Time lower bounds for nonadaptive turnstile streaming algorithms

Jelani Nelson
Harvard

August 25, 2014

joint work with Kasper Green Larsen (Aarhus) and Nguyễn Lê Huy (Simons Institute)

Turnstile streaming (Muthukrishnan'05)

- Given some family $\mathcal{Q}$ of query functions mapping $\mathbb{R}^n$ to $\mathbb{R}$
- Vector $\mathbf{x}$ starts as $\vec{0}^n$
- m updates each causing some change $\mathbf{x}_i \leftarrow \mathbf{x}_i + v$, $v \in \{-M, \ldots, M\}$

Turnstile streaming (Muthukrishnan'05)

- Given some family $\mathcal{Q}$ of query functions mapping $\mathbb{R}^n$ to $\mathbb{R}$
- Vector $\mathbf{x}$ starts as $\vec{0}^n$
- m updates each causing some change $\mathbf{x}_i \leftarrow \mathbf{x}_i + v$, $v \in \{-M, \ldots, M\}$
- **Goal:** Answer queries $q(\mathbf{x})$, $q \in \mathcal{Q}$, minimize space

Turnstile streaming (Muthukrishnan'05)

- Given some family $\mathcal{Q}$ of query functions mapping $\mathbb{R}^n$ to $\mathbb{R}$
- Vector $\mathbf{x}$ starts as $\vec{0}^n$
- m updates each causing some change $\mathbf{x}_i \leftarrow \mathbf{x}_i + v$, $v \in \{-M, \ldots, M\}$
- **Goal:** Answer queries $q(\mathbf{x})$, $q \in \mathcal{Q}$, minimize space and time

Turnstile streaming (Muthukrishnan'05)

- Given some family $\mathcal{Q}$ of query functions mapping $\mathbb{R}^n$ to $\mathbb{R}$
- Vector $\mathbf{x}$ starts as $\vec{0}^n$
- m updates each causing some change $\mathbf{x}_i \leftarrow \mathbf{x}_i + v$, $v \in \{-M, \ldots, M\}$
- **Goal:** Answer queries $q(\mathbf{x})$, $q \in \mathcal{Q}$, minimize space and time

  update time: time to process (i, v)

  query time: time to answer a query for $q(\mathbf{x})$

Turnstile streaming (Muthukrishnan'05)

- Given some family $\mathcal{Q}$ of query functions mapping $\mathbb{R}^n$ to $\mathbb{R}$
- Vector $\mathbf{x}$ starts as $\vec{0}^n$
- m updates each causing some change $\mathbf{x}_i \leftarrow \mathbf{x}_i + v$, $v \in \{-M, \ldots, M\}$
- **Goal:** Answer queries $q(\mathbf{x})$, $q \in \mathcal{Q}$, minimize space and time

  update time: time to process (i, v)

  query time: time to answer a query for $q(\mathbf{x})$
- Assume Word RAM model: memory is divided into words of w bits each, and reading/writing a word and performing standard C operations take constant time.
- In this talk: m, M are $\mathrm{poly}(n)$ and $w = \log_2 n$.

Turnstile problem examples

- Distinct elements: $q(\mathbf{x}) = |\mathrm{support}(\mathbf{x})|$
- F_p moments: $q(\mathbf{x}) = \|\mathbf{x}\|_p^p$
- Entropy: $q(\mathbf{x}) = -\sum_i \tilde{\mathbf{x}}_i \log_2(\tilde{\mathbf{x}}_i)$, $\tilde{\mathbf{x}}_i = |\mathbf{x}_i|/\|\mathbf{x}\|_1$
- Point query: $q_i(\mathbf{x}) = \mathbf{x}_i$
- ...

Turnstile problem examples

- Distinct elements: $q(\mathbf{x}) = |\mathrm{support}(\mathbf{x})|$
- F_p moments: $q(\mathbf{x}) = \|\mathbf{x}\|_p^p$
- Entropy: $q(\mathbf{x}) = -\sum_i \tilde{\mathbf{x}}_i \log_2(\tilde{\mathbf{x}}_i)$, $\tilde{\mathbf{x}}_i = |\mathbf{x}_i|/\|\mathbf{x}\|_1$
- Point query: $q_i(\mathbf{x}) = \mathbf{x}_i$
- ...

- Typically settle for randomization and approximation, e.g. $q_i(\mathbf{x}) \pm \varepsilon\|\mathbf{x}\|_1$ for point query, or $(1 \pm \varepsilon)q(\mathbf{x})$ for the others succeeds with probability $1 - \delta$
- Often rand/approx are necessary to get sublinear space [Alon, Matias, Szegedy '96]

Space lower bounds

Over time we've gotten good at proving space lower bounds ...

[Munro, Paterson'78]
[Gluskin'82]
[Alon, Matias, Szegedy'96]
[Bar-Yossef, Kumar, Sivakumar'02]
[Bar-Yossef, Jayram, Kumar, Sivakumar'02]
[Chakrabarti, Khot, Sun'03]
[Indyk, Woodruff'03]
[Woodruff'04]
[Feigenbaum, Kannan, McGregor, Suri, Zhang'05]
[Jowhari, Ghodsi'05]
[Grohe, Koch, Schweikardt'05]
[Liben-Nowell, Vee, Zhu'05]
[Chakrabarti, Cormode, McGregor'07]
[Guha, McGregor'07]
[Gopalan, Jayram, Krauthgamer, Kumar'07]
[Guha, Indyk, McGregor'07]
[Beame, Jayram, Rudra'07]
[Gál, Gopalan'07]
[Ganguly, Majumder'07]
[Chakrabarti, Jayram, Pătraşcu'08]
[Ergün, Jowhari'08]
[Guha, McGregor'08]
[Chakrabarti, Cormode, McGregor'08]
[Ganguly'08]
[Jayram, Kumar, Sivakumar'08]
[Gronemeier'09]
...

