Semi-Streaming Algorithms for Annotated Graph Streams

Justin Thaler, Yahoo Labs
Data Streaming Model

• Stream: m elements from universe of size N
  • e.g., \(<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., number of distinct elements, frequency moments, heavy hitters.

• Challenge:
  (i) Limited working memory, i.e., polylog(m,N).
  (ii) Sequential access to adversarially ordered data.
  (iii) Process each update quickly.
Graph Streams

- In a graph stream, elements are **edges** in a graph G on n nodes.

- Goal: Compute properties of G, e.g., Is it connected? Approximately how many triangles does it have? What is its maximum weight matching?

   - Example: distinguishing graphs with 0 triangles from those with 1 triangle.

   - A bright spot: some simple properties can be solved in $O(n^{\text{polylog}}(n))$ space.

   Examples: **bipartiteness**, **connectivity**. These are called **semi-streaming algorithms**.
Graph Streams

- In a graph stream, elements are edges in a graph G on n nodes.

- Goal: Compute properties of G, e.g., Is it connected? Approximately how many triangles does it have? What is its maximum weight matching?

- Bad news: many graph problems cannot be solved (or even approximated) by a streaming algorithm in o(n^2) space.
  - Example: distinguishing graphs with 0 triangles from those with 1 triangle.
Graph Streams

- In a graph stream, elements are edges in a graph G on n nodes.

- Goal: Compute properties of G, e.g., Is it connected? Approximately how many triangles does it have? What is its maximum weight matching?

- Bad news: many graph problems cannot be solved (or even approximated) by a streaming algorithm in \( o(n^2) \) space.
  - Example: distinguishing graphs with 0 triangles from those with 1 triangle.

- A bright spot: some simple properties can be solved in \( O(n^{*\text{polylog}(n)}) \) space.
  - Examples: bipartiteness, connectivity
  - These are called semi-streaming algorithms.
Outsourcing

- Many applications require outsourcing computation to untrusted service providers.
  - Main motivation: commercial cloud computing services.
  - Also, weak peripheral devices; fast but faulty co-processors.
  - Volunteer Computing (SETI@home, World Community Grid, etc.)

- User requires a guarantee that the cloud performed the computation correctly.
AWS Customer Agreement

WE… MAKE NO REPRESENTATIONS OF ANY KIND … THAT THE SERVICE OR THIRD PARTY CONTENT WILL BE UNINTERRUPTED, ERROR FREE OR FREE OF HARMFUL COMPONENTS, OR THAT ANY CONTENT … WILL BE SECURE OR NOT OTHERWISE LOST OR DAMAGED.
Model of Streaming Verification for This Work

- Chakrabarti et al. [CCM09/CCMT14] introduced the model of annotated data streams.
- One message (non-interactive) model: $P$ and $V$ both observe stream. Afterward, $P$ sends $V$ an email with the answer, and a proof attached.
- Think of $V$’s streaming pass over the input as occurring while $V$ is uploading data to the cloud.
Annotated Data Streams

Cloud Provider

Business/Agency/Scientist
Annotated Data Streams

Cloud Provider

Data

Business/Agency/Scientist
Annotated Data Streams
Annotated Data Streams

Cloud Provider

Data

Question

Business/Agency/Scientist

Summary
Annotated Data Streams

Cloud Provider

Question

Answer + Proof

Business/Agency/Scientist

Summary
Annotated Data Streams

Cloud Provider

Data

Business/Agency/Scientist

Question

Answer + Proof

Accept or Reject
Annotated Data Streams

• Prover P and Verifier V observe a stream.
  • P solves problem, tells V the answer.
    • P appends a proof that the answer is correct.

• Requirements:
  • 1. Completeness: an honest P can convince V to accept.
  • 2. Soundness: V will catch a lying P with high probability (secure even if P is computationally unbounded).
Costs of Annotated Data Streams

- Two main costs: proof length, and \( V \)'s working memory. Both must be sublinear in input size.

