Sublinear Algorithms January 26, 2012

Lecture 6

Lecturer: Sofya Raskhodnikova *Scribe(s): Ishan Behoora*

1 Introduction

In today's lecture, we will talk about proving lower bounds for sublinear algoritms by utilizing known lower bounds from other models in theoretical computer science.

2 Reductions using Randomized Communication Complexity

One of the models we can use for such reductions is randomized communication complexity. This is based on a paper by Blais et al. [1]

2.1 Randomized Communication Complexity model

In the randomized communication complexity model, there are two parties Alice and Bob. Alice has a string x and Bob has a string y. Additionally both Alice and Bob has access to a shared random string and the goal is to compute a desired function $C(x, y)$ while minimizing the number of bits exchanged between them. Here the communication complexity of a given protocol is defined as the maximum number of bits exchanged by the protocol and the communication complexity of a function C, denoted $R(C)$, is the communication complexity of the best protocol for computing C.

Note that this model has no cryptographic considerations as both parties just want to compute C and thus this can trivially be achieved using at most $min(x, y)$ complexity. (One sided variants of this model also exist where Alice can send as many bits as she wants and Bob has to try and minize his bits sent but we wil not consider those in this lecture.)

We will use reductions to known lower bounds in this Randomized Communication Complexity (RCC) model to prove lower bounds for sublinear algorithms. An important aspect of such reductions is that they are unconditional and are thus not dependant on conjectures such as $P! = NP$. We now illustrate the technique using the following example problems.

2.2 Set Disjointness $DISJ_k$ in RCC model

An example problem from the RCC model is when both Alice and Bob are given two k bit strings and they have to determine if their strings are disjoint. Formally Alice has string S where $|S| \subseteq [n], |S| = k$ and Bob has string T where $|T| \subseteq [n], |T| = k$ and they have to compute

$$DISJ_k = \begin{cases} ACCEPT & \text{if } S \cap T = \emptyset \\ REJECT & \text{otherwise.} \end{cases}$$

This problem has a known complexity lower bound by Hastad et al.[2]

Theorem 1. $R(DISJ_k) \geq \Omega(k), \forall k \geq \frac{n}{2}$

2.3 Sublinear testing k-parity of boolean functions

We will use the previously known lower bound for Set Disjointness to prove lower bounds for sublinearly testing k-parity of boolean functions. However before we proceed we need to review some definitions. We start with linear functions. For us it is sufficient to consider linear functions over the finite field $\mathbb{F}_2$.

Definition 2. *A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is linear if $f(x_1, ..x_n) = a_1x_1 + a_2x_2..a_nx_n$ where $a_1, a_2...a_n \in \{0,1\}$*

Notice that there is no free term. This is the because the problems are equivalent for testing linearity. Also, Given the definition of a linear function f it is easy to see an equivalanet way to express f as $\mathcal{X}_s(x)$.

Claim 3. *$f(x_1, ..x_n) = \mathcal{X}_s(x)$ where $\mathcal{X}_s(x) = \sum_{i \in S} x_i$ for some $S \subseteq [n]$*

We can now define k-parity functions.

Definition 4. *A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is k-parity if $f(x_1, ..x_n) = \mathcal{X}_s(x) = \sum_{i \in S} x_i$ for some $S \subseteq [n]$ of size $|S| = k$*

With this we ask the question given as input a boolean function $f\{0,1\}^n \to \{0,1\}$ and an integer k is f k-parity or ϵ-far from a k-parity function. A paper by Chakraborty et al. [3] showed that this can be determined in $O(klogk)$ and Blais et al. [1] proved a lower bound of $\Omega(min(k, n-k))$.

In today's lecture we will prove the bound $\Omega(k)$ for $k \leq \frac{n}{2}$. For this we will use the following.

Claim 5. *Two different linear functions $\mathcal{X}_S(x)$ and $\mathcal{X}_T(x)$ over $x \in \{0,1\}^n$ such that $S \neq T$ differ on half of the values of x.*

Proof. Let x be an element on which $\mathcal{X}_S$ and $\mathcal{X}_T$ differ. We can assume W.L.O.G $x \in S \setminus T$.

We can pair the strings x,x^i where x^i is x with the i^{th} bit flipped. For each such pair $\mathcal{X}_S(x) \neq \mathcal{X}_S(x^i)$ but $\mathcal{X}_T(x) = \mathcal{X}_T(x^i)$ so $\mathcal{X}_S$ and $\mathcal{X}_T$ differ on exactly one of x, x^i. Since all x's are paired up $\mathcal{X}_S$ differs from $\mathcal{X}_T$ on half of the values. □

A corollary follows

Corollary 6. *A k^*-parity function where $k^* \neq k$ is $\frac{1}{2}$-far from a k-parity function*

2.4 Reduction from $DISJ_{\frac{k}{2}}$ to Testing k-Parity

At this point, we are in a position to construct the required reduction from set disjointness to testing k-parity. Asssuming B is the best tester for the k-parity property for $\epsilon = \frac{1}{2}$, let the query complexity of B be q(test k-parity)

Claim 7. *$2.q$(test k-parity) $\geq R(DISJ_{\frac{k}{2}})$*

Proof. .

Given the best tester B

Alice has string S where $|S| \subseteq [n], |S| = \frac{k}{2}$

Bob has string T where $|T| \subseteq [n], |T| = \frac{k}{2}$.

$|T|$ and $|S|$ correspond to linear functions $\mathcal{X}_S$ and $\mathcal{X}_T$ using 3. Call $\mathcal{X}_S$ f and $\mathcal{X}_T$ g.

.

Both Alice and Bob can get a random $x \in \{0,1\}^n$ using their common access to the random stream.