[Jayram'09]
[Ganguly, Sohler'09]
[Clarkson, Woodruff'09]
[Brody, Chakrabarti'09]
[Jayram, Woodruff'09]
[Do Ba, Indyk, Price, Woodruff'10]
[Kane, N., Woodruff'10]
[Magniez, Mathieu, Nayak'10]
[Chakrabarti, Cormode, Kondapally, McGregor'10]
[Jain, Nayak'10]
[Brody, Chakrabarti, Regev, Vidick, de Wolf'10]
[Jayram, Woodruff'11]
[Jowhari, Saglam, Tardos'11]
[Chakrabarti, Regev'11]
[N., Nguyễn, Woodruff'12]
[Sherstov'12]
[Ganguly'12]
[Molinaro, Woodruff, Yaroslavtsev'13]
[Guruswami, Onak'13]
[Vidick'13]
[Andoni, Nguyễn, Polyanskiy, Wu'13]
[Braverman, Ostrovsky, Vilenchik'13]
[Li, Nguyễn, Woodruff'14]
[Liberty, Mitzenmacher, Thaler'14]
[Cormode, Jowhari'14]
...

Time lower bounds

But not as good at time lower bounds . . .

[Clifford, Jalsenius'11]
[Clifford, Jalsenius, Sach'13]

Time lower bounds

But not as good at time lower bounds ...

[Clifford, Jalsenius'11]
[Clifford, Jalsenius, Sach'13]

- neither is for a turnstile streaming problem
- lower bounds are for query+update (the sum)

We have ZERO time lower bounds in turnstile streaming.

Time lower bounds

But not as good at time lower bounds . . .

[Clifford, Jalsenius'11]
[Clifford, Jalsenius, Sach'13]

- neither is for a turnstile streaming problem
- lower bounds are for query+update (the sum)

We have ZERO time lower bounds in turnstile streaming.

Is the lack of progress due to lack of importance?

Time lower bounds

But not as good at time lower bounds . . .

[Clifford, Jalsenius'11]
[Clifford, Jalsenius, Sach'13]

- neither is for a turnstile streaming problem
- lower bounds are for query+update (the sum)

We have ZERO time lower bounds in turnstile streaming.

Is the lack of progress due to lack of importance?

NO!, e.g. [Thorup, Zhang'04]

1.1.3. Time is critical. Motivating the need for speed, we note that time is critical for many streaming algorithms. For example, we may be analyzing packet headers coming through a high-end Internet router. Such a router uses fast forwarding technology and is therefore hard to keep up with. If we cannot keep up, the information is lost. Asking the router to slow down for the sake of measurements is not an option. One might try investing in faster or specialized hardware for the measurements, but then we could also invest in faster routers.

Computing the hash function is a bottleneck in the streaming algorithm for second moment estimation. While 2-independent hashing may give fewer guarantees for the output, it had the advantage of being an order of magnitude faster than existing methods for 4-independent hashing. However, here we will improve the speed of 4-independent hashing, making it a more viable option in time critical scenarios.

To be very concrete, the application [18] that originally motivated us for this research in 2002 was to monitor packets passing through an OC48 Internet backbone link (2.48 Gigabits per second). At this link speed, it is common for packets to arrive at a rate of more than a million per second, thus leaving us with less than a microsecond per packet. In the worst case, when all packets are of the minimum Internet packet size of 40 bytes (320 bits), the packet arrival rate can be as high as $2.48 \times 10^9/320 = 7.75 \times 10^6$ per second, leaving us with less than 130 nanoseconds per packet. A critical part of the application is to compute the second moment, with the packet key being its single word 32-bit IP address and the packet weight being its size in bytes. The speed-up achieved in this paper (down from 182 to 30 nanoseconds for the 4-independent hashing itself, plus 20 nanoseconds to update the sketch) made the application possible. Since then, computers and routers have both become faster (the experiments reported in this paper are run on newer computers), but it remains true that time is critical if you want to keep up with a high-end router.

Our contribution

Our work: first turnstile update time lower bound for any problem*

Our contribution

Our work: first turnstile update time lower bound for any problem*

- **model:** cell probe (more on this soon)
- **problems:** moments, entropy, ℓ_1 point query, ℓ_1 heavy hitters

Our contribution

Our work: first turnstile update time lower bound for any problem*

- **model:** cell probe (more on this soon)
- **problems:** moments, entropy, ℓ_1 point query, ℓ_1 heavy hitters

*Lower bound only holds against *non-adaptive* algorithms

(every turnstile algorithm ever developed has been non-adaptive)

Non-adaptivity

Definition

A *non-adaptive* streaming algorithm selects some random string r at the beginning, before seeing the stream. Afterward, the memory cells probed during an update (i, v) depend only on i, r.

Non-adaptivity

Definition

A *non-adaptive* streaming algorithm selects some random string r at the beginning, before seeing the stream. Afterward, the memory cells probed during an update (i, v) depend only on i, r.

Note: querying can be adaptive (e.g. dyadic trick for HH).

Non-adaptivity

Definition

A *non-adaptive* streaming algorithm selects some random string r at the beginning, before seeing the stream. Afterward, the memory cells probed during an update (i, v) depend only on i, r.

Note: querying can be adaptive (e.g. dyadic trick for HH).

- Every known turnstile algorithm is a lin. sketch: store $y = Ax$

Non-adaptivity

Definition

A *non-adaptive* streaming algorithm selects some random string r at the beginning, before seeing the stream. Afterward, the memory cells probed during an update (i, v) depend only on i, r.

Note: querying can be adaptive (e.g. dyadic trick for HH).