For graph problems on \( n \) nodes, refer to a protocol of total cost \( O(n \text{polylog}(n)) \) as a semi-streaming scheme.
Costs of Annotated Data Streams

- Two main costs: proof length, and \( V \)'s working memory. Both must be sublinear in input size.
  - Notation: an \((h,v)\)-protocol is one with proof length \( O(h) \) and memory cost \( O(v) \) for \( V \).
  - The total cost of the protocol is \( h+v \).
  - For graph problems on \( n \) nodes, refer to a protocol of total cost \( O(n*\text{polylog}(n)) \) as a semi-streaming scheme.
- Other costs: running time of both \( P \) and \( V \).
Another Model of Streaming Verification

- Cormode et al. [CTY12] introduced more general model called streaming interactive proofs (SIPs) that allows multiple rounds of interaction between P and V.
- Annotated data streams correspond to 1-message SIPs.
Comparison of Two Models

- Pros of multi-round model:
  1. Exponentially reduces space and communication cost. Often (polylog n, polylog n).

- Cons of multi-round model:
  1. \( P \) must do significant computation after each message.
  2. More coordination needed; network latency might be an issue.

- Pros of single-message model:
  1. Space and communication still reasonable.
  2. \( P \) can do all computation at once, just send an email with proof attached.
  3. Reusability: can run the protocol on a stream, then receive more stream updates and seamlessly run the protocol on the updated stream.
History of Annotated Data Streams and SIPs

- [CCM09, CTY12, KP13, GR13, CTY12, PSTY13, CCMTV14, KP14, DTV15, ADDRV16] all study variants of these models.

- [CMT12] gave efficient implementations of protocols from [CCM09, CMT10] (and from the literature on “classical” interactive proofs).
Our Results

- Part 1: We give semi-streaming schemes for exactly solving two graph problems in dynamic graphs streams that require $\Omega(n^2)$ space in the standard streaming model.
  - Counting triangles.
  - Maximum cardinality matching.
  - These protocols are provably optimal.
Our Results

• Part 1: We give semi-streaming schemes for exactly solving two graph problems in dynamic graphs streams that require $\Omega(n^2)$ space in the standard streaming model.
  • Counting triangles.
  • Maximum cardinality matching.
  • These protocols are provably optimal.
  • Only known semi-streaming schemes were for bipartite perfect matching, and shortest s-t path in graphs of polylogarithmic diameter [CMT10, CCM09/CCMT14].

• Part 2: We show two graph problems that are just as hard in the annotated data streaming model.
  • Connectivity and bipartiteness.
  • Caveat: the result holds in the "XOR edge update" model.
Our Results

- Part 1: We give semi-streaming schemes for exactly solving two graph problems in dynamic graphs streams that require \( \Omega(n^2) \) space in the standard streaming model.
  - Counting triangles.
  - Maximum cardinality matching.
  - These protocols are provably optimal.
  - Only known semi-streaming schemes were for bipartite perfect matching, and shortest s-t path in graphs of polylogarithmic diameter [CMT10, CCM09/CCMT14].

- Part 2: We show two graph problems that are just as hard in the annotated data streaming model.
  - Connectivity and bipartiteness.
  - Caveat: the result holds in the “XOR edge update” model.
Semi-Streaming Schemes for Counting Triangles
Summary of Annotated Data Streaming Protocols for Counting Triangles

<table>
<thead>
<tr>
<th>Reference</th>
<th>(Proof Length, Space Cost)</th>
<th>Total Cost Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CCMT14]</td>
<td>((n^2, 1))</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>[CCMT14]</td>
<td>((h, v): \text{for any } h\cdot v = n^3)</td>
<td>(O(n^{3/2}))</td>
</tr>
<tr>
<td>This work</td>
<td>((n, n))</td>
<td>(O(n))</td>
</tr>
</tbody>
</table>

