower bounds on a quantity.
   - One bound often easy: just give feasible solution.
   - Proving optimality more difficult. Usually requires problem structure.

2. Use H to "verify" execution of a non-streaming algorithm.

Max-Matching

- [Chakrabarti et al. 09]: (m, 1)-protocol for bipartite Perfect Matching. Also $hv = \Omega(n^2)$ lower bound.

Max-Matching

- [Chakrabarti et al. 09]: (m, 1)-protocol for bipartite Perfect Matching. Also $hv = \Omega(n^2)$ lower bound.
- We give (m, 1)-protocol for general max-cardinality matching.

Max-Matching

- [Chakrabarti et al. 09]: (m, 1)-protocol for bipartite Perfect Matching. Also $h_V = \Omega(n^2)$ lower bound.
- We give (m, 1)-protocol for general max-cardinality matching.
- **(Tutte-Berge Formula):** The size of a maximum matching of a graph $G = (V, E)$ equals

$$\tfrac{1}{2} \min_{U \subset V} (|U| - occ(G-U) + |V|)$$

where $occ(H)$ is the number of connected components in the graph H with an odd number of vertices.

Max-Matching

- [Chakrabarti et al. 09]: (m, 1)-protocol for bipartite Perfect Matching. Also $h_V = \Omega(n^2)$ lower bound.
- We give (m, 1)-protocol for general max-cardinality matching.
- **(Tutte-Berge Formula):** The size of a maximum matching of a graph $G = (V, E)$ equals

$$\tfrac{1}{2} \min_{U \subset V} (|U| - occ(G-U) + |V|)$$

where $occ(H)$ is the number of connected components in the graph H with an odd number of vertices.

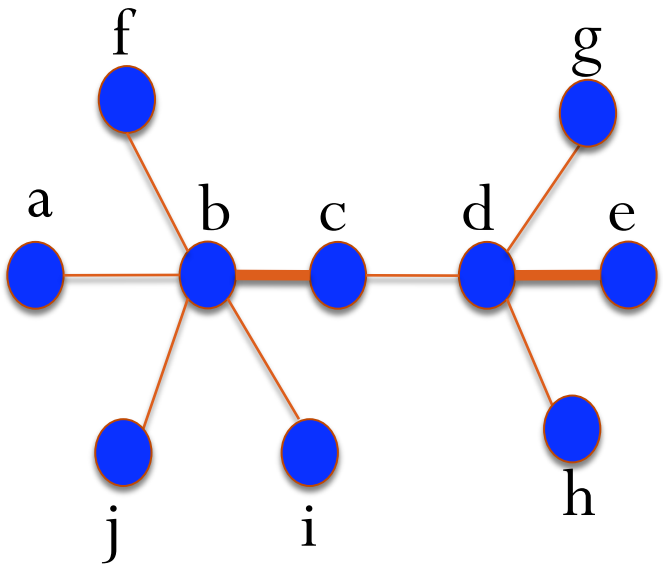
- So for any $U \subset V$, $\tfrac{1}{2} (|U| - occ(G-U) + |V|)$ is an upper bound on size of max-matching.

Max-Matching

- **(Tutte-Berge Formula):** The size of a maximum matching of a graph $G = (V, E)$ equals

$$\tfrac{1}{2} \min_{U \subset V} (|U| - occ(G-U) + |V|)$$

where *occ(H)* is the number of components in the graph *H* with an odd number of vertices.

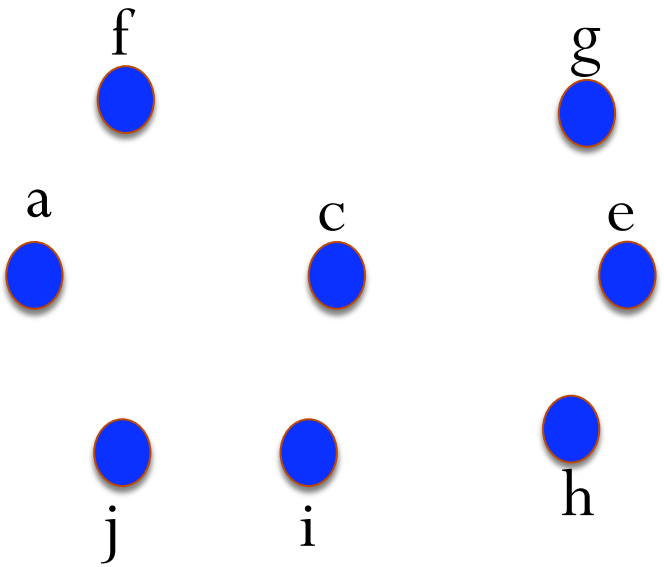


Max-Matching

- **(Tutte-Berge Formula):** The size of a maximum matching of a graph $G = (V, E)$ equals

$$\tfrac{1}{2} \min_{U \subset V} (|U| - occ(G-U) + |V|)$$

where $occ(H)$ is the number of components in the graph H with an odd number of vertices.



