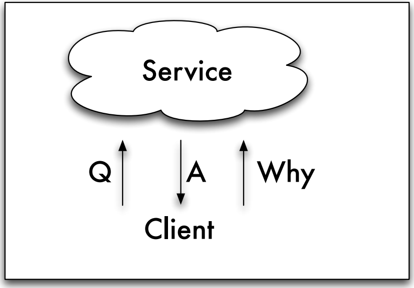
Streaming verification of graph problems

Suresh Venkatasubramanian
The University of Utah

Joint work with Amirali Abdullah, Samira Daruki and Chitradeep Dutta Roy

Outsourcing Computations

We no longer need to do our own computations: we can outsource them !



- Client (verifier) has computationally limited access to the data.
- Server (prover) reads data and has all-powerful access.
- Server must **convince** client that provided answer is correct.

Prior Work

IPs for Muggles [GKR,KRR,others]

- weaker verifiers and provers
- cryptographic assumptions
- verifier TIME key bottleneck

Prior Work

IPs for Muggles [GKR,KRR,others]
- weaker verifiers and provers
- cryptographic assumptions
- verifier TIME key bottleneck

Rational IPs [AM,CMS,others]
- Prover is rational, not adversarial
- design a "payment" scheme to convince prover that honesty is optimal

Prior Work

IPs for Muggles [GKR,KRR,others]
- weaker verifiers and provers
- cryptographic assumptions
- verifier TIME key bottleneck

Rational IPs [AM,CMS,others]
- Prover is rational, not adversarial
- design a "payment" scheme to convince prover that honesty is optimal

Proofs of proximity [RVW,GR]
- sublinear TIME verifier
- sublinear communication

Prior Work

IPs for Muggles [GKR,KRR,others]

- weaker verifiers and provers
- cryptographic assumptions
- verifier TIME key bottleneck

Rational IPs [AM,CMS,others]

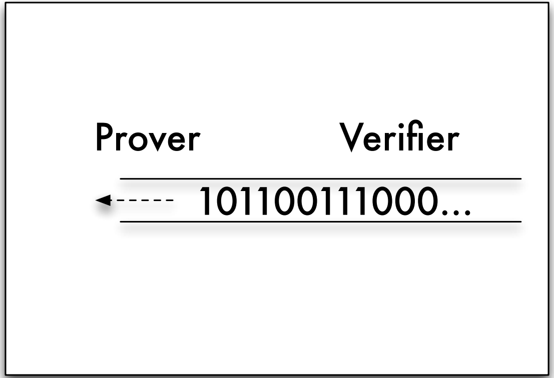
- Prover is rational, not adversarial
- design a "payment" scheme to convince prover that honesty is optimal

Proofs of proximity [RVW,GR]

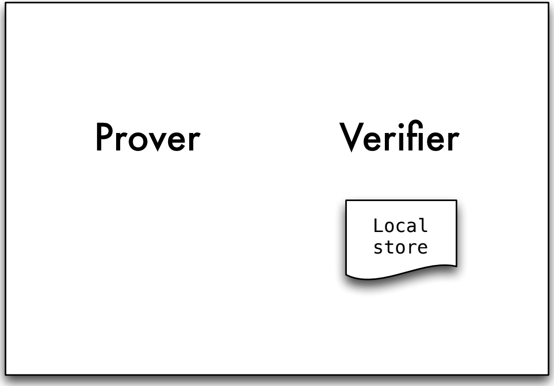
- sublinear TIME verifier
- sublinear communication

Streaming IPs [CTY,others]

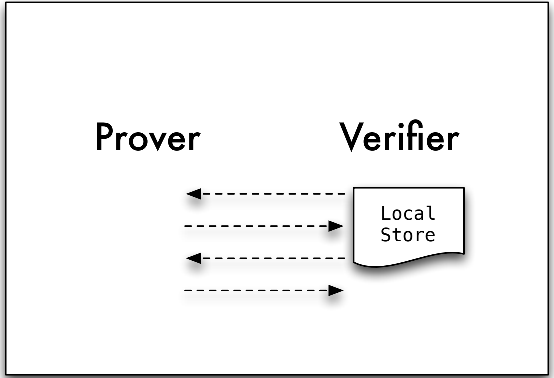
- STREAMING verifier
- sublinear communication



