Annotations For Sparse Data Streams

Justin Thaler, Simons Institute, UC Berkeley

Joint Work with:

Amit Chakrabarti, Dartmouth

Graham Cormode, University of Warwick

Navin Goyal, Microsoft Research India

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.

Goals of Verifiable Computation

- Goal 1: Provide user with a correctness guarantee.
- Goal 2: User must operate within the restrictive **data streaming paradigm** (models a user who lacks the resources to store the input locally).

Annotated Data Stream (ADS) Model

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

$$S = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9 \ldots, x_m]$$

- **Prover P:** Augments S with h-bit annotation.

$$(S, a) = [a_0, x_1, x_2, x_3, a_1, x_4, x_5, x_6, x_7, a_2, x_8, x_9 \ldots, x_m, a_h]$$

Annotation is a function of previous stream elements

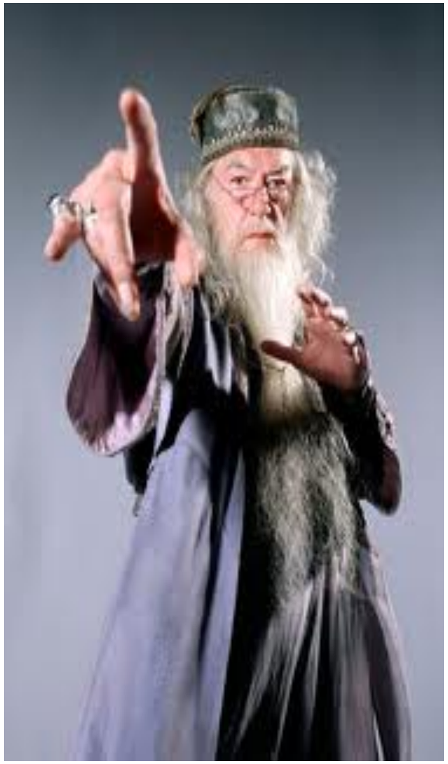
- **Verifier V:** Process annotated data stream. Output an answer, or reject annotation as invalid.
- Captures "Merlin-Arthur protocols with a streaming verifier". Introduced in [CCM09/CCMT14].
- All algorithms in this talk apply to **strict turnstile** streaming model.

Annotated Data Streams

- Requirements:

  1. Completeness: honest P will convince verifier to output correct answer.

  2. Soundness: **no** P can convince V to output an incorrect answer, except with tiny probability.



- Goal: Minimize annotation length and size of V's working memory.

Prior Work

- [CCM09/CCMT14] introduced ADS model, gave optimal (annotation length, space) tradeoffs for INDEX, frequency moments, some graph problems, etc.
- [CMT10] gave optimal ADS protocols for still more problems.
- [CMT12] gave efficient implementations of protocols from [CCM09/CCMT14, CMT10].
- [KP13, GR13, CTY12, CCMTV14] study variants of the ADS model.

This Work: “Sparse” Streams

- Many streams are over enormous domain sizes (e.g. IPv6 flows).
  - Existing results have costs that depend on **domain size** n.
  - E.g. [CCM09] gives ($\sqrt{n}$ annotation, $\sqrt{n}$ space)-protocol for F_2.
  - This is optimal for “dense” streams (with **length** $m = \Omega(n)$).
- We want costs to depend only on the stream length m.
- Bottom line: we give near-optimal tradeoffs in terms of m for frequency moments, graph problems, etc.

Problem	Our Costs (ann. length, space)	Previous Best (ann. length, space) [CCM09/CCMT14, CMT10]	Lower Bound
INDEX, MEDIAN	$(x, y) : x \cdot y \geq m$ E.g. $(\sqrt{m}, \sqrt{m})$	$(x, y) : x \cdot y \geq n.$ E.g. $(\sqrt{n}, \sqrt{n})$	$x \cdot y = \Omega(m).$
F_2, PERFECT MATCHING, CONNECTIVITY, BIPARTITENESS	$(x, y) : x \cdot \sqrt{y} \geq m$ E.g. $(m^{2/3}, m^{2/3})$	$(x, y) : x \cdot y \geq n.$ E.g. $(\sqrt{n}, \sqrt{n})$	$x \cdot y = \Omega(m).$

Problem	Our Costs (ann. length, space)	Previous Best (ann. length, space) [CCM09/CCMT14, CMT10]	Lower Bound
INDEX, MEDIAN	$(x, y): x \cdot y \geq m$ E.g. $(\sqrt{m}, \sqrt{m})$	$(x, y): x \cdot y \geq n.$ E.g. $(\sqrt{n}, \sqrt{n})$	$x \cdot y = \Omega(m).$
F_2, PERFECT MATCHING, CONNECTIVITY, BIPARTITENESS	$(x, y): x \cdot \sqrt{y} \geq m$ E.g. $(m^{2/3}, m^{2/3})$	$(x, y): x \cdot y \geq n.$ E.g. $(\sqrt{n}, \sqrt{n})$	$x \cdot y = \Omega(m).$

Other Results:

- Give the first explicit f for which any ADS protocol must have $\max\{\text{ann. length, space cost}\} = \tilde{\Omega}(C(f))$, where $C(f)$ is space complexity of f in standard streaming model.
- Improved protocol for counting triangles in sparse graphs.
- Extensions to general turnstile stream update model.

Case Study: Frequency Moments

Second Frequency Moment (F_2)

- F_2 is a central streaming problem.
  - Captures sample variance, Euclidean norm, data similarity.