Let $U = \{b, d\}$. Then

$$\tfrac{1}{2} (|U| - occ(G-U) + |V|) =$$

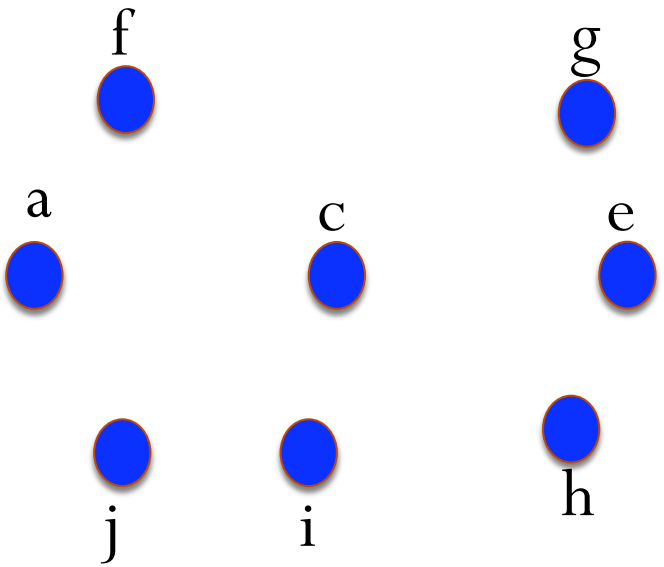
$$\tfrac{1}{2} (2 - 8 + 10) = 2.$$

Max-Matching

- **(Tutte-Berge Formula):** The size of a maximum matching of a graph $G = (V, E)$ equals

$$\tfrac{1}{2} \min_{U \subset V} (|U| - occ(G-U) + |V|)$$

where $occ(H)$ is the number of components in the graph H with an odd number of vertices.



Let U = {b, d}. Then

$\tfrac{1}{2} (|U| - occ(G-U) + |V|) =$

$\tfrac{1}{2} (2 - 8 + 10) = 2.$

For all other U,

$\tfrac{1}{2} (|U| - occ(G-U) + |V|) \geq 2.$

Max-Matching Protocol

1. H provides a feasible matching of size k. V checks feasibility with fingerprints.
2. H provides $U \subset V$ and claims $\frac{1}{2}(|U| - occ(G-U) + |V|) = k$. If so, V accepts answer k. Else, V rejects.

- Caveat: H must provide proof of the value of $occ(G-U)$, because V cannot do this on her own.

Lower Bound: $hv=\Omega(n^2)$

- Suppose protocol "A" has parameters (h, v), error $\delta = 1/3$.

Slide derived from [McGregor 10]

Lower Bound: $hv=\Omega(n^2)$

- Suppose protocol "A" has parameters (h, v), error $\delta = 1/3$.
- Define "B":

> Protocol "B"
>
> 1. V runs $t=\Theta(h)$ copies of her part of "A" in parallel
> 2. Use H's annotation a for all copies
> 3. Output majority answer if it exists, else reject.

- Protocol "B" has parameters (h, hv), error $(1/3)2^{-h}$ (i.e. each a is "bad" for only $(1/3)2^{-h}$- fraction of V's possible coin flips).

Slide derived from [McGregor 10]

Lower Bound: hv=Ω(n^2)

- Suppose protocol "A" has parameters (h, v), error $\delta = 1/3$.
- Define "B":

> Protocol "B"
>
> 1. V runs t=Θ(h) copies of her part of "A" in parallel
> 2. Use H's annotation a for all copies
> 3. Output majority answer if it exists, else reject.

- Protocol "B" has parameters (h, hv), error $(1/3)2^{-h}$ (i.e. each a is "bad" for only $(1/3)2^{-h}$- fraction of V's possible coin flips).
- Define "C": V ignores annotation, tries all 2^h possible annotations and accepts if *any one* of them causes "B" to accept.
  - Ensures error at most 1/3, no helper needed.
- Algorithm "C" solves max-matching in hv space. But we know max-matching requires Ω(n^2) space (randomized).

Slide derived from [McGregor 10]

MST

- First idea: Simulate Kruskal's algorithm, which builds min spanning tree T incrementally.
  - H replays edges in increasing order of weight.
  - If an edge e should not be added to T, make H demonstrate a cycle in $T \cup \{e\}$.
    - Problem: Each cycle can be of length O(n). Results in (mn, 1) protocol.

MST

- Second idea: Simulate implementation of Kruskal that tracks connected components induced by T. Add to T any edge connecting two nodes in different components.
  - H must annotate each edge (u, v) with component of u and v. Use fingerprinting to ensure consistency.
  - H must play entire list of components each time it changes (once for each edge added to T).
  - Yields $(m + n^2, 1)$ protocol.

MST

- Second idea: Simulate implementation of Kruskal that tracks connected components induced by T. Add to T any edge connecting two nodes in different components.
  - H must annotate each edge (u, v) with component of u and v. Use fingerprinting to ensure consistency.
  - H must play entire list of components each time it changes (once for each edge added to T).
  - Yields $(m + n^2, 1)$ protocol.

- Third idea: V remembers "batches" of edges so H doesn't have to replay connected components as often. Gives $(m + n^{1+\alpha}, n^{1-\alpha})$ protocol.

Diameter

- Theorem: $(n^2 \log n, 1)$ protocol. Lower bound: $hv = \Omega(n^2)$.
- [Chakrabarti et al. 09]: $(n^2, 1)$ protocol for matrix-matrix multiplication
- Let A be adjacency matrix of G
- $(I + A)^l_{ij} > 0$ if and only if there is a path of length at most l from i to j.
- Protocol:

1. H claims diameter is l
2. Use repeated squaring to prove $(I+A)^l$ has an entry that is 0, and $(I+A)^{l+1} \neq 0$ for all i,j).

BFS and DFS

- Goal: Force H to replay edges of G in "BFS" or "DFS" order.
- Useful subroutines for more complicated graph algorithms.
- Idea: Make H include extra information so V can tell if edges presented out of order (use lots of fingerprints).
- Can accomplish this with only constant increase in annotation per edge for both BFS and DFS.
- DFS is tricky.
- Theorem: (m, 1) protocols for "verifying" BFS and DFS.
- Corollary: (m, 1) protocol for bipartiteness.