Prover and verifier read the stream



Verifier stores a small amount of information



Prover and verifier interact to determine the answer

Inputs

Stream of updates τ of the form $\tau_j = (i, \Delta_{i,j})$

- $i \in [u]$
- $\Delta \in \{+1, -1\}$

Updates can be assembled into a vector

$$\mathbf{a} = (a_1, a_2, \ldots, a_u)$$

where $a_i = \sum_j \Delta_{i,j}$

Measuring cost

Space:
We would like the verifier to use a working space that is *sublinear* in the input domain size:

$$s = o(u)$$

Communication:
Total communication between the prover and verifier should also be *sublinear* in u:

$$c = o(u)$$

Rounds:
Ideally, total rounds of communication should be small:

$$r \text{ should be } O(\log u) \text{ or even } O(1).$$

We will describe the cost of a protocol by the pair (s, c)

Correctness:
Protocol is randomized:

- If answer is correct, then there exists a proof that convinces verifier with certainty.
- If answer is wrong, then no proof convinces verifier with probability more than $1/3$

- Annotated streams [CCM,CCMY,CTM]: Prover helps verifier as stream goes along
- Streaming interactive proofs [CTY]: Introduce the idea of streaming interactive proofs
- Constant-round SIPs [CCMT**V**] for near neighbors, classification, and median finding, as well as complexity characterization.
- Constant- and $\log n$ round SIPs for clustering, shape fitting and eigenvector verification [DT**V**]

Graph Streams

Graph $G = (V, E), |V| = u, |E| = m$ is presented as:

Insert-only stream of edges $e \in E$

dynamic stream of updates $(e, \Delta), \Delta \in \{+1, -1\}$.

Can't do anything with $o(u)$ space !

Semi-streaming model: allow space $\Omega(u)$ but $o(m)$.

- Connectivity easy in insert-only stream.
- Connectivity easy in dynamic streams (via linear sketches)
- Matchings hard to approximate in dynamic streams
  - Cannot get better than a constant factor approximation using $\tilde{O}(u)$ space [K]
  - Linear sketches require $\Omega(u^{2-o(1)})$ space for constant factor approximation [AKLY]
- If we allow one round of communication (P $\rightarrow$ V), then space $\times$ `communication` is $\Omega(u^2)$ for exact matching [T]

Our Results

Matchings (all flavors): $O(\log u, \rho + \log u)$ protocols in $\log n$ rounds (ρ is the certificate size). Rounds can be reduced to constant if certificate is large enough.

TSP $O(\log n, n \log n)$ protocol for verifying $1.5 + \epsilon$ approximation to TSP (open whether semi-streaming algorithm can do better than 2 even for insert-only streams).

Triangle Counting $O(\log n, \log n)$ in $\log n$ rounds (exact).

Connectivity, Bipartiteness, MST $(\log n, n \log n)$ protocols.

In all cases, we linearize the graph (via matrix or tensor operations) and do (low-degree) algebraic testing on the resulting vectors.

Some Tools

Lemma (S-Z D-L)

If $p \neq q$ are degree-d polynomials, then

$$\Pr_{r \in_R \mathbb{F}}[p(r) = q(r)] \leq \frac{d}{|\mathbb{F}|}$$

Fix a function $h : \mathbb{Z} \to \mathbb{Z}$. Set $F(\mathbf{a}) = \sum_{i \in [u]} h(a_i)$

Problem (SumCheck)

Verify a claim that $F(\mathbf{a}) = K$

Problem formulated in context of interactive proofs.

Lemma (S-Z D-L)

If $p \neq q$ are degree-d polynomials, then

$$\Pr_{r \in_R \mathbb{F}}[p(r) = q(r)] \leq \frac{d}{|\mathbb{F}|}$$

Fix a function $h : \mathbb{Z} \to \mathbb{Z}$. Set $F(\mathbf{a}) = \sum_{i \in [u]} h(a_i)$

Problem (SumCheck)

Verify a claim that $F(\mathbf{a}) = K$

Problem formulated in context of interactive proofs.

Theorem (CTY)

Fix a finite field $\mathbb{F}$. There is a $\log u$-round SIP for SumCheck *with cost $(\log u, deg(h) \log u)$, where $deg(h)$ is the degree of a relaxation of h to $\mathbb{F}$.*

Note that by interpolation, any function h over a domain of size m can be written as a polynomial of degree m. Costs are expressed as the number of *words* of $\mathbb{F}$ needed.

Implications

- If $h(x) = x^2$, we get F_2 estimation: $\sum_i a_i^2$
- If $h(x) = 1$ for $x > 0$ and 0 otherwise, we get F_0: number of nonzero entries of $\mathbf{a}$.
- We can verify $F_0, F_2, F_k, F_{\max}$ *exactly* using $\log n$ space with a streaming verifier.