- Every known turnstile algorithm is a lin. sketch: store $y = Ax$
- **[Li, Nguyễn, Woodruff'14]**: every turnstile streaming algo can be made a a linear sketch (space blowup of $\log n$)

Non-adaptivity

Definition

A *non-adaptive* streaming algorithm selects some random string r at the beginning, before seeing the stream. Afterward, the memory cells probed during an update (i, v) depend only on i, r.

Note: querying can be adaptive (e.g. dyadic trick for HH).

- Every known turnstile algorithm is a lin. sketch: store $y = Ax$
- **[Li, Nguyễn, Woodruff'14]**: every turnstile streaming algo can be made a a linear sketch (space blowup of $\log n$)

  (**one caveat:** transformation doesn't preserve update time)

Non-adaptivity

Definition

A *non-adaptive* streaming algorithm selects some random string r at the beginning, before seeing the stream. Afterward, the memory cells probed during an update (i, v) depend only on i, r.

Note: querying can be adaptive (e.g. dyadic trick for HH).

- Every known turnstile algorithm is a lin. sketch: store $y = Ax$
- **[Li, Nguyễn, Woodruff'14]**: every turnstile streaming algo can be made a a linear sketch (space blowup of $\log n$)

  (**one caveat:** transformation doesn't preserve update time)
- **dichotomy after our contribution:**
  (1) **either** linear sketching is the only possible algorithmic paradigm for the turnstile model, for which our LB applies, **or**
  (2) somewhere out there waiting to be discovered is a radically different way to design turnstile streaming algorithms

Problem details

Recall δ is the failure probability.

- **moments:** output $(1 \pm \varepsilon) \sum_i |\mathbf{x}_i|^p$ ($p > 0$, known)
- **entropy:** output $-(1 \pm \varepsilon) \sum_i \tilde{\mathbf{x}}_i \log_2(\tilde{\mathbf{x}}_i)$, $\tilde{\mathbf{x}}_i = |\mathbf{x}_i| / \|\mathbf{x}\|_1$
- ℓ_1 **point query:** given i, output $\mathbf{x}_i \pm \varepsilon \|\mathbf{x}\|_1$

Problem details

Recall δ is the failure probability.

- **moments:** output $(1 \pm \varepsilon) \sum_i |\mathbf{x}_i|^p$ ($p > 0$, known)
- **entropy:** output $-(1 \pm \varepsilon) \sum_i \tilde{\mathbf{x}}_i \log_2(\tilde{\mathbf{x}}_i)$, $\tilde{\mathbf{x}}_i = |\mathbf{x}_i| / \|\mathbf{x}\|_1$
- ℓ_1 **point query:** given i, output $\mathbf{x}_i \pm \varepsilon \|\mathbf{x}\|_1$

- Common wisdom: first achieve failure probability $1/3$, then reduce to δ via median of parallel instantiations
- Blows up both time and space by $\Theta(\log(1/\delta))$

Problem details

Recall δ is the failure probability.

- **moments:** output $(1 \pm \varepsilon) \sum_i |\mathbf{x}_i|^p$ ($p > 0$, known)
- **entropy:** output $-(1 \pm \varepsilon) \sum_i \tilde{\mathbf{x}}_i \log_2(\tilde{\mathbf{x}}_i)$, $\tilde{\mathbf{x}}_i = |\mathbf{x}_i| / \|\mathbf{x}\|_1$
- ℓ_1 **point query:** given i, output $\mathbf{x}_i \pm \varepsilon \|\mathbf{x}\|_1$

- Common wisdom: first achieve failure probability $1/3$, then reduce to δ via median of parallel instantiations
- Blows up both time and space by $\Theta(\log(1/\delta))$
- This blow-up is provably space-optimal for several problems [Jayram,Woodruff'11]

Problem details

Recall δ is the failure probability.

- **moments:** output $(1 \pm \varepsilon) \sum_i |\mathbf{x}_i|^p$ ($p > 0$, known)
- **entropy:** output $-(1 \pm \varepsilon) \sum_i \tilde{\mathbf{x}}_i \log_2(\tilde{\mathbf{x}}_i)$, $\tilde{\mathbf{x}}_i = |\mathbf{x}_i| / \|\mathbf{x}\|_1$
- ℓ_1 **point query:** given i, output $\mathbf{x}_i \pm \varepsilon \|\mathbf{x}\|_1$

- Common wisdom: first achieve failure probability $1/3$, then reduce to δ via median of parallel instantiations
- Blows up both time and space by $\Theta(\log(1/\delta))$
- This blow-up is provably space-optimal for several problems [Jayram,Woodruff'11]
- **We show**: time also must increase as $\mathrm{polylog}(1/\delta)$

  (in what follows ε is a constant, say $1/10$)

Main result

Our theorem (informal): For $\delta = 1/\mathrm{poly}(n)$, any optimal space algorithm for any problem above (i.e. using $S = O(\log n)$ words of memory) must have update time $t_u = \Omega(\sqrt{\log n / \log\log n})$.