Streaming LP problem

- Suppose stream A contains (only the non-zero) entries of matrix $\mathbf{A}$, vectors $\mathbf{b}$ and $\mathbf{c}$, interleaved in any order (updates are of the form e.g. "add y to entry (i,j) of $\mathbf{A}$"). The LP streaming problem on A is to determine $\min \{\mathbf{c}^T \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$.

Streaming LP problem

- Suppose stream A contains (only the non-zero) entries of matrix $\mathbf{A}$, vectors $\mathbf{b}$ and $\mathbf{c}$, interleaved in any order (updates are of the form e.g. "add y to entry (i,j) of $\mathbf{A}$"). The LP streaming problem on A is to determine $\min \{\mathbf{c}^T \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$.

- Theorem: There is a $(|\mathbf{A}|, 1)$ protocol for the LP streaming problem, where $|\mathbf{A}|$ is number of non-zero entries in $\mathbf{A}$.

Streaming LP problem

- Suppose stream A contains (only the non-zero) entries of matrix $\mathbf{A}$, vectors $\mathbf{b}$ and $\mathbf{c}$, interleaved in any order (updates are of the form e.g. "add y to entry (i,j) of $\mathbf{A}$"). The LP streaming problem on A is to determine $\min \{\mathbf{c}^T \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$.

- Theorem: There is a $(|\mathbf{A}|, 1)$ protocol for the LP streaming problem, where $|\mathbf{A}|$ is number of non-zero entries in $\mathbf{A}$.
  - Protocol ("naïve" matrix-vector multiplication):
    1. H provides primal-feasible solution $\mathbf{x}$.
    2. For each row i of $\mathbf{A}$:

       Repeat entries of $\mathbf{x}$ and row i of $\mathbf{A}$ as necessary to prove feasibility. Fingerprints ensure consistency.
    3. Repeat for dual-feasible solution $\mathbf{y}$. Accept if value($\mathbf{x}$)=value($\mathbf{y}$).

Streaming LP problem

$A = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $x^T = [1, 1, 1]$, $b^T = [2, 3, 3]$. Want to prove $Ax <= b$.

V

$A_1 \cdot x \mathrel{+}= 1$

H ➡ $A_{1,1} = 1, x_1 = 1$

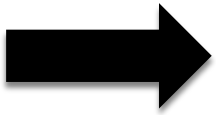
Streaming LP problem

$\mathbf{A} = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $\mathbf{x}^T = [1, 1, 1], \mathbf{b}^T = [2, 3, 3]$

V

$\mathbf{A}_1 \cdot \mathbf{x} \mathrel{+}= 1$

H



$\mathbf{A}_{1,3} = 1, \mathbf{x}_3 = 1$

H need not replay $\mathbf{A}_{1,2}$ because $\mathbf{A}_{1,2}=0$ and doesn't affect inner product

Streaming LP problem

$A = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $x^T = [1, 1, 1], b^T = [2, 3, 3]$

V

$A_1 \cdot x = 2$

H ➡ $b_1 = 2$ ✔

Streaming LP problem

$\mathbf{A} = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $\mathbf{x}^T = [1, 1, 1], \mathbf{b}^T = [2, 3, 3]$

V

$\mathbf{A}_2 \cdot \mathbf{x} \mathrel{+}= 1$

H need not replay $\mathbf{A}_{2,1}$

H → $\mathbf{A}_{22} = 1, \mathbf{x}_2 = 1$

Streaming LP problem

$A = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $x^T = [1, 1, 1], b^T = [2, 3, 3]$

V

$A_2 \cdot x \mathrel{+}= 2$

H ➡ $A_{23} = 2, x_3 = 1$

Streaming LP problem

$A = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $x^T = [1, 1, 1], b^T = [2, 3, 3]$

V

$A_2 \cdot x = 3$

H ➡ $b_2 = 3$ ✔

Streaming LP problem

$A = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $x^T = [1, 1, 1], b^T = [2, 3, 3]$

V

$A_3 \cdot x \mathrel{+}= 1$

H ➡ $A_{31} = 1, x_1 = 1$

Streaming LP problem

$\mathbf{A} = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $\mathbf{x}^T = [1, 1, 1], \mathbf{b}^T = [2, 3, 3]$

V

$\mathbf{A}_3 \cdot \mathbf{x} \mathrel{+}= 2$

H need not replay $\mathbf{A}_{3,2}$

H ➡ $\mathbf{A}_{33} = 2, \mathbf{x}_1 = 1$

Streaming LP problem

$A = \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 2 \\ 1\ 0\ 2 \end{matrix}$ $x^T = [1, 1, 1], b^T = [2, 3, 3]$

V

$A_3 \cdot x = 3$

H ➡ $b_3 = 3$ ✔

Application to Graph Streams

- Corollary: Protocol for TUM IPs, since optimality of a feasible solution is shown by a matching feasible solution of the dual of its LP relaxation.

