Sublinear Algorithms

Lecture 6

Last time
- Limitations of sublinear-time algorithms
- Yao’s Minimax Principle
  - Example: testing monotonicity
- Communication complexity

Today
- Communication complexity
- Other models of computation

HW1 resubmission, project guidelines
Sign up for project meetings, scribing, grading

10/5/2020

Sofya Raskhodnikova; Boston University
Communication Complexity

A Method for Proving Lower Bounds

[Blais Brody Matulef 11]

Use known lower bounds for other models of computation

Partially based on slides by Eric Blais
(Randomized) Communication Complexity

**Goal:** minimize the number of bits exchanged.

- **Communication complexity of a protocol** is the maximum number of bits exchanged by the protocol.
- **Communication complexity of a function** $C$, denoted $R(C)$, is the communication complexity of the best protocol for computing $C$. 
Example: Set Disjointness $DISJ_k$

Compute $DISJ_k(S,T)$

- **accept** if $S \cap T = \emptyset$
- **reject** otherwise

**Theorem** \([\text{Kalyanasundaram Schmitger 92, Razborov 92}]\)

$$R(DISJ_k) \geq \Omega(k) \text{ for all } k \leq \frac{n}{2}.$$
A lower bound using CC method

Testing if a Boolean function is a k-parity
A Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$ is linear (also called parity) if
$$f(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n$$
for some $a_1, \ldots, a_n \in \{0,1\}$.

- Work in finite field $\mathbb{F}_2$
  - Other accepted notation for $\mathbb{F}_2$: $G\mathbb{F}_2$ and $\mathbb{Z}_2$
  - Addition and multiplication is mod 2
  - $x=(x_1, \ldots, x_n)$, $y=(y_1, \ldots, y_n)$, that is, $x, y \in \{0,1\}^n$
    $$x + y=(x_1 + y_1, \ldots, x_n + y_n)$$

Example:

\[
\begin{array}{c}
001001 \\
011001 \\
\hline
010000
\end{array}
\]

\[
+ \\
010000
\]
A Boolean function \( f: \{0,1\}^n \rightarrow \{0,1\} \) is linear (also called parity) if
\[
f(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n \text{ for some } a_1, \ldots, a_n \in \{0,1\}
\]
\[
\iff \quad f(x_1, \ldots, x_n) = \sum_{i \in S} x_i \text{ for some } S \subseteq [n].
\]

Notation: \( \chi_S(x) = \sum_{i \in S} x_i \).
A function $f : \{0,1\}^n \to \{0,1\}$ is a $k$-parity if

$$f(x) = \chi_S(x) = \sum_{i \in S} x_i$$

for some set $S \subseteq [n]$ of size $|S| = k$. 
Testing if a Boolean Function is a k-Parity

Input: Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$ and an integer $k$

Question: Is the function a $k$-parity or $\varepsilon$-far from a $k$-parity

($\geq \varepsilon 2^n$ values need to be changed to make it a $k$-parity)?

Time:

$O(k \log k)$ [Buhrman, Garcia–Soriano, Matsliah, de Wolf 13]

$\Omega(\min(k, n - k))$ [Blais Brody Matulef 12]

• Today: $\Omega(k)$ for $k \leq n/2$

Today’s bound implies $\Omega(\min(k, n - k))$
Important Fact About Linear Functions

**Fact.** Two different linear functions disagree on half of the values.

- Consider functions $\chi_S$ and $\chi_T$ where $S \neq T$.
  - Let $i$ be an element on which $S$ and $T$ differ (w.l.o.g. $i \in S \setminus T$)
  - Pair up all $n$-bit strings: $(x, x^{(i)})$
    where $x^{(i)}$ is $x$ with the $i^{th}$ bit flipped.
  - For each such pair, $\chi_S(x) \neq \chi_S(x^{(i)})$
    but $\chi_T(x) = \chi_T(x^{(i)})$
    So, $\chi_S$ and $\chi_T$ differ on exactly one of $x, x^{(i)}$.
  - Since all $x$’s are paired up,
    $\chi_S$ and $\chi_T$ differ on half of the values.