Implications

- If $h(x) = x^2$, we get F_2 estimation: $\sum_i a_i^2$
- If $h(x) = 1$ for $x > 0$ and 0 otherwise, we get F_0: number of nonzero entries of $\mathbf{a}$.
- We can verify $F_0, F_2, F_k, F_{\max}$ *exactly* using $\log n$ space with a streaming verifier.

By comparison with streaming:

- $\Omega(n)$ space lower bound for an exact streaming algorithm.
- Cannot even *approximate* $F_k, k \geq 3$ in $o(n^{1-2/k})$ space streaming.

Let $M = max_i a_i$. Fix $k \in [M]$.

$$F_k^{-1}(\mathbf{a}) = |\{a_i \mid a_i = k\}|$$

$F_k^{-1}(\mathbf{a})$ is the number of elements with frequency k.

Theorem (Finv)

There is a SIP to verify a claim that $F^{-1}(\mathbf{a}) = K$ that has cost $(\log n, M \log n)$ and takes $\log n$ rounds.

Let $h_k(i) = 1$ if $i = k$ and is zero otherwise. Then

$$F_k^{-1}(\mathbf{a}) = \sum_i h_k(a_i)$$

and h has degree at most M by interpolation.

Bipartite Maximum Cardinality Matchings

Problem

Given a bipartite graph $G = (A \cup B, E)$, find a set of edges $M \subset E$ so that

- *each vertex of $A \cup B$ is adjacent to at most one edge of M*
- *$|M|$ is maximized.*

Prover has to do two things

- Present a candidate matching
- Convince the verifier that this is optimal

Theorem (König)

In a bipartite graph, size of maximum cardinality matching equal size of minimum vertex cover.

Protocol:

1. V preprocesses the input stream
2. P sends V a matching, and convinces V that it is indeed a matching.
3. P sends V a vertex cover, and convinces V that it is indeed a vertex cover.

Certifying a Matching I: Subgraph check

A matching M has two properties:

1. $M \subset E$
2. Each vertex touches M at most once.

Certifying a Matching I: Subgraph check

A matching M has two properties:

1. $M \subset E$
2. Each vertex touches M at most once.

Checking that $M \subset E$

Vector **a** has one entry for each edge.

1. P and V agree on a canonical ordering of all edges
2. V processes input stream for F_{-1}^{-1} query.
3. P sends back claimed matching M *in increasing order*. V checks that there are no duplicate edges and decrements **a** for each edge in M.
4. V verifies that $F_{-1}^{-1}(\mathbf{a}) = 0$.

Certifying a Matching I: Subgraph check

A matching M has two properties:

1. $M \subset E$
2. Each vertex touches M at most once.

Checking that $M \subset E$
Vector $\mathbf{a}$ has one entry for each edge.

1. P and V agree on a canonical ordering of all edges
2. V processes input stream for F_{-1}^{-1} query.
3. P sends back claimed matching M *in increasing order*. V checks that there are no duplicate edges and decrements $\mathbf{a}$ for each edge in M.
4. V verifies that $F_{-1}^{-1}(\mathbf{a}) = 0$.

- If $M \subset E$, P passes the test.
- If $M \not\subset E$, then for $e \in M \setminus E$, $a_e = -1$ and so $F_{-1}^{-1}(\mathbf{a}) \neq 0$. If M has duplicate entries to inflate the alleged matching, then it will be detected.

Theorem (Multiset Equality, CMT)

Suppose we have streaming updates to two vectors $\mathbf{a}, \mathbf{a}' \in \mathbb{Z}^u$ *such that* $\max_i a_i, \max_i a'_i \leq M$. *Let* $t = \max(M, u)$. *Then there is a streaming algorithm using* $\log t$ *space that outputs* 1 *if* $\mathbf{a} = \mathbf{a}'$ *and outputs* 1 *with probability* $1/t^2$ *if* $\mathbf{a} \neq \mathbf{a}'$.

Theorem (Multiset Equality, CMT)

Suppose we have streaming updates to two vectors $\mathbf{a}, \mathbf{a}' \in \mathbb{Z}^u$ *such that* $\max_i a_i, \max_i a'_i \leq M$. *Let* $t = \max(M, u)$. *Then there is a streaming algorithm using* $\log t$ *space that outputs* 1 *if* $\mathbf{a} = \mathbf{a}'$ *and outputs* 1 *with probability* $1/t^2$ *if* $\mathbf{a} \neq \mathbf{a}'$.