Application to Graph Streams

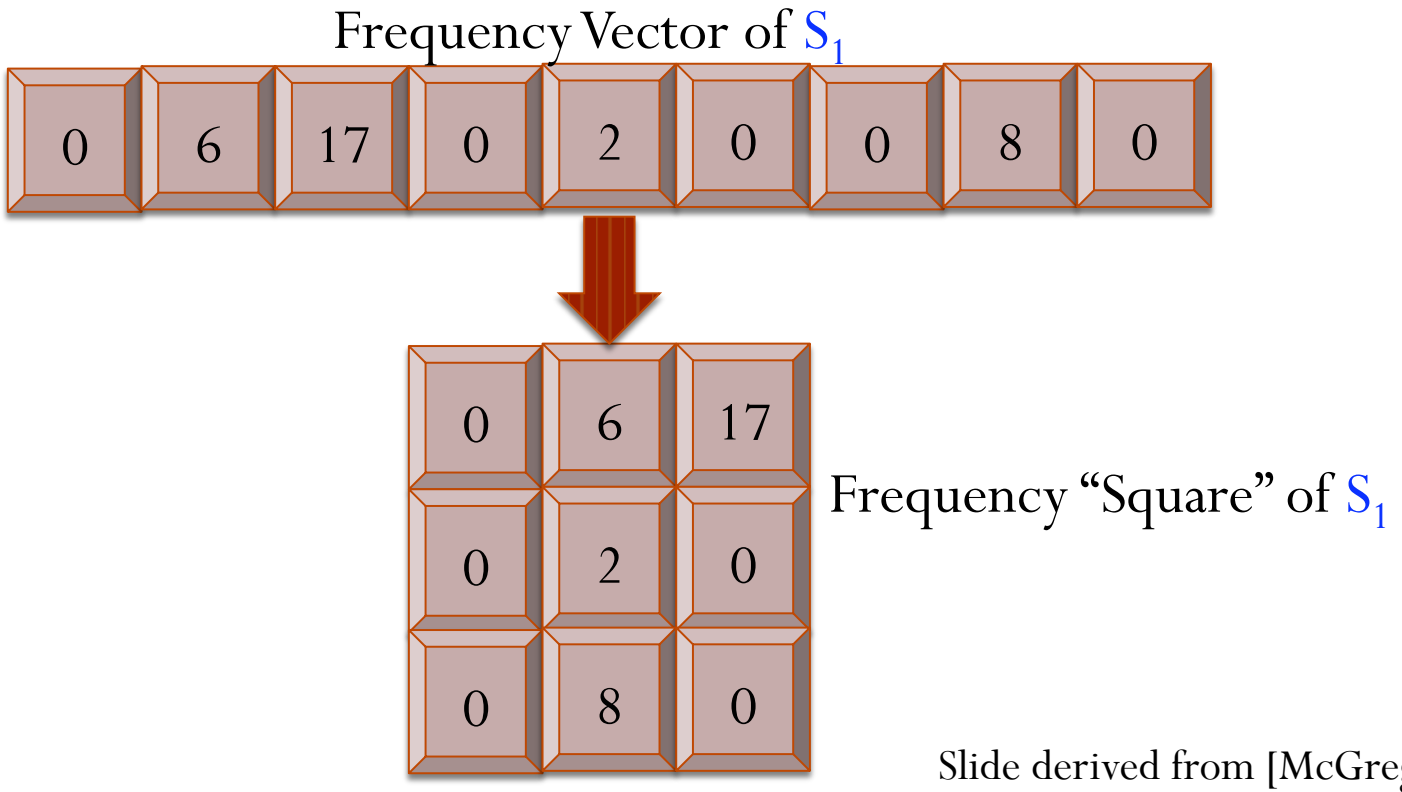
- Corollary: Protocol for TUM IPs, since optimality of a feasible solution is shown by a matching feasible solution of the dual of its LP relaxation.
- Corollary: (m, 1) protocols for max-flow, min-cut, minimum-weight bipartite perfect matching, and shortest *s*-*t* path. Lower bound of $hv = \Omega(n^2)$ for all four.

Application to Graph Streams

- Corollary: Protocol for TUM IPs, since optimality of a feasible solution is shown by a matching feasible solution of the dual of its LP relaxation.
- Corollary: (m, 1) protocols for max-flow, min-cut, minimum-weight bipartite perfect matching, and shortest *s*-*t* path. Lower bound of $hv = \Omega(n^2)$ for all four.
- **A** is sparse for the problems above, which suits the naïve protocol. For denser **A**, can get optimal tradeoffs between *h* and *v*.

Matrix-Vector Multiplication

- Background: [Chakrabarti et al. 09] ($\sqrt{n}$, $\sqrt{n}$)-protocol for inner product of frequency vectors of two streams S_1, S_2.
- View universe [n] as [$\sqrt{n}$] x [$\sqrt{n}$].

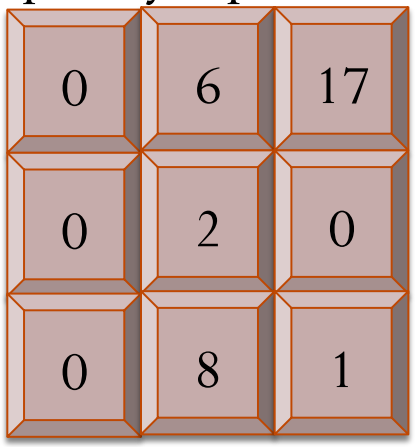


Slide derived from [McGregor 10]

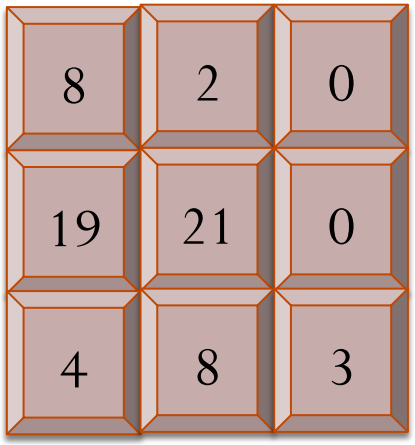
Inner-Product Protocol (1/4)

- Want to compute inner product of frequency vectors of S_1, S_2.