- [CCMT14] proved a lower bound that any \((h, v)\) protocol must satisfy \(h\cdot v > n^2\).
- Question of whether there is semi-streaming scheme for the problem is Question #47 on sublinear.info (posed by Cormode at Bertinoro 2011).
- Interesting properties of our solution:
  - \(V\)'s final state depends on the order of the stream.
  - Our approach does not allow smooth tradeoffs of proof length and space cost.
Outline of the Exposition

1. Sum-Check Protocol of [LFKN90]
   (a) Simple, non-interactive variant
   (b) Full Interactive Sum-Check Protocol

2. Low-Degree Extensions

3. A Simple, Interactive Protocol for Counting Triangles, via (b)

4. The Annotated Data Streaming Protocol, via (a).
Sum-Check Protocol [LFKN90], Simplified

- Let $\mathbb{F}$ be a finite field of (prime) size at least $n^3$.
- Associate elements of $\mathbb{F}$ with integers in the natural way.
Sum-Check Protocol [LFKN90], Simplified

- Let $\mathbb{F}$ be a finite field of (prime) size at least $n^3$.
- Associate elements of $\mathbb{F}$ with integers in the natural way.
- Claim: Suppose we identify a univariate polynomial $g$ (that depends on the input stream) over $\mathbb{F}$ such that
  1. The number of triangles in the graph equals $\sum_{b \in [n]} g(b)$.
  2. For a randomly chosen point $r \in \mathbb{F}$, $V$ can evaluate $g(r)$ using space $v$ with a single streaming pass over the stream.

Then there is a $(\deg(g), v)$ -protocol for counting triangles.
Sum-Check Protocol [LFKN90], Simplified

- Let $\mathbb{F}$ be a finite field of (prime) size at least $n^3$.
- Associate elements of $\mathbb{F}$ with integers in the natural way.
- Claim: Suppose we identify a univariate polynomial $g$ (that depends on the input stream) over $\mathbb{F}$ such that
  1. The number of triangles in the graph equals $\sum_{b \in [n]} g(b)$.
  2. For a randomly chosen point $r \in \mathbb{F}$, $V$ can evaluate $g(r)$ using space $v$ with a single streaming pass over the stream.

Then there is a $(\deg(g), v)$-protocol for counting triangles.

- Proof: $P$ sends a polynomial $s$ (specified by its coefficients) claimed to equal $g$. $V$ checks if $s(r) = g(r)$ and if so outputs $\sum_{b \in [n]} s(b)$.
- Completeness is obvious. Soundness error is at most $\deg(g)/|\mathbb{F}|$. 
Sum-Check Protocol [LFKN90]

• Suppose the input specifies a $d$-variate polynomial $g$ over field $\mathbf{F}$.

• Goal: compute the quantity:

\[
\sum \sum \cdots \sum g(b_1, \ldots, b_d)
\]

\[
\begin{aligned}
&b_1 \in [n] \quad b_2 \in [n] \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad b_d \in [n]
\end{aligned}
\]

• Costs:
  • $d$ rounds of interaction.
  • Total communication is $O(d \cdot \deg(g))$.
  • Space cost for $\mathbf{V}$ is the space to evaluate $g$ at a random point.
Low-Degree Extensions

- Define $E : [n] \times [n] \rightarrow \{0, 1\}$ by:
  
  \[ E(u, v) = 1 \text{ if edge } (u, v) \text{ appears in } G. \]
  
  \[ E(u, v) = 0 \text{ otherwise.} \]

- Let $F$ be a field, and let $\tilde{E}(u, v)$ denote the bivariate polynomial over $F$ of degree $n$ in each variable that agrees with $E$ at all inputs in $[n] \times [n]$.