**Corollary.** A $k'$—parity function, where $k' \neq k$, is $\frac{1}{2}$-far from any $k$-parity.
Reduction from $\text{DISJ}_{k/2}$ to Testing $k$-Parity

- Let $T$ be the best tester for the $k$-parity property for $\epsilon = 1/2$
  - query complexity of $T$ is $q$ (testing $k$–parity).
- We will construct a communication protocol for $\text{DISJ}_{k/2}$ that runs $T$ and has communication complexity $2 \cdot q(\text{testing } k\text{–parity})$.

$2 \cdot q(\text{testing } k\text{–parity}) \geq R(\text{DISJ}_{k/2}) \geq \Omega(k/2)$ for $k \leq n/2$

$\Downarrow$

$q(\text{testing } k\text{-parity}) \geq \Omega(k)$ for $k \leq n/2$
Reduction from $\text{DISJ}_{k/2}$ to Testing $k$-Parity

**Input:** $S \subseteq [n], |S| = k/2$.
Compute: $f = \chi_S$

**Output T’s answer**

- $T$ receives its random bits from the shared random string.
Analysis of the Reduction

Queries: Alice and Bob exchange 2 bits for every bit queried by $T$

Correctness:

- $h = f + g \pmod{2} = \chi_S + \chi_T \pmod{2} = \chi_{S\Delta T}$
- $|S\Delta T| = |S| + |T| - 2|S \cap T|$

- $|S\Delta T| = \begin{cases} k & \text{if } S \cap T = \emptyset \\ \leq k - 2 & \text{if } S \cap T \neq \emptyset \end{cases}$

- $h$ is $\begin{cases} k-\text{parity} & \text{if } S \cap T = \emptyset \\ k'-\text{parity where } k' \neq k & \text{if } S \cap T \neq \emptyset \end{cases}$

Summary: $q(\text{testing } k\text{-parity}) \geq \Omega(k)$ for $k \leq n/2$
Testing Lipschitz Property on Hypercube

Lower Bound
Lipschitz Property of Functions \( f: \{0,1\}^n \rightarrow \mathbb{R} \)

- A function \( f: \{0,1\}^n \rightarrow \mathbb{R} \) is Lipschitz if changing a bit of \( x \) changes \( f(x) \) by at most 1.

- Is \( f \) Lipschitz or \( \epsilon \)-far from Lipschitz (\( f \) has to change on many points to become Lipschitz)?
  - Edge \( x - y \) is violated by \( f \) if \( |f(x) - f(y)| > 1 \).

Time:
- \( O(n/\epsilon) \), logarithmic in the size of the input, \( 2^n \)
  - [Chakrabarty Seshadhri]

- \( \Omega(n) \) [Jha Raskhodnikova]
# Testing Lipschitz Property

## Theorem

Testing Lipschitz property of functions $f: \{0,1\}^n \rightarrow \{0,1,2\}$ requires $\Omega(n)$ queries.

Prove it.
Summary of Lower Bound Methods

• Yao’s Principle
  – testing membership in $1^*$, sortedness of a list and monotonicity of Boolean functions

• Reductions from communication complexity problems
  – testing if a Boolean function is a $k$-parity
Other Models of Sublinear Computation
Algorithms Resilient to Erasures (or Errors)

- ≤ $\alpha$ fraction of the input is erased (or modified) adversarially before algorithm runs
- Algorithm does not know in advance what’s erased (or modified)
- Can we still perform computational tasks?
Property Testing

Property Tester [Rubinfeld Sudan 96, Goldreich Goldwasser Ron 98]

Two objects are at distance $\varepsilon = \text{they differ in an } \varepsilon \text{ fraction of places}$
Property Testing with Erasures

Property Tester [Rubinfeld Sudan 96, Goldreich Goldwasser Ron 98]

- YES
- \(\varepsilon\) far from YES
- Accept with probability \(\geq 2/3\)
- Don’t care
- Reject with probability \(\geq 2/3\)

Erasure-Resilient Property Tester [Dixit Raskhodnikova Thakurta Varma 16]

- \(\leq \alpha\) fraction of the input is erased adversarially
- Can be completed to YES
- \(\varepsilon\) any completion is far from YES
- Accept with probability \(\geq 2/3\)
- Don’t care
- Reject with probability \(\geq 2/3\)

Two objects are at distance \(\varepsilon = \) they differ in an \(\varepsilon\) fraction of places
Can We Make Testers $\alpha$-Erasure-Resilient?