Frequency Square of S_1



Frequency Square of S_2



Slide derived from [McGregor 10]

Inner-Product Protocol (2/4)

- First idea: Have H send the inner product "in pieces":
  - row 1 · row 1, row 2 · row 2, etc. Requires $\sqrt{n}$ communication.
- V exactly tracks a piece at random (denoted in yellow) so if H lies about any piece, V has a chance of catching her. Requires space $\sqrt{n}$.

Frequency Square of S_1

0	6	17
0	2	0
0	8	1

Frequency Square of S_2

8	2	0
19	21	0
4	8	3

H sends

12

42

67

Slide derived from [McGregor 10]

Inner-Product Protocol (3/4)

- Problem: If H lies in only one place, V has small chance of catching her.
- Solution: Have H commit (succinctly) to inner products of pieces of a high-distance encoding of the input. If H lies about one piece, she will have to lie about many.
- Need V to evaluate any piece of the encoding in a streaming fashion. Can do this for "low-degree extension" code.

Inner-Product Protocol (4/4)

High-Distance Encoding g of Frequency Square of S_1

0	6	17
0	2	0
0	8	1
2	8	1
3	7	3
1	2	2

Input is embedded in encoding (low-degree extension)

High-Distance Encoding h of Frequency Square of S_2

8	2	0
19	21	0
4	8	3
2	3	5
7	8	1
0	2	0

These values will all lie on low-degree polynomial s(x)

H sends

12

42

67

33

80

4

Matrix-Vector Multiplication (1/8)

- Want to verify A$\mathbf{x}$=$\mathbf{b}$, for n x n matrix A. We will get optimal ($n^{1+\alpha}$, $n^{1-\alpha}$) protocol. Lower bound: $hv = \Omega(n^2)$.

Matrix-Vector Multiplication (1/8)

- Want to verify $A\mathbf{x}=\mathbf{b}$, for n x n matrix A. We will get optimal ($n^{1+\alpha}$, $n^{1-\alpha}$) protocol. Lower bound: $hv=\Omega(n^2)$.
  - Corollary: Protocols for dense LPs, effective resistances, verifying eigenvalues of Laplacian.

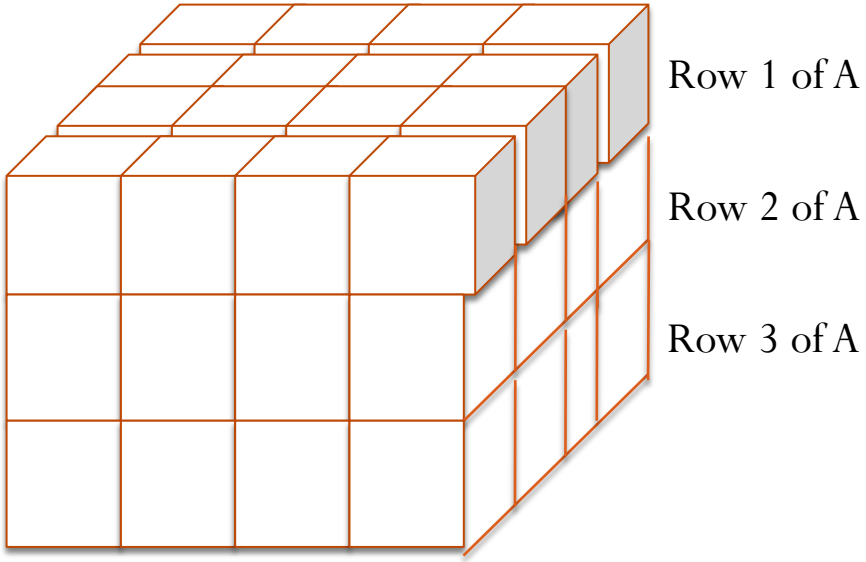
Matrix-Vector Multiplication (1/8)

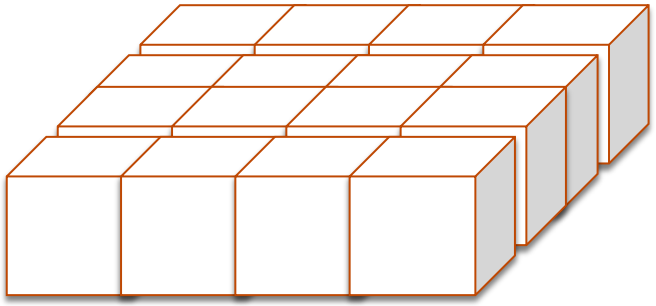
- First idea: Treat as n separate inner-product queries, one for each row of A.
  - Worse than "naïve" solution.
  - Multiplies *both* h and v by n, as compared to a single inner-product query.

Matrix-Vector Multiplication (2/8)

- First idea: Treat as n separate inner-product queries, one for each row of A.
  - Worse than "naïve" solution.
  - Multiplies *both* h and v by n, as compared to a single inner-product query.
- Key insight: one vector, $\mathbf{x}$, in each inner-product query is constant.
  - This plus linear fingerprints lets us just multiply h by n.
  - v will be the same as for a *single* inner product query.