- Fact: For any point $(r_1, r_2) \in F^2$, $V$ can evaluate $\tilde{E}(r_1, r_2)$ in constant space with a single streaming pass over the input.
$E : [n] \times [n] \rightarrow \{0, 1\}$
\[ \tilde{E} : \mathbb{F}^2 \to \mathbb{F} \]
\[ \tilde{E} : \mathbf{F}^2 \rightarrow \mathbf{F} \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>-3</td>
<td>-5</td>
<td>-7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>-2</td>
<td>-5</td>
<td>-8</td>
<td>-11</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-3</td>
<td>-7</td>
<td>-11</td>
<td>-15</td>
<td></td>
</tr>
</tbody>
</table>

-8 is highlighted in yellow.
A Simple Interactive Protocol for Counting Triangles

- The number of triangles in G equals
  \[ \sum_{u \in [n]} \sum_{v \in [n]} \sum_{z \in [n]} E(u,v) \cdot E(v,z) \cdot E(u,z). \]

- Get a 3-round (n, 1)-protocol by applying sum-check to the trivariate polynomial \( g(X,Y,Z) = E(X,Y) \cdot E(Y,Z) \cdot E(X,Z). \)
A Simple Interactive Protocol for Counting Triangles

- The number of triangles in $G$ equals
  \[
  \sum_{u \in [n]} \sum_{v \in [n]} \sum_{z \in [n]} \tilde{E}(u, v) \cdot \tilde{E}(v, z) \cdot \tilde{E}(u, z).
  \]

- Get a 3-round $(n, 1)$-protocol by applying sum-check to the \textbf{trivariate} polynomial $g(X, Y, Z) = \tilde{E}(X, Y) \cdot \tilde{E}(Y, Z) \cdot \tilde{E}(X, Z)$.

- Can get a 2-round $(n, n)$-protocol by applying sum-check to the \textbf{bivariate} polynomial $g'(X, Y) = \tilde{E}(X, Y) \cdot \sum_{z \in [n]} \tilde{E}(Y, z) \cdot \tilde{E}(X, z)$.

- $V$ can evaluate $g'$ at a random point $(r_1, r_2) \in \mathbb{F}^2$ in space $O(n)$ by computing $\tilde{E}(r_1, r_2)$, as well as $\tilde{E}(r_1, z)$ and $\tilde{E}(r_2, z)$ for all $z \in [n]$. 

The Annotated Data Streaming Protocol: Outline

• To get a semi-streaming scheme, we need to write the number of triangles in the graph as $\sum_{b \in [n]} g(b)$ for a univariate polynomial $g$ of degree $O(n)$ that $V$ can evaluate at any point in $O(n)$ space.
To get a semi-streaming scheme, we need to write the number of triangles in the graph as $\sum_{b \in [n]} g(b)$ for a \textit{univariate} polynomial $g$ of degree $O(n)$ that $V$ can evaluate at any point in $O(n)$ space.

Key idea: $g$ will itself be a \textit{sum} of polynomials $g_i$, one for each stream update.

$\sum g_i(z)$ will count the number of triangles \textit{completed} at time $i$.

Hence, the total number of triangles will be

$$\sum_{i \leq m} \left( \sum_{z \in [n]} g_i(z) \right) = \sum_{z \in [n]} \left( \sum_{i \leq m} g_i(z) \right) = \sum_{z \in [n]} g(z).$$

Need to ensure each $g_i$ has degree $O(n)$ and that for any $r$ and all $i$, $V$ can evaluate $g_i(r)$ in $O(n)$ space.
The Annotated Data Streaming Protocol: Details

- Define $E_i : [n] \times [n] \rightarrow \{0,1\}$ by:
  $$E_i(u, v) = 1 \text{ if edge } (u, v) \text{ appears in } G \text{ after } i \text{ stream updates.}$$
  $$E_i(u, v) = 0 \text{ otherwise.}$$

- If the $i$'th stream update is edge $(u_i, v_i)$, define
  $$g_i(Z) = \tilde{E}_i(u_i, Z) \cdot \tilde{E}_i(v_i, Z).$$
The Annotated Data Streaming Protocol: Details