It is easy if a tester makes only uniform queries (and the property is extendable).

- Use the original tester as black box and ignore erasures:

\[ O \left( \frac{1}{1-\alpha} \right) \] factor query complexity overhead for all $\alpha \in (0,1)$.

- Applies to many properties
  - Monotonicity over poset domains
    [Fischer Lehman Newman Raskhodnikova Rubinfeld Samorodnitsky 02]
  - Convexity of black and white images
    [Berman Murzabulatov Raskhodnikova 16]
  - Boolean arrays having at most $k$ alternations in values
  - ...
Example: Testing sortedness of $n$-element arrays

- Every uniform tester requires $\Omega(\sqrt{n})$ queries.
- [EKKRV00] (optimal) tester that makes $O(\log n)$ queries

Can we make it erasure-resilient $O\left(\frac{1}{1-\alpha}\right)$ factor overhead?

All known optimal sortedness testers [EKKRV00, BGJRW09, CS13a] break with just one erasure.

Known optimal testers for monotonicity, Lipschitz property and convexity of functions [GGLRS00, DGLRRS99, EKKRV00, F04, CS13a, CS13b, CST14, BRY14, BRY14, CDST15, KMS15, BB16, JR13, CS13a, BRY14, BRY14, CDJS15, PRR03, BRY14] break on a constant number of erasures.
Erasure-Resilient Sortedness Tester

Input: $\varepsilon, \alpha \in (0,1)$; query access to an array

1. Repeat $\Theta(1/\varepsilon)$ times:
   a. Sample uniformly until you get a nonerased search point $s$.
   b. Binary search for $s$ with uniform nonerased split points.
   c. Reject if there are violations along the search path.

2. Accept if no violations were found.
Analysis of the Sortedness Tester

1. Array is sorted \(\implies\) tester accepts
2. Array is \(\varepsilon\)-far from sorted \(\implies\) one iteration rejects with probability \(\geq \varepsilon\)
   - Need to repeat only \(\Theta(1/\varepsilon)\) times to get error probability 2/3
3. Want to show: expected # of queries per iteration is \(O\left(\frac{\log n}{1-\alpha}\right)\)
   - Tester traverses a uniformly random search path in a random binary search tree.
   - The # of levels in a random binary search is \(O(\log n)\) w.h.p.

Claim. Expected # of queries to one level of binary search is

\[O\left(\frac{1}{1-\alpha}\right)\]
**Expected Number of Queries in One Iteration**

At level $k$

\[ Q = \# \text{ of queries} \]

Pr [search point $s$ is in $I$] = \( \frac{\# \text{ nonerased points in I}}{\text{total } \# \text{ nonerased points}} \leq \frac{|I|(1-\alpha_I)}{n(1-\alpha)} \)

\[
E[Q] = \sum_{\text{intervals } I \text{ in level } k} E[Q \mid s \in I] \cdot \Pr[s \in I]
\]

\[
= \sum_{I} \frac{1}{1-\alpha_I} \cdot \frac{|I|(1-\alpha_I)}{n(1-\alpha)} \leq \frac{1}{1-\alpha}
\]
What We Proved

- [Dixit Raskhodnikova Thakurta Varma 16]

**Theorem**

Our $\alpha$-erasure-resilient $\varepsilon$-tester for sortedness of $n$-element arrays makes $O\left(\frac{\log n}{\varepsilon (1-\alpha)}\right)$ queries for all $\alpha, \varepsilon \in (0,1)$. 
Property Testing with Erasures

**Property Tester** [Rubinfeld Sudan 96, Goldreich Goldwasser Ron 98]

- **YES**
- **far from YES**
- $\varepsilon$

- Accept with probability $\geq 2/3$
- Don’t care
- Reject with probability $\geq 2/3$