Matrix-Vector Multiplication (3/8)





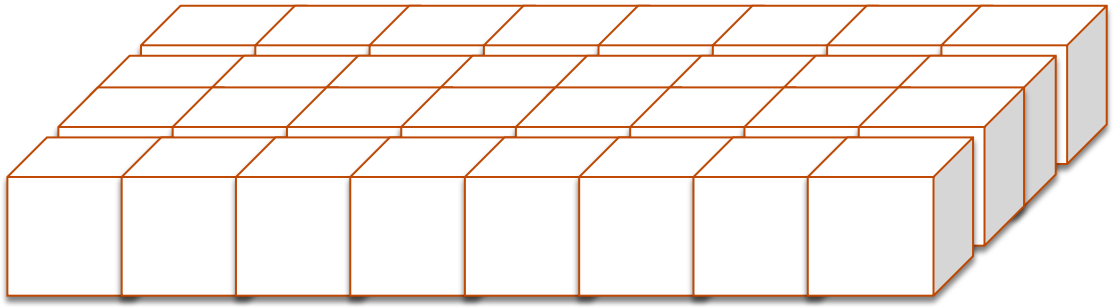
x

A

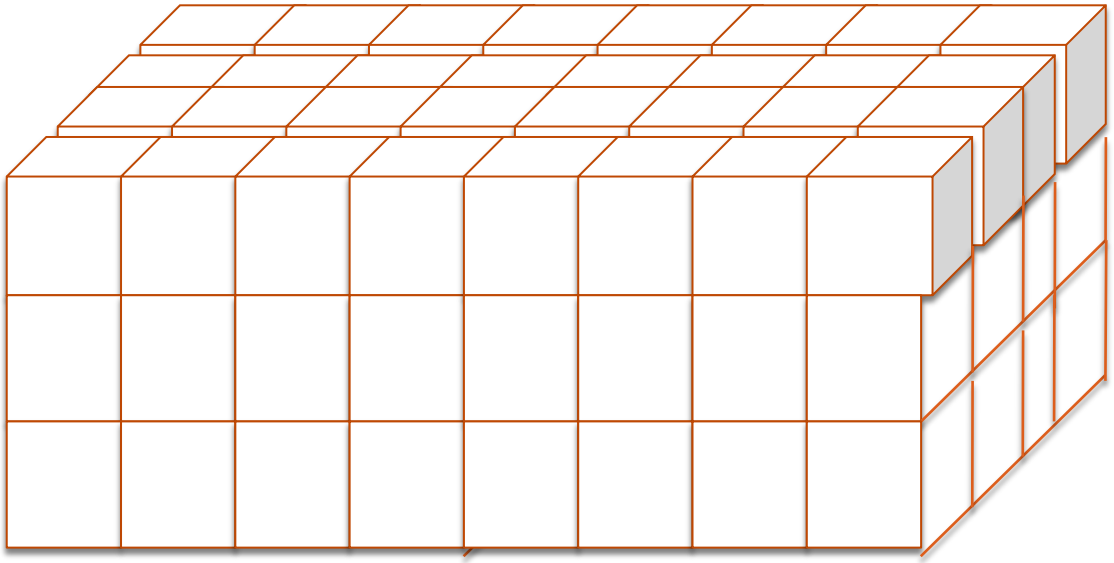
Even though this is drawn as a cube,
suppose box has dimensions n x $\sqrt{n}$ x $\sqrt{n}$

Matrix-Vector Multiplication (4/8)