- Define $E_i : [n] \times [n] \to \{0, 1\}$ by:
  
  $$E_i(u, v) = \begin{cases} 
  1 & \text{if edge } (u, v) \text{ appears in } G \text{ after } i \text{ stream updates.} \\
  0 & \text{otherwise.}
  \end{cases}$$

- If the $i'$th stream update is edge $(u_i, v_i)$, define
  
  $$g_i(Z) = \tilde{E}_i(u_i, Z) \cdot \tilde{E}_i(v_i, Z).$$

- Observe:
  
  - $g_i$ is a univariate polynomial of degree at most $2n$.
  - $\sum_{z \in [n]} g_i(z)$ is the number of triangles completed by $(u_i, v_i)$ at time $i$.

- $V$ can evaluate $g_i(r) = \tilde{E}_i(u_i, r) \cdot \tilde{E}_i(v_i, r)$ by maintaining
  
  $\tilde{E}_i(u, r)$ for all $u \in [n]$ at all times $i$.

- Hence, $V$ can also evaluate $g(r) = \sum_{i \leq m} g_i(r)$ in $O(n)$ space.
Semi-Streaming Scheme for Maximum Cardinality Matching
## Summary of Annotated Data Streaming Protocols for Maximum Cardinality Matching

<table>
<thead>
<tr>
<th>Reference</th>
<th>(Proof Length, Space Cost)</th>
<th>Total Cost Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CMT10]</td>
<td>(m, 1)</td>
<td>O(m)</td>
</tr>
<tr>
<td>This work</td>
<td>(n, n)</td>
<td>O(n)</td>
</tr>
</tbody>
</table>

- [CCMT14] proved a lower bound that any (h, v) protocol must satisfy $h \times v > n^2$ (even in the bipartite case).
Lower Bounds for Connectivity and Bipartiteness
Overview of Lower Bound and Proof

- Claim: In the XOR update model, any annotated data streaming protocol for Connectivity and Bipartiteness must have total cost $\Omega(n)$. These problems are solvable in $O(n \cdot \text{polylog}(n))$ space without a prover.
Overview of Lower Bound and Proof

- Claim: In the XOR update model, any annotated data streaming protocol for Connectivity and Bipartiteness must have total cost $\Omega(n)$. These problems are solvable in $O(n^{*\text{polylog}(n)})$ space without a prover.

- Proof sketch:
  - Known fact: any annotated data streaming protocol for the INDEX problem on $N$ bits must have total cost $\Omega(N^{1/2})$ (this is tight).
  - We reduce INDEX on $n^2$ bits to Connectivity on graphs with $n$ nodes.
Overview of Lower Bound and Proof

- Claim: In the XOR update model, any annotated data streaming protocol for Connectivity and Bipartiteness must have total cost $\Omega(n)$. These problems are solvable in $O(n \cdot \text{polylog}(n))$ space without a prover.

- Proof sketch:
  - Known fact: any annotated data streaming protocol for the INDEX problem on $N$ bits must have total cost $\Omega(N^{1/2})$ (this is tight).
  - We reduce INDEX on $n^2$ bits to Connectivity on graphs with $n$ nodes.
  - Reduction is tailored to the annotated data streaming model: $P$ helps $V$ perform the reduction.
  - This is necessary.
    - Connectivity on $n$ nodes is easier than INDEX on $n^2$ bits in the standard streaming model, but they’re equally hard in annotated data streaming model.
Open Questions
Open Questions

- Exhibit any graph problem that \textbf{cannot} be solved by a semi-streaming scheme.
- Do there exist non-trivial (i.e., $o(n^2)$ total cost) annotated data streaming protocols for any of the following?
  - Shortest $s$-$t$ path in general graphs
  - Graph diameter
  - Computing the value of a maximum flow.
- Do there exist annotated data streaming protocols of $o(n)$ total cost for Connectivity or Bipartiteness in the insert-only update model? The strict turnstile update model?
- Is it possible to give an annotated data streaming protocols for Counting Triangles of space cost $o(n)$ and help cost $o(n^2)$?
Thank you!