- Definition:
  - Let X be the frequency vector of the stream.
  - $$F_2(X) = \sum_{i=1}^{n} X_i^2$$

Raw data stream over universe {a, b, c, d}

a	**b**	**a**	**c**	**b**	**a**

→

Frequency Vector X

a	b	c	d
3	2	1	0

$$F_2(X) = 3^2 + 2^2 + 1^2 = 14$$

Prior Work

- [CCM09]: ($\sqrt{n}$ annotation, $\sqrt{n}$ space)-protocol for F_2.
- Protocol is more general: applies to any function

$$H(X) = \sum_{i=1}^{n} p(X_i), \text{ where } p \text{ is a polynomial of constant degree.}$$

F_2 Protocol for Sparse Streams

Protocol Overview

- Basic idea: Domain reduction.
  - At start of S, P gives hash function g mapping huge domain $[n]$ to small domain $[r]$. Then P and V run "dense" F_2 protocol on $[r]$.

Protocol Overview

- Basic idea: Domain reduction.
  - At start of S, P gives hash function g mapping huge domain $[n]$ to small domain $[r]$. Then P and V run "dense" F_2 protocol on $[r]$.
  - Many challenges!
    - Ensuring P does not introduce collisions in remapping to cause errors (need a way for V to 'detect' collisions under g).
    - P does not know g in advance, because g depends on the stream.
    - To achieve general (annotation length, space) tradeoffs, need a way for V to avoid storing complete description of g.

Basic Idea: Domain Reduction

- At start of S, P gives hash function g mapping huge domain $[n]$ to small domain $[r]$. Then P and V run "dense" F_2 protocol on "mapped-down" stream over $[r]$.
- P claims g is injective on all items with non-zero frequency in S.
- The larger r, the smaller g's description length.
- But the larger r, the more expensive the dense F_2 protocol.
- We choose r to balance these costs.

Challenge 1: How Can V Check Injectivity?

- Suppose we have r buckets, and a stream S' of updates of the form $(i,b) \in [n] \times [r]$, indicating that item i is inserted into bucket b.
- Call S' an **INJECTION** if no bucket b receives two distinct elements $i \neq j$.
- If V can solve the **INJECTION** problem, V can determine whether g is injective on S.

An Optimal **INJECTION** Protocol

- Solution: Let $X_{(i,b)}$ denote the number of times item i is inserted into bucket b.
- Define three r-dimensional vectors u, v, w via:

$$u_b = \sum_{j \in [n]} X_{(j,b)},$$
$$v_b = \sum_{j \in [n]} X_{(j,b)} \cdot j,$$
$$w_b = \sum_{j \in [n]} X_{(j,b)} \cdot j^2.$$

- Lemma: $\sum_{b \in [r]} v_b^2 = \sum_{b \in [r]} u_b \cdot w_b$ iff the stream is an injection.
- We extend "dense" F_2 protocol to check this equality with ($\sqrt{r}$ annotation, $\sqrt{r}$ space).

Challenge 2: P Does Not Know g In Advance

- How does one construct a hash function g that is injective on a set T with $|T| \leq m$?(cf. [FK84]).
- Step 1: Choose $g_1 : [n] \rightarrow [r]$ at random from a pairwise independent hash family (g_1 requires O(log n) bits to specify).
- Step 2: Append to g_1 a list L of all items in T that collide with any other item, with a special hash value for each.
- In expectation, at most m^2 / r items are involved in a collision, so total description length of g is $O(m^2 \log n / r)$.

"Complete" F_2 Protocol

- P sends only g_1 at start of S.
- While processing S, V runs "dense" F_2 protocol on the "mapped-down" stream, using g_1 as the hash function.
- At end of S, P gives list L of items involved in a collision under g_1, along with their frequencies.
- Assuming L is honestly specified, V can compute these items' contribution to F_2 and **remove them** from the stream.
- g_1 is (claimed to be) injective on the remaining items. V checks this using the INJECTION protocol.
- It remains for V to check that the list L was honestly specified.

MULTI-INDEX Protocol

- Given: A stream S, followed by a list L of items and their claimed frequencies X_i^*.
- Goal: Check whether $X_i = X_i^*$ for all $i \in L$ with cost equal to that of a **single** INDEX query.
- Basic Idea: Let z be the n-dimensional vector such that $z_i = 1$ for all $i \in L$ and $z_i = 0$ otherwise. Enough to check that

$$0 = \sum_{i \in [n]} z_i \cdot (X_i - X_i^*)^2.$$

MULTI-INDEX Protocol

- Enough to check that $0 = \sum_{i \in [n]} z_i \cdot (X_i - X_i^*)^2$.
- Protocol proceeds in "stages". Stage j makes use of a separate pair-wise independent hash function $h_j : [n] \to [r]$.
- Stage j used to check that $0 = \sum_i z_i \cdot (X_i - X_i^*)^2$, where the sum is only over items i "isolated" under h_j, but not under $h_{j'}$ for $j' < j$.
- W.h.p., only $O(1)$ stages needed w.h.p. before all $i \in L$ have been isolated.
- Inductive soundness proof: V can "trust" the results of Stage j as long as she can also trust the results of Stage j+1. Final stage can be trusted directly.

Open Questions

- We gave F_2 protocol with ann. length x and space y for any $x \cdot \sqrt{y} \geq m$. Best lower bound says $x \cdot y = \Omega(m)$. Close this gap.
- Give **any** explicit function for which any ADS protocol must have $\max\{\text{ann. length, space cost}\} = \Omega(N^{1/2+\delta})$, where N is input size.
- Understand the power of **interaction** in streaming verification.
  - [CTY10]: A logarithmic cost protocol for F_2 with $\log n$ rounds of interaction between P and V.
  - [CCMTV14]: A logarithmic cost protocol for INDEX with 2 rounds of interaction between P and V.
  - Is there a logarithmic cost protocol for F_2 with O(1) rounds of interaction? Lower bounds of [CCMTV14] give evidence for "NO".
  - Closely related to long-open questions in communication complexity.

Thank you!