Low-degree extension of each row of A and of **x**



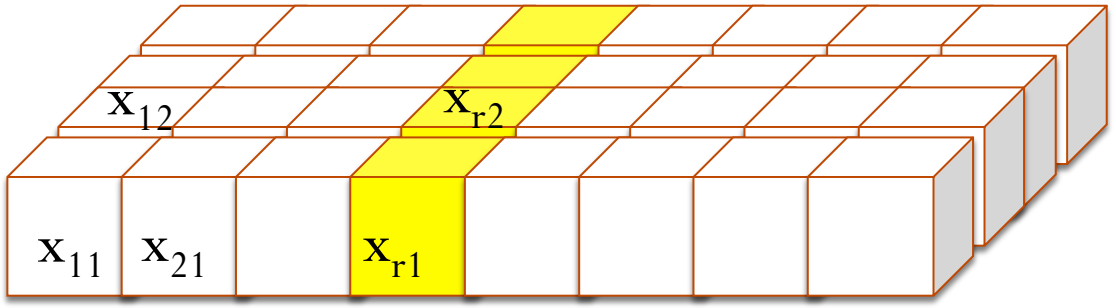
LDE of **x**



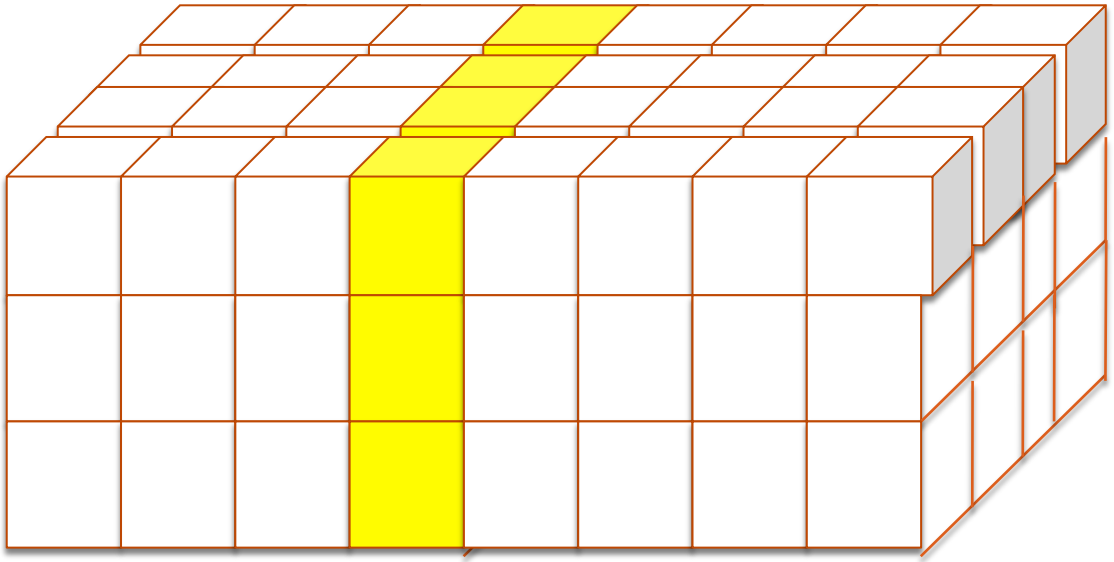
LDE of A

Matrix-Vector Multiplication (5/8)

V evaluates each row of A and **x** at random "piece" r (same r for all rows)



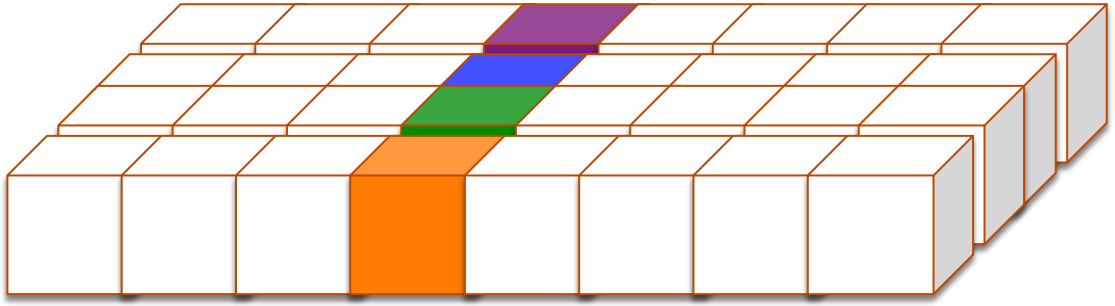
LDE of **x**



LDE of A

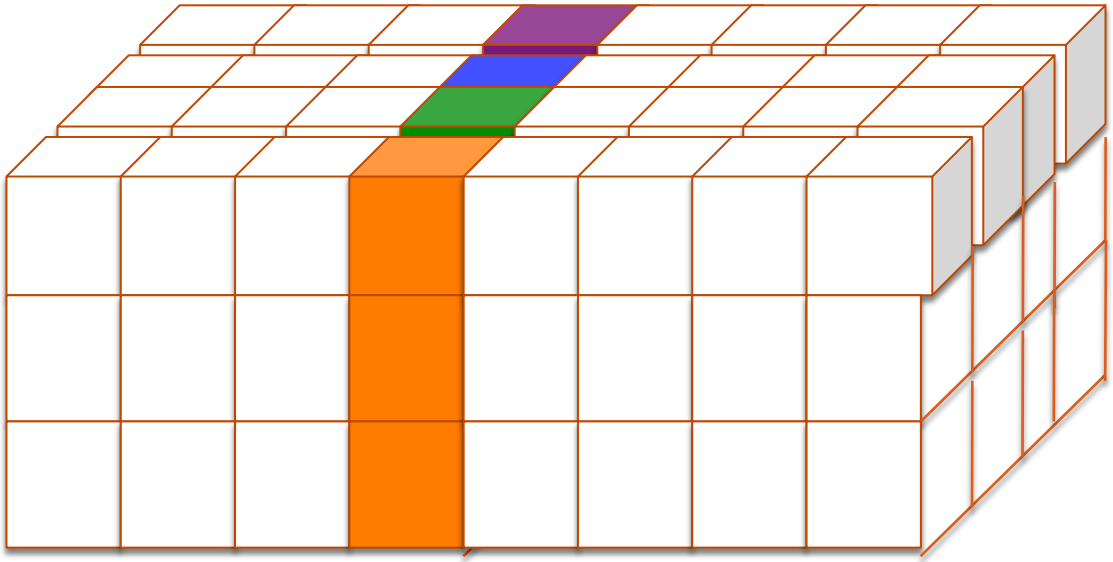
Matrix-Vector Multiplication (6/8)

Each orange entry of A gets multiplied by orange entry of **x** when computing inner product of its "piece".



LDE of **x**

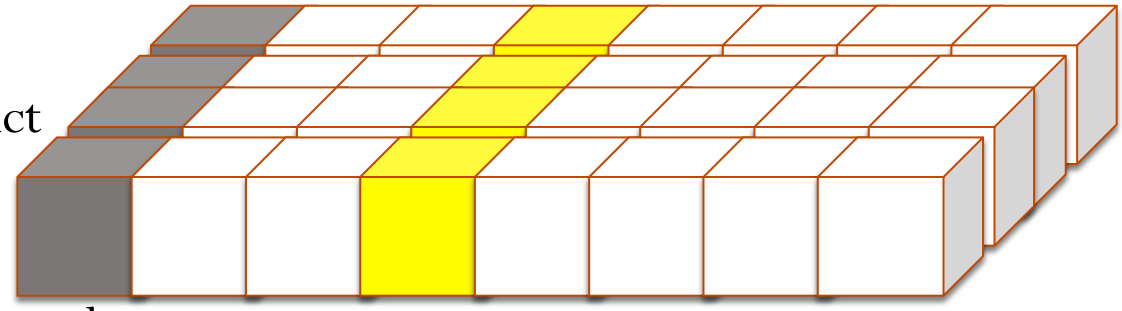
Only need to keep one fingerprint for each color.