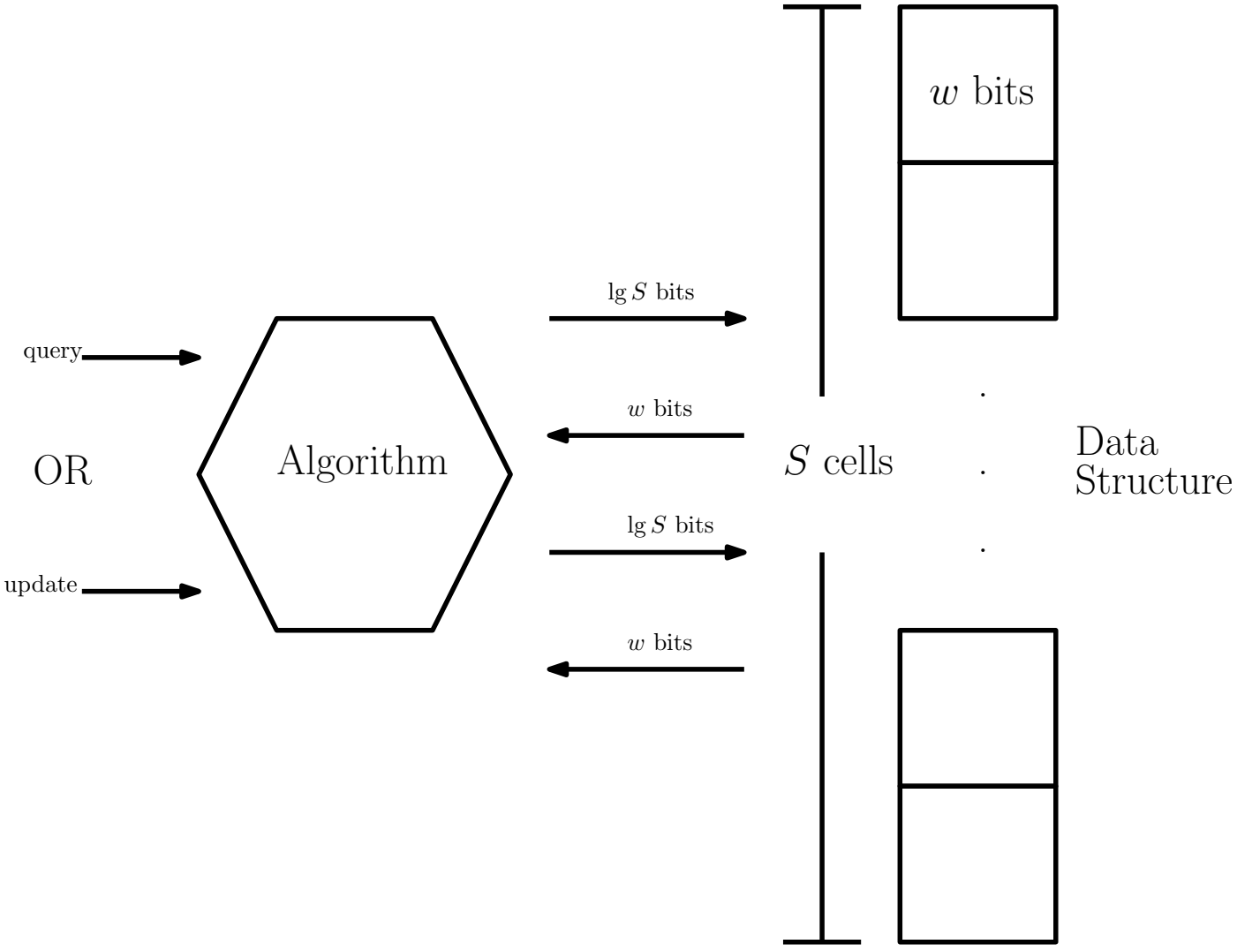
Main result

Our theorem (informal): For $\delta = 1/\mathrm{poly}(n)$, any optimal space algorithm for any problem above (i.e. using $S = O(\log n)$ words of memory) must have update time $t_u = \Omega(\sqrt{\log n / \log\log n})$.

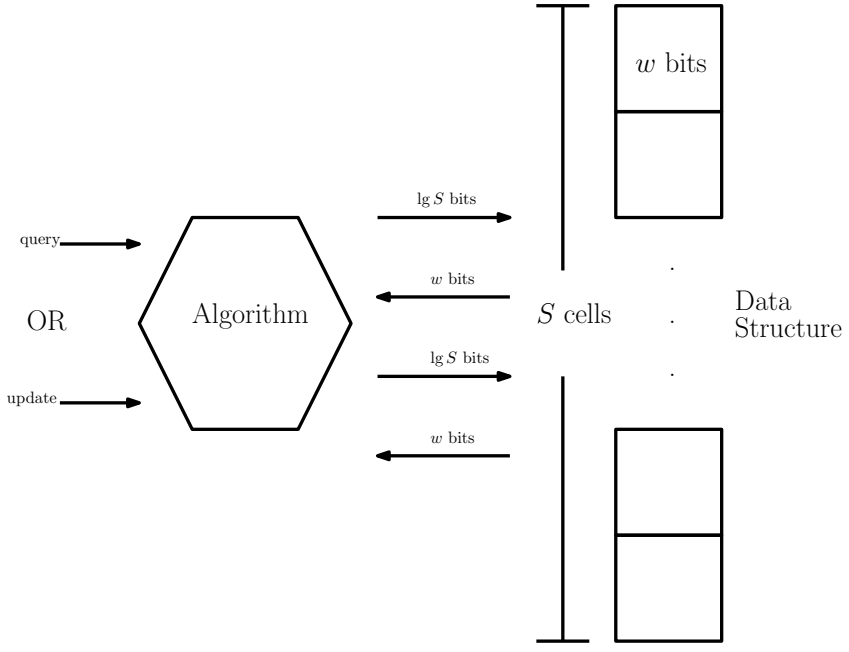
Also, any deterministic ℓ_1 point query algorithm using space ...