Now for each such x Alice generates $f(x)$ and sends it to Bob who then calculates $h(x) = (g(x)+f(x))($ mod 2) and returns it to Alice. Alice passes $h(x)$ to the tester B.

Now $h = f + g($ mod 2$) = \mathcal{X}_S + \mathcal{X}_T($ mod 2$) = \mathcal{X}_{S \Delta T}$

$|S \Delta T| = |S| + |T| - 2|S \cap T|$

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

$$\text{Thus } h = \begin{cases} k\text{-parity} & \text{if } S \cap T = \emptyset \\ k^*\text{-parity where } k* \neq k & \text{if } S \cap T \neq \emptyset \end{cases}$$

As a result using 6 we can use the tester B to determine if h is $\frac{1}{2}$-far from k-parity and thus if S and T are disjoint. Now since Alice and Bob exchanged 2 bits for every bit queried by B hence $2.q$(test k-parity) $\geq R(DISJ_{\frac{k}{2}})$.

□

We can combine this with known lower bound for $DISJ_k$ using 1 and obtain the required lower bound

Theorem 8. $2.q(\textit{test k-parity}) \geq R(DISJ_{\frac{k}{2}}) \geq \Omega(\frac{k}{2})$

2.5 Other applications

Another area where this method can be applied is to to determine if a function f is Lipschitz or ϵ far from Lipschitz. Jha et al. [4] showed that this be achieved in time $O(\frac{n^2}{\epsilon})$ and has a lower bound of $\Omega(n)$.

3 Other interesting models

There are a variety of other models which arise when we investigate sublinear algorithms in more detail. These include.

3.1 Tolerant property testers

Tolerant property testers were introduced by Rubinfeld et al. [5]. Whereas a normal randomnized algorithm can be viewed as

$$RANDOM - ALGORITHM = \begin{cases} YES & \text{Accept with probability} \geq \frac{2}{3} \\ NO & \text{Reject with probability} \geq \frac{2}{3} \end{cases}$$

A model more suited to real world applications where we would also like to know how close or how far we are from a desired property is

$$TOLERANT - TESTER = \begin{cases} YES & \text{Accept with probability} \geq \frac{2}{3} \\ \delta \text{ close to YES} & \text{Accept with probability} \geq \frac{2}{3} \\ \epsilon \text{ far from YES} & \text{Reject with probability} \geq \frac{2}{3} \\ OTHERWISE & \text{Don't Care} \end{cases}$$

3.2 Sublinear-time restoration models

These include the following:

3.2.1 Local Decoding

Here our input is a slightly corrupted codeword and we want to recover individual bits of the closest codeword with a constant number of queries per recovered bit.

3.2.2 Program Checking

Here we are given a program P computing f correctly on most inputs and we need to self-correct program P by computing $f(x)$ correctly for a given x by making a few calls to P.

3.2.3 Local Reconstruction

We are given a function f nearly satisfying some property P and we need to reconstruct function g to ensure that g satisfies P, changing f only when necessary. For each input x, compute $g(x)$ with a few queries to f. An example of this could be that we want f to be Lipschitz. This can be very useful because Lipschitz functions need to have very little noise added to make them differentially private.

3.2.4 Local Computation

A generalization of this by Rubinfeld et al. [6] is to compute the i^{th} character y_i of a legal output y.If there are several legal outputs for a given input we should be consistent with one of them.

3.3 Sublinear-space models

Sometimes we may not be able to find sublinear time algorithms. Then the question arises can we find sublinear space algorithms? We explore this because sublinear space is broader. (For any algorithm space complexity $\leq$ time complexity).

3.3.1 Data Stream Model

A real world example of this arises from internet traffic analysis. While processing data from the internet the data appears as a stream which can be thought of as m elemetns from $[n]$ appearing one by one and our goal is to compute some function of this stream such as median or number of distinct elements. However given that m can be very large, we often have to quickly process each element with limited working memory and quickly produce output "on the fly".

3.3.2 Sampling from a stream of unknown length

An interesting problem which stems from this is given a stream of x_i's of unknown length m how can we find an uniform sample s. For this we can use the following algorithm

Algorithm 1: Sampling algorithm

Set $s \rightarrow x_1$
On seeing the t^{th} element, $s \rightarrow x_t$ with probability $\frac{1}{t}$

This is correct because the probability $s = x_i$ at some time $t \geq i$ is
$Pr[s = x_i] = \frac{1}{i}(1 - \frac{1}{i+1})...(1 - \frac{1}{t}) = \frac{1}{t}$
thus we obtain the required sample s.

4 Conclusion

What we notice is sublinear algorithms arise in many settings. They may be simple algorithms but have a more involved analysis. They can also lead to nice combinatorial problems. Also there are many unexpected connections to other areas and the field has many open questions which can lead to future research avenues.

References

[1] Eric Blais, Joshua Brody, Kevin Matulef, *Property Testing Lower Bounds via Communication Complexity.*, Electronic Colloquium on Computational Complexity (ECCC) (2011), 18-45.

[2] Johan Hstad, Avi Wigderson, *The Randomized Communication Complexity of Set Disjointness*, Theory of Computing 3(1) (2007), 211-219.

[3] Sourav Chakraborty, David Garca-Soriano, Arie Matsliah, *Nearly Tight Bounds for Testing Function Isomorphism*, SODA (2011), 1683-1702.

[4] Madhav Jha, Sofya Raskhodnikova, *Testing and Reconstruction of Lipschitz Functions with Applications to Data Privacy.*, FOCS (2011), 433-442.

[5] Michal Parnas, Dana Ron, Ronitt Rubinfeld, *Tolerant Property Testing and Distance Approximation* , Electronic Colloquium on Computational Complexity (ECCC) (2004), 010.

[6] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, Ning Xie, *Fast Local Computation Algorithms*, ICS (2011), 223-238.