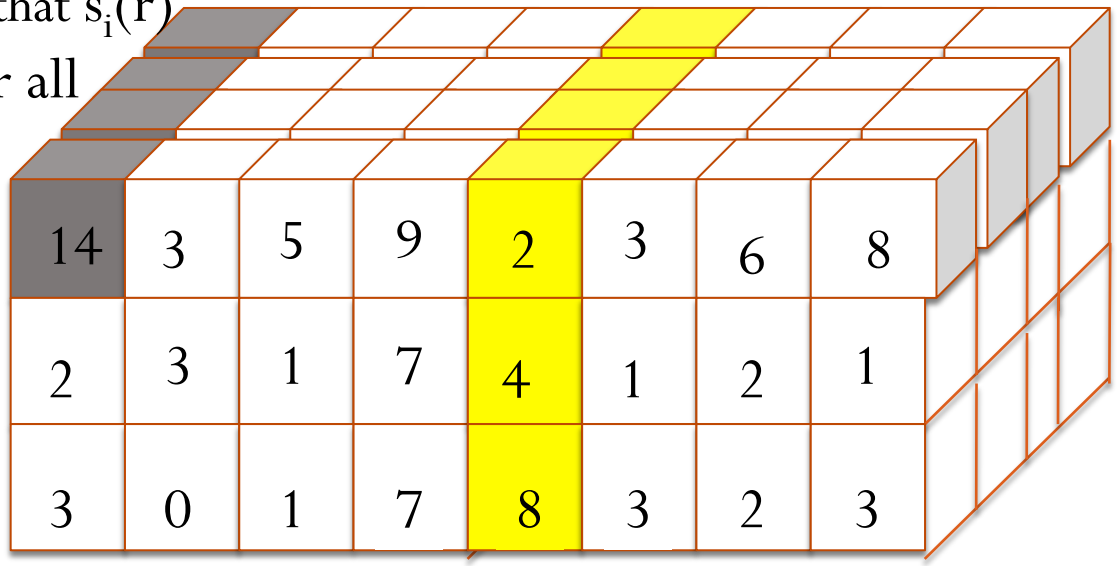
LDE of A

Matrix-Vector Multiplication (7/8)

H commits to inner product of each piece via a separate polynomial for each row. V will check that $s_i(r)$ is correct for all rows i.



LDE of **x**



LDE of A

Matrix-Vector Multiplication (8/8)

- Summary:
  - H sends the inner product of each piece of each row.
  - Conceptually, V will track a random piece of each row (the yellow entries) to catch H in any lies w.h.p.
  - But V need not store all n * $\sqrt{n}$ yellow entries!
    - Can store just $\sqrt{n}$ fingerprints $f_1, \ldots, f_{\sqrt{n}}$
    - Each fingerprint aggregates over n rows, can be computed incrementally by streaming verifier.
    - Works because vector **x** is fixed.

Summary

- (m, 1)-protocol for max-matching. $hv = \Omega(n^2)$ lower bound for dense graphs, so we can't do better.
- $(m+n^{1+\alpha}, n^{1-\alpha})$-protocol for MST.
- (m, 1)-protocols for LPs and totally-unimodular integer programs (TUM IPs). $hv = \Omega(n^2)$ lower bound for several TUM IPs.
- For any $0 <= \alpha <= 1$, optimal $(n^{1+\alpha}, n^{1-\alpha})$-protocol for dense matrix-vector multiplication. Gives $(n^{1+\alpha}, n^{1-\alpha})$-protocols for effective resistance, verifying eigenvalues of Laplacian or Adjacency matrix.
- $(n^2 \log n, 1)$ protocol for Diameter. $hv = \Omega(n^2)$ lower bound.
- Verifying BFS and DFS. Useful subroutines for future work.

Open questions

- Tradeoffs between h, v for matching, MST, diameter?

Open questions

- Tradeoffs between h, v for matching, MST, diameter?
- Other non-graph problems: Optimal tradeoffs for DISTICT, F_{max}?

Open questions

- Tradeoffs between h, v for matching, MST, diameter?
- Other non-graph problems: Optimal tradeoffs for DISTICT, F_{max}?
- Extensions to multiple rounds of interaction between H and V.
  - Surprising observation: Universal arguments [Kilian 92] can be made to work with streaming verifier, etc.
  - So can a construction in [Goldwasser et al. 08]!

Open questions

- Tradeoffs between h, v for matching, MST, diameter?
- Other non-graph problems: Optimal tradeoffs for DISTICT, F_{max}?
- Extensions to multiple rounds of interaction between H and V.
  - Surprising observation: Universal arguments [Kilian 92] can be made to work with streaming verifier, etc.
  - So can a construction in [Goldwasser et al. 08]!
- Power of closely related models.
  - [Chakrabarti et al. 09] allowed annotation to be intermingled with the stream.
  - Does this give any added power?

Our Model

- We require all annotation come at the *end* of the stream.

- Why?
  - Realistic: No synchronization required between H and V.
  - Natural: V uploads data to the cloud as it is collected, and later poses questions to H: "Is the graph connected?", "Is it bipartite?"
  - We know of no non-trivial problems that separate the "at the end" and "in-place" versions of the model.