- ... $O(\log n)$ has $t_u \geq C \log n$ (matches upper bound of [N., Nguyễn, Woodruff'12])
- ... $O(\log^{1+\gamma} n)$ has $t_u \geq \frac{C \log n}{\log\log n}$ (matches [N., Nguyễn, Woodruff'12]) for $\gamma = 1$)

Cell-probe model [Yao'78]

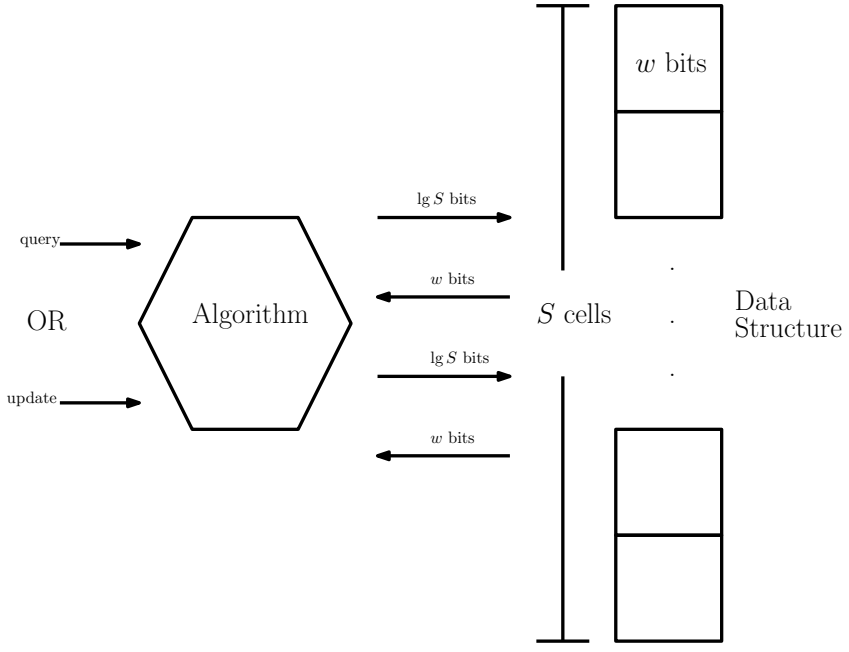


Cell-probe model [Yao'78]



- Runtime is at least number of cells probed
- Communication game: cells probed $=$ number of rounds$/2$
- Want to lower bound number of rounds of communication

Cell-probe model [Yao'78]



- Runtime is at least number of cells probed
- Communication game: cells probed $=$ number of rounds$/2$
- Want to lower bound number of rounds of communication
- Popular model for data structure lower bounds.

Space lower bounds

Space lower bounds via communication complexity

Alice

$\alpha \in \mathcal{X}$

Bob

$\beta \in \mathcal{Y}$

- Alice, Bob know $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$
- Bob needs to compute $f(\alpha, \beta)$
- Communication lower bounds $\Rightarrow$ space LB's [Alon, Matias, Szegedy'96]

Space lower bound: toy example

Consider $\varepsilon = \delta = 0$, $q(\mathbf{x}) = |\mathrm{support}(\mathbf{x})|$ ("distinct elements").

Claim: Space is $\Omega(n)$ bits.

Space lower bound: toy example

Consider $\varepsilon = \delta = 0$, $q(\mathbf{x}) = |\mathrm{support}(\mathbf{x})|$ ("distinct elements").

Claim: Space is $\Omega(n)$ bits.
Proof: Reduction from $\textsc{Equality}$, $f(\alpha, \beta) = \mathbb{1}_{\alpha=\beta}$.

Space lower bound: toy example

Consider $\varepsilon = \delta = 0$, $q(\mathbf{x}) = |\mathrm{support}(\mathbf{x})|$ ("distinct elements").

Claim: Space is $\Omega(n)$ bits.
Proof: Reduction from $\textsc{Equality}$, $f(\alpha, \beta) = \mathbb{1}_{\alpha=\beta}$.

Alice and Bob get α and β in $\{0,1\}^n$, resp., each of Hamming weight $n/2$. Suppose a space-S streaming algo. A exists. For each i with $\alpha_i = 1$, Alice feeds update $(i, 1)$ to A then sends memory contents of A to Bob. Bob continues running A with updates $(i, 1)$ for $\beta_i = 1$. The number of distinct elements is $n/2$ iff $\alpha = \beta$.

Space lower bound: toy example

Consider $\varepsilon = \delta = 0$, $q(\mathbf{x}) = |\mathrm{support}(\mathbf{x})|$ ("distinct elements").

Claim: Space is $\Omega(n)$ bits.
Proof: Reduction from $\textsc{Equality}$, $f(\alpha, \beta) = \mathbb{1}_{\alpha=\beta}$.

Alice and Bob get α and β in $\{0,1\}^n$, resp., each of Hamming weight $n/2$. Suppose a space-S streaming algo. A exists. For each i with $\alpha_i = 1$, Alice feeds update $(i, 1)$ to A then sends memory contents of A to Bob. Bob continues running A with updates $(i, 1)$ for $\beta_i = 1$. The number of distinct elements is $n/2$ iff $\alpha = \beta$.

$\textsc{Equality}$ requires $\Omega(n)$ communication, therefore $S = \Omega(n)$.

Space lower bound: toy example

Consider $\varepsilon = \delta = 0$, $q(\mathbf{x}) = |\mathrm{support}(\mathbf{x})|$ ("distinct elements").

Claim: Space is $\Omega(n)$ bits.
Proof: Reduction from EQUALITY, $f(\alpha, \beta) = \mathbb{1}_{\alpha=\beta}$.

Alice and Bob get α and β in $\{0,1\}^n$, resp., each of Hamming weight $n/2$. Suppose a space-S streaming algo. A exists. For each i with $\alpha_i = 1$, Alice feeds update $(i, 1)$ to A then sends memory contents of A to Bob. Bob continues running A with updates $(i, 1)$ for $\beta_i = 1$. The number of distinct elements is $n/2$ iff $\alpha = \beta$.

EQUALITY requires $\Omega(n)$ communication, therefore $S = \Omega(n)$.

Note: Alice's half of the stream has $a = n/2$ updates.