**Erasure-Resilient Property Tester** [Dixit Raskhodnikova Thakurta Varma 16]

- $\leq \alpha$ fraction of the input is erased adversarially

- **CAN BE COMPLETED TO YES**
- $\varepsilon$

- Accept with probability $\geq 2/3$
- Don’t care
- Reject with probability $\geq 2/3$

- **ANY COMPLETION IS FAR FROM YES**
- $\varepsilon$

- Accept with probability $\geq 2/3$
- Don’t care
- Reject with probability $\geq 2/3$

Two objects are at distance $\varepsilon$ = they differ in an $\varepsilon$ fraction of places.
Property Testing with Errors

Property Tester [Rubinfeld Sudan 96, Goldreich Goldwasser Ron 98]

Tolerant Property Tester [Parnas Ron Rubinfeld 06]

• \( \leq \alpha \) fraction of the input is wrong

Two objects are at distance \( \varepsilon \) = they differ in an \( \varepsilon \) fraction of places
Property Testing with Errors

**Property Tester** [Rubinfeld Sudan 96, Goldreich Goldwasser Ron 98]

- **YES**
- far from **YES**

Accept with probability $\geq 2/3$
Don’t care
Reject with probability $\geq 2/3$

**Tolerant Property Tester** [Parnas Ron Rubinfeld 06]

- $\leq \alpha$ fraction of the input is wrong

- **YES**
- far from **YES**

Accept with probability $\geq 2/3$
Don’t care
Reject with probability $\geq 2/3$

Two objects are at distance $\varepsilon = \text{they differ in an } \varepsilon \text{ fraction of places}$
Relationships Between Models

Containments are strict:

- [Fischer Fortnow 05]: standard vs. tolerant
- [Dixit R Thakurta Varma 16]: standard vs. erasure-resilient
- [R Ron-Zewi Varma 19]: erasure-resilient vs. tolerant
**Distance Approximation for Boolean Functions**

[Paras Ron Rubinfeld 06]

**Goal:** Output $\text{dist}(f, \mathcal{P}) \pm \epsilon$ in sublinear time
Sublinear-Time “Restoration” Models

Local Decoding
Input: A slightly corrupted codeword
Requirement: Recover individual bits of the closest codeword with a constant number of queries per recovered bit.

Program Checking
Input: A program $P$ computing $f$ correctly on most inputs.
Requirement: Self-correct program $P$: for a given input $x$, compute $f(x)$ by making a few calls to $P$.

Local Reconstruction
Input: Function $f$ nearly satisfying some property $P$
Requirement: Reconstruct function $f$ to ensure that the reconstructed function $g$ satisfies $P$, changing $f$ only when necessary. For each input $x$, compute $g(x)$ with a few queries to $f$. 
Generalization: Local Computation

[Rubinfeld Tamir Vardi Xie 2011]

• Compute the $i$-th character $y_i$ of a legal output $y$.

• If there are several legal outputs for a given input, be consistent with one.

• Example: maximal independent set in a graph.
Sublinear-Space Algorithms

What if we cannot get a sublinear-time algorithm?
Can we at least get sublinear space?

Note: sublinear space is broader (for any algorithm, space complexity $\leq$ time complexity)
**Data Stream Model**

**Motivation:** internet traffic analysis

Model the **stream** as \( m \) elements from \([n]\), e.g.,

\[ \langle x_1, x_2, \ldots, x_m \rangle = 3, 5, 3, 7, 5, 4, \ldots \]

**Goal:** Compute a function of the stream, e.g., median, number of distinct elements, longest increasing sequence.

Based on Andrew McGregor's slides: http://www.cs.umass.edu/~mcgregor/slides/10-jhu1.pdf
Streaming Puzzle

A stream contains \( n - 1 \) distinct elements from \([n]\) in arbitrary order.

**Problem:** Find the missing element, using \( O(\log n) \) space.
Conclusion

Sublinear algorithms are possible in many settings

• simple algorithms, more involved analysis
• nice combinatorial problems
• unexpected connections to other areas
• many open questions

In the remainder of the course, we will cover research papers in the area.