Now to check if M is a matching:

1. V uses M to construct a stream of updates to the vertices of G.
2. V asks P to replay the vertices of M in a canonical order.
3. V verifies that these two sets are identical using Multiset Equality

Canonical ordering of vertices is needed so that prover cannot *cheat* by not sending a matching.

Certifying a matching III: Vertex Cover

A set $S \subset V$ is a vertex cover if each edge $e \in E$ is adjacent to some vertex of S.

Vector $\mathbf{a}$ has one entry for each edge in E.

1. V processes data stream for F_1^{-1} query
2. P sends a stream of vertices in S as claimed vertex cover.
3. For each vertex $v \in S$, V simulates the stream of updates $(v, w, -1)$ for all $w \in V$.
4. V verifies at end of stream that $F_1^{-1}(\mathbf{a}) = 0$.

If any edge is left uncovered, then its original count is 1 and this is never decremented.

Complexity of Protocol

Subgraph Check $(\log n, |M| + \log n)$ via Finv

Matching Check $(\log n, |M| + \log n)$ via MultiSetEquality

Vertex Cover Check $(\log n, |M| + \log n)$ via Finv

- Note that in all invocations of Finv the range of values of a_i is small.
- Overall protocol takes $\log n$ rounds.

Verifying matchings in weighted nonbipartite graphs

- Let w_{ij} be the weight of an edge $e = \{i, j\}$.
- Fix (dual) variables y_v and z_U, where U is odd-size subset of V

Theorem (Cunningham-Marsh, LP-duality)

For every integral edge weights $\{w_{ij}\}$, and choices of y, z such that for all i, j

$$y_i + y_j + \sum_{odd\ U, i, j \in U} z_U \geq w_{ij}$$

we have that

$$c^* \leq \sum_v y_v + \sum_{odd\ U} z_U \lfloor \frac{1}{2} |U| \rfloor$$

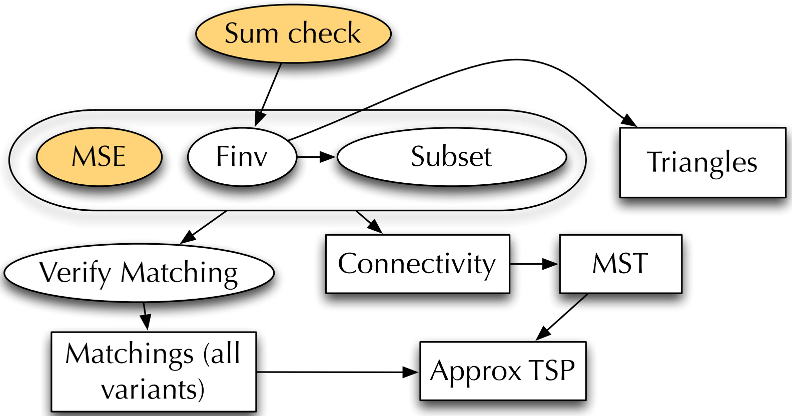
And this bound is tight for a laminar family $\{U \mid z_U > 0\}$

- In a laminar family of sets any pair of sets are either disjoint or are nested.
- Therefore a laminar family over a universe of size u is of size at most u.

A few more technical notes

- We can reduce the number of verification rounds to $c = O(1)$ if we allow communication to increase to $n^{1/c}$
- Protocols ignore verifier time: this can also be reduced by increasing the space slightly.

Overview Of Results



Conclusions

- Graphs are hard to process in a stream: but with a little help, we can solve many graph problems with limited space.
- We don't understand the full power of SIPs: lower bounds (for constant rounds) are linked to known hard classes like AM.
- There are three canonical hard problems for streaming problems: INDEX, DISJOINTNESS and Boolean Hidden (hyper)Matching. All are easy for SIPs.
- What are candidate hard problems for the SIP model in $\log n$ rounds ?

Conclusions

- Graphs are hard to process in a stream: but with a little help, we can solve many graph problems with limited space.
- We don't understand the full power of SIPs: lower bounds (for constant rounds) are linked to known hard classes like AM.
- There are three canonical hard problems for streaming problems: INDEX, DISJOINTNESS and Boolean Hidden (hyper)Matching. All are easy for SIPs.
- What are candidate hard problems for the SIP model in $\log n$ rounds ?

Thank You !

suresh@cs.utah.edu

http://www.cs.utah.edu/∼suresh