Our approach: Sketch compression

Theorem (Our work)

If there is a rand. non-adap. streaming algo. with $a \leq \sqrt{n}$, *space* S *(words), error prob.* δ, *and* $t_u \leq (1/2)\log(n/\log(S/t_u))$, *then there is a protocol for the corresponding one-way communication game in which Alice sends*

$$O\left(a \cdot t_u \log(S/t_u) + \log\log(n/a) + t_u w\right)$$

bits to Bob with error probability $O(e^a(eS/t_u)^{t_u a}\delta)$.

Our approach: Sketch compression

Theorem (Our work)

If there is a rand. non-adap. streaming algo. with $a \leq \sqrt{n}$, *space* S *(words), error prob.* δ, *and* $t_u \leq (1/2)\log(n/\log(S/t_u))$, *then there is a protocol for the corresponding one-way communication game in which Alice sends*

$$O\left(a \cdot t_u \log(S/t_u) + \log\log(n/a) + t_u w\right)$$

bits to Bob with error probability $O(e^a(eS/t_u)^{t_u a}\delta)$.

Punchline: a, t_u small $\Rightarrow$ more efficient communication protocol

Our approach: Sketch compression

Theorem (Our work)

If there is a rand. non-adap. streaming algo. with $a \leq \sqrt{n}$, *space* S *(words), error prob.* δ, *and* $t_u \leq (1/2)\log(n/\log(S/t_u))$, *then there is a protocol for the corresponding one-way communication game in which Alice sends*

$$O\left(a \cdot t_u \log(S/t_u) + \log\log(n/a) + t_u w\right)$$

bits to Bob with error probability $O(e^a(eS/t_u)^{t_u a}\delta)$.

Punchline: a, t_u small $\Rightarrow$ more efficient communication protocol

Fine print: Only holds when the streaming problem is "permutation-invariant" (something that holds for all problems I mentioned before).

Our approach: Sketch compression

Theorem (Our work)

If there is a rand. non-adap. streaming algo. with $a \leq \sqrt{n}$, *space* S *(words), error prob.* δ, *and* $t_u \leq (1/2)\log(n/\log(S/t_u))$, *then there is a protocol for the corresponding one-way communication game in which Alice sends*

$$O\left(a \cdot t_u \log(S/t_u) + \log\log(n/a) + t_u w\right)$$

bits to Bob with error probability $O(e^a(eS/t_u)^{t_u a}\delta)$.

Punchline: a, t_u small $\Rightarrow$ more efficient communication protocol

Fine print: Only holds when the streaming problem is "permutation-invariant" (something that holds for all problems I mentioned before).

Next: dig through literature and find existing lower bounds with small a

Case study: ℓ_1 point query

Will reuse lower bound from [Jowhari, Saglam, Tardos'11]

Case study: ℓ_1 point query

Will reuse lower bound from [Jowhari, Saglam, Tardos'11]

Reduction from following problem: Alice gets vector $\alpha \in [n]^a$. Bob gets no input and must recover vector. One round (Alice speaks).

Case study: ℓ_1 point query

Will reuse lower bound from [Jowhari, Saglam, Tardos'11]

Reduction from following problem: Alice gets vector $\alpha \in [n]^a$. Bob gets no input and must recover vector. One round (Alice speaks).

Fano's inequality implies communication must be $\Omega(a \log n)$.

Case study: ℓ_1 point query

Will reuse lower bound from [Jowhari, Saglam, Tardos'11]

Reduction from following problem: Alice gets vector $\alpha \in [n]^a$. Bob gets no input and must recover vector. One round (Alice speaks).

Fano's inequality implies communication must be $\Omega(a \log n)$.

[JST'11] reduction to several point queries

Given: streaming algo. A for ℓ_1 point query, $\varepsilon = 1/1000$

Case study: ℓ_1 point query

Will reuse lower bound from [Jowhari, Saglam, Tardos'11]

Reduction from following problem: Alice gets vector $\alpha \in [n]^a$. Bob gets no input and must recover vector. One round (Alice speaks).

Fano's inequality implies communication must be $\Omega(a \log n)$.

[JST'11] reduction to several point queries

Given: streaming algo. A for ℓ_1 point query, $\varepsilon = 1/1000$

- For each $i \in [a]$ feed update $(\alpha_i, 100^{a-i})$ to A

  send memory contents of A to Bob

Case study: ℓ_1 point query

Will reuse lower bound from [Jowhari, Saglam, Tardos'11]

Reduction from following problem: Alice gets vector $\alpha \in [n]^a$. Bob gets no input and must recover vector. One round (Alice speaks).

Fano's inequality implies communication must be $\Omega(a \log n)$.

[JST'11] reduction to several point queries

Given: streaming algo. A for ℓ_1 point query, $\varepsilon = 1/1000$

- For each $i \in [a]$ feed update $(\alpha_i, 100^{a-i})$ to A

  send memory contents of A to Bob
- To recovery α_i, Bob tries point query on all $j \in [n]$. $\mathbf{x}_j$ big only for $j = \alpha_i$. Bob then updates $(\alpha_i, -100^{a-i})$.

Case study: ℓ_1 point query

Will reuse lower bound from [Jowhari, Saglam, Tardos'11]

Reduction from following problem: Alice gets vector $\alpha \in [n]^a$. Bob gets no input and must recover vector. One round (Alice speaks).

Fano's inequality implies communication must be $\Omega(a \log n)$.

[JST'11] reduction to several point queries

Given: streaming algo. A for ℓ_1 point query, $\varepsilon = 1/1000$

- For each $i \in [a]$ feed update $(\alpha_i, 100^{a-i})$ to A

  send memory contents of A to Bob
- To recovery α_i, Bob tries point query on all $j \in [n]$. $\mathbf{x}_j$ big only for $j = \alpha_i$. Bob then updates $(\alpha_i, -100^{a-i})$.
- Do this in a loop for $i = 1, \ldots, a$.

Case study: ℓ_1 point query

The calculation:

Case study: ℓ_1 point query

The calculation:

- Let's say $\delta = 0$ (A provides deterministic point query)

Case study: ℓ_1 point query

The calculation:

- Let's say $\delta = 0$ (A provides deterministic point query)
- Choose $a = c_0 t_u$ for large constant c_0.
- Sketch compression: for contradiction, $t_u \ll \log n / \log(eS/t_u)$. There exists a protocol with complexity

$$\begin{aligned} & a \cdot t_u \log(eS/t_u) + \log\log(n/a) + t_u w \\ & \quad \leq c_0 t_u^2 \log(eS/t_u) + t_u w + o(t_u \log n) \leq 2 t_u w \end{aligned}$$

Case study: ℓ_1 point query

The calculation:

- Let's say $\delta = 0$ (A provides deterministic point query)
- Choose $a = c_0 t_u$ for large constant c_0.
- Sketch compression: for contradiction, $t_u \ll \log n / \log(eS/t_u)$. There exists a protocol with complexity

$$\begin{aligned} & a \cdot t_u \log(eS/t_u) + \log\log(n/a) + t_u w \\ & \leq c_0 t_u^2 \log(eS/t_u) + t_u w + o(t_u \log n) \leq 2 t_u w \end{aligned}$$

- Communication lower bound (last slide) is $a \log n$

$$\Rightarrow c_1 a \log n \leq 2 t_u w$$

$$\Rightarrow w \geq (c_1 c_0 \log n)/2$$

contradiction (c_0 is large)

Case study: ℓ_1 point query

The calculation:

- Let's say $\delta = 0$ (A provides deterministic point query)
- Choose $a = c_0 t_u$ for large constant c_0.
- Sketch compression: for contradiction, $t_u \ll \log n / \log(eS/t_u)$. There exists a protocol with complexity

$$\begin{aligned} & a \cdot t_u \log(eS/t_u) + \log\log(n/a) + t_u w \\ & \leq c_0 t_u^2 \log(eS/t_u) + t_u w + o(t_u \log n) \leq 2 t_u w \end{aligned}$$

- Communication lower bound (last slide) is $a \log n$

$$\begin{aligned} & \Rightarrow c_1 a \log n \leq 2 t_u w \\ & \Rightarrow w \geq (c_1 c_0 \log n)/2 \\ & \text{contradiction } (c_0 \text{ is large}) \end{aligned}$$

- $S = \Theta(\log n)$ gives $t_u = \Omega(\log n)$
  $S = \Theta(\log^{1+\gamma} n)$ gives $t_u = \Omega(\log n / \log\log n)$

Extensions to other problems/settings

- For $\delta > 0$ (randomized algorithms), still have a lower bound from Fano's inequality, and sketch compression theorem tells us how δ blows up. Just do some more calculations.

Extensions to other problems/settings

- For $\delta > 0$ (randomized algorithms), still have a lower bound from Fano's inequality, and sketch compression theorem tells us how δ blows up. Just do some more calculations.
- For other problems ...

  In the hard instance we covered, heavy hitters $\approx$ point query

Extensions to other problems/settings

- For $\delta > 0$ (randomized algorithms), still have a lower bound from Fano's inequality, and sketch compression theorem tells us how δ blows up. Just do some more calculations.
- For other problems ...

  In the hard instance we covered, heavy hitters $\approx$ point query

  We can reduce from heavy hitters.

  e.g. for moment estimation $q(\mathbf{x}) = \sum_i |\mathbf{x}_i|^p$, moment is dominated by the single heavy hitter.

  We can repeatedly guess index i of the heavy hitter, update $(i, -C^j)$, and re-estimate $\|\mathbf{x}\|_p^p$ (need $\delta \ll 1/n$).

Proof overview of sketch compression

Given: Non-adaptive algo. T with failure prob. δ.

Proof overview of sketch compression

Given: Non-adaptive algo. T with failure prob. δ.

- Consider comm. problem where Alice gets first half of stream D, Bob gets second half D', Bob must compute $(q_j(D \circ D'))_j$.

Proof overview of sketch compression

Given: Non-adaptive algo. T with failure prob. δ.

- Consider comm. problem where Alice gets first half of stream D, Bob gets second half D', Bob must compute $(q_j(D \circ D'))_j$.
- Will show existence of efficient private-coin protocol; by Yao's principle can show deterministic protocol for any distro. μ.

Proof overview of sketch compression

Given: Non-adaptive algo. T with failure prob. δ.

- Consider comm. problem where Alice gets first half of stream D, Bob gets second half D', Bob must compute $(q_j(D \circ D'))_j$.
- Will show existence of efficient private-coin protocol; by Yao's principle can show deterministic protocol for any distro. μ.
- Let $\mu = (\mu_1, \mu_2)$ be distro. on $D \times D'$, π rand. perm. on $[n]$
- Let $(A, B) \sim \mu$, define γ as the distribution $\pi(\mu)$

Proof overview of sketch compression

Given: Non-adaptive algo. T with failure prob. δ.

- Consider comm. problem where Alice gets first half of stream D, Bob gets second half D', Bob must compute $(q_j(D \circ D'))_j$.
- Will show existence of efficient private-coin protocol; by Yao's principle can show deterministic protocol for any distro. μ.
- Let $\mu = (\mu_1, \mu_2)$ be distro. on $D \times D'$, π rand. perm. on $[n]$
- Let $(A, B) \sim \mu$, define γ as the distribution $\pi(\mu)$
- Randomness-fixing: T gives det. protocol for γ with error δ

Proof overview of sketch compression

- Randomness-fixing: T gives det. protocol for γ with error δ

Proof overview of sketch compression

- Randomness-fixing: T gives det. protocol for γ with error δ
- By pigeonhole, there exists $C \subset [S], |C| = t_u$ and set I^C of $n/\binom{S}{t_u}$ indices $i \in [n]$ which only probe cells in C.

  $n/\binom{S}{t_u} > n/(eS/t_u)^{t_u} \geq \sqrt{n} \geq a$.

  Alice+Bob both know I^C since T is deterministic.

Proof overview of sketch compression

- Randomness-fixing: T gives det. protocol for γ with error δ
- By pigeonhole, there exists $C \subset [S], |C| = t_u$ and set I^C of $n/\binom{S}{t_u}$ indices $i \in [n]$ which only probe cells in C.

  $n/\binom{S}{t_u} > n/(eS/t_u)^{t_u} \geq \sqrt{n} \geq a$.

  Alice+Bob both know I^C since T is deterministic.
- Alice+Bob both agree on set of permutations $\{\rho_i\}_{i=1}^{k}$ s.t. for any set of a indices I' that can occur in D, $\exists\ i \in [k]$ s.t.

Proof overview of sketch compression

- Randomness-fixing: T gives det. protocol for γ with error δ
- By pigeonhole, there exists $C \subset [S], |C| = t_u$ and set I^C of $n/\binom{S}{t_u}$ indices $i \in [n]$ which only probe cells in C.

  $n/\binom{S}{t_u} > n/(eS/t_u)^{t_u} \geq \sqrt{n} \geq a$.

  Alice+Bob both know I^C since T is deterministic.
- Alice+Bob both agree on set of permutations $\{\rho_i\}_{i=1}^k$ s.t. for any set of a indices I' that can occur in D, $\exists\ i \in [k]$ s.t.
  1. $\rho_i(j) \in I^C$ for all $j \in I'$
  2. Let $I(A)$ denote the random indices updated in A.Then the prob. that T errors on $\rho_i(A)$, conditioned on $I(A) = I'$, is at most $2e^a(eS/t_u)^{t_u a} \cdot \varepsilon_{I(A)=I'}$ where $\varepsilon_{I(A)=I'}$ is the error prob. of T on γ conditioned on $I(A) = I'$.

Proof overview of sketch compression

Conditions on the ρ_i:

1. $\rho_i(j) \in I^C$ for all $j \in I'$
2. The prob. that T errors on $\rho_i(A)$, conditioned on $I(A) = I'$, is at most $2e^a(eS/t_u)^{t_u a} \cdot \varepsilon_{I(A)=I'}$ where $\varepsilon_{I(A)=I'}$ is the error prob. of T on γ conditioned on $I(A) = I'$.

- **The protocol:** Alice finds ρ_i satisfying the above for $I(A)$ and runs T on $\rho_i(A)$. She then sends i and all memory contents of cells in C to Bob; communication is $\log k + |C| \cdot w$.

Proof overview of sketch compression

Conditions on the ρ_i:

1. $\rho_i(j) \in I^C$ for all $j \in I'$
2. The prob. that T errors on $\rho_i(A)$, conditioned on $I(A) = I'$, is at most $2e^a(eS/t_u)^{t_u a} \cdot \varepsilon_{I(A)=I'}$ where $\varepsilon_{I(A)=I'}$ is the error prob. of T on γ conditioned on $I(A) = I'$.

- **The protocol:** Alice finds ρ_i satisfying the above for $I(A)$ and runs T on $\rho_i(A)$. She then sends i and all memory contents of cells in C to Bob; communication is $\log k + |C| \cdot w$.
- Bob continues running on $\rho_i(B)$. Error probability over μ is at most $2e^a(eS/t_u)^{t_u a} \cdot \mathbb{E}_{I'} \varepsilon_{I(A)=I'} = 2e^a(eS/t_u)^{t_u a}\delta$.

Proof overview of sketch compression

Conditions on the ρ_i:

1. $\rho_i(j) \in I^C$ for all $j \in I'$
2. The prob. that T errors on $\rho_i(A)$, conditioned on $I(A) = I'$, is at most $2e^a(eS/t_u)^{t_u a} \cdot \varepsilon_{I(A)=I'}$ where $\varepsilon_{I(A)=I'}$ is the error prob. of T on γ conditioned on $I(A) = I'$.

- **The protocol:** Alice finds ρ_i satisfying the above for $I(A)$ and runs T on $\rho_i(A)$. She then sends i and all memory contents of cells in C to Bob; communication is $\log k + |C| \cdot w$.
- Bob continues running on $\rho_i(B)$. Error probability over μ is at most $2e^a(eS/t_u)^{t_u a} \cdot \mathbb{E}_{I'} \varepsilon_{I(A)=I'} = 2e^a(eS/t_u)^{t_u a}\delta$.
- Can show desired $\{\rho_j\}_{j=1}^k$ exists by probabilistic method (pick the set at random) for some $k \leq 2e^a(eS/t_u)^{t_u a} a \log(en/a)$.

Conclusion

Open Problems

- Get good lower bounds for non-constant ε
- Randomized lower bound for $S = O(\log n)$ is $t_u = \Omega(\sqrt{\log n / \log\log n})$. Can we get $\Omega(\log n)$?
- Can adaptivity help in the turnstile model?

  (would be amazing if Count{Min/Sketch} were sub-optimal)
- ... or can we get an improved [Li, Nguyễn, Woodruff'14] preserving t_u?
- For deterministic ℓ_1 point query, the best known small-space algorithms have $t_u \gg 1/\varepsilon$ [N., Nguyễn, Woodruff'12]. Meanwhile in the randomized case we have CountMin, with t_u independent of ε. Is there a fundamental separation?