CS 59000 CTT Current Topics in Theoretical CS Oct. 25, 2012

Lecture 20

Lecturer: Elena Grigorescu Scribe: Vaishnavi Chandrasekaran, Pawan Nagesh, Abram Magner

1 Introduction

The focus of today's lecture will be Yao's principle used to show the lower bounds on the cost of a randomized algorithm. The cost could be measured in terms of time complexity, space complexity or any other quantity that describes an algorithm. We will see that in order to compute lower bounds on the cost of a randomized algorithm, it is enough to look at deterministic algorithms over a distribution of inputs. We will use Yao's principle to show a lower bound on testing LDPC codes, and we will make use of properties of expander graphs and codes that we have seen in previous lectures.

2 Yao's Principle

Roughly speaking, the principle says that to compute a lower bound for the cost of a randomized algorithm, it is sufficient to look at the complexities of deterministic algorithms with inputs drawn from a distribution. A high average-case complexity for all deterministic algorithms leads to a high randomized complexity for any randomized algorithm.

Fact 1 *Any randomized algorithm R that terminates and flips t bits is equivalent to a distribution $\mathcal{R}$ over deterministic algorithms $\{R_{r_1,r_2,\ldots,r_t}\}$. That is, each deterministic algorithm over which $\mathcal{R}$ is distributed is R with the values of the random bits fixed. The distribution $\mathcal{R}$ can be visualized as a binary tree, where each internal node represents a coin flip: the left subtree represents the execution of the randomized algorithm with the bit set to* 0*, while the right represents the execution with the bit set to* 1*.*

The statement of Yao's principle requires some notation. For any deterministic algorithm A and input x, let $\text{cost}(A, x)$ be some real-valued measure of the cost of A on x. This could be, for instance, the running time of A or the number of queries to x by A (if x is viewed as a function).

Now, in the context of a problem $\mathcal{P}$ with inputs over some finite set X, with $\mathcal{A}$ being a finite set of deterministic algorithms that solve $\mathcal{P}$ and terminate, we define some other cost functions in terms of the basic one.

Definition 2 (Cost of a randomized algorithm on a given input)

$$\text{cost}(R, x) = \text{cost}(\mathcal{R}, x) = \mathbb{E}_{A \leftarrow \mathcal{R}}[\text{cost}(A, x)].$$

Definition 3 (Cost of a deterministic algorithm on a distribution over inputs)

$$\text{cost}(A, \mathcal{D}) = \mathbb{E}_{x \leftarrow \mathcal{D}}\text{cost}(A, x).$$

These cost definitions lead to different definitions for measures of complexity of $\mathcal{P}$.

Definition 4 (Randomized complexity of a problem) *The* randomized complexity *of $\mathcal{P}$ is given by*

$$\min_{\mathcal{R}} \max_{x} \text{cost}(\mathcal{R}, x).$$

Informally, it is the cost of the best randomized algorithm on its worst input.

Definition 5 (Distributional complexity of a problem) *The* distributional complexity *of $\mathcal{P}$ is given by*

$$\max_{\mathcal{D}} \min_{A \in \mathcal{A}} \text{cost}(A, \mathcal{D}).$$

I.e., it is the cost of the worst input distribution for the best deterministic algorithm in the set we're considering.

Now we can state Yao's principle formally.

Theorem 6 (Yao's principle) *The randomized complexity and distributional complexity of $\mathcal{P}$ are the same:*

$$\min_{\mathcal{R}} \max_{x \in X} \text{cost}(\mathcal{R}, x) = \max_{\mathcal{D}} \min_{A \in \mathcal{A}} \text{cost}(A, \mathcal{D}).$$

Now, for any specific distribution $\mathcal{R}$, clearly, its worst-case cost must be greater than or equal to the minimum possible. Similarly, for any specific input distribution $\mathcal{D}$, its cost for the best deterministic algorithm for it must be less than or equal to the maximum possible. Thus, for any $\mathcal{R}, \mathcal{D}$,

$$\max_{x \in X} \text{cost}(\mathcal{R}, x) \geq \min_{A \in \mathcal{A}} \text{cost}(A, \mathcal{D}).$$

This is the result that we use to prove lower bounds on the cost of a randomized algorithm. The strategy is to find a $\mathcal{D}$ for which any algorithm in $\mathcal{A}$ has high cost.

3 Background on LDPC Codes

We review here the definition and properties of low density parity check codes, as testing membership in them provides a good example application of Yao's principle.

An $[n, k, d]$-LDPC code is a binary linear block code of length n and whose $n \times (n - k)$ parity check matrix H is sparse, in the sense captured by the following definition.

Definition 7 (LDPC matrix, code) *H is an LDPC matrix if each column has $\leq c$ 1s, for some constant c.* code(H) *is then an LDPC code.*

Recall that any linear code code(H) can be seen as a bipartite graph, where the left vertices correspond to a codeword (i.e., the ith vertex corresponds to the ith coordinate of the codeword), and the right vertices correspond to the columns of H. H is then the adjacency matrix of the graph. A (d, c)-regular code is a code whose left degree is $\leq d$ and whose right degree is $\leq c$. An LDPC code can be chosen by constructing a random (d, c)-regular code.

4 Testing Membership in Linear Codes

In order to be able to apply Yao's principle, we need to restrict the types of deterministic algorithms for membership testing that we consider. In what follows we will be often view words (and codewords) as functions as we have previously done it many times.

Definition 8 (Linear test) *A* linear test *for membership of input f in a linear code of length $n = 2^\ell$ is a test that picks, for some t, $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_2^\ell$ and computes*

$$S = \sum_{i=1}^{t} f(\alpha_i).$$

It accepts if and only if $S = 0$.

The importance of linear tests is that they play a role similar to canonical testers in graph property testing:

Theorem 9 (Linear tests are canonical) *For any test of membership in a linear code, there exists a linear tester with the same query complexity.*

5 A Query Complexity Lower Bound for LDPC Code Membership

As an example application of Yao's principle, we shall derive a lower bound for the query complexity of testing membership in low density parity check codes. This was originally proven in [1].

Theorem 10 *There exist $c, d \in \mathbb{Z}$ and constants $\delta > 0$, $\epsilon > \frac{1}{2}$, $\gamma > 0$, such that for all large enough n, there exists a (d, c)-regular LDPC code of block length n such that every linear $(k, 0, \epsilon, \delta)$-test has query complexity $k > \delta \cdot n$. I.e., any test must query "almost all" of the input.*

We will only prove the theorem as stated, but it should be noted that this holds for most (d, c)-regular graphs for any choice of d, c (in the sense that it holds with high probability as $n \to \infty$.)

Proof First, note that, for any test of membership in a linear code, there exists a linear test with the same query complexity. Thus, it suffices for us to consider only linear tests.

We will need some notation. We will consider $n = 2^\ell$, and since we are testing membership in codes whose codewords have length n define the set of inputs to be the set $X = \{f \in \{0,1\}^n\}$, where we think of a function $f \in X$ as $f : \mathbb{F}_2^\ell \to \{0,1\}$. Fix some LDPC code $C = \text{code}(H) \subset \{0,1\}^n$. Define $X_\delta = \{f | d_H(f, C) \geq \delta\}$. I.e., X_δ is the set of X elements that are δ-far from the code. Our hard distribution for deterministic algorithms will be supported only on X_δ.

Let $\mathcal{A}$ be the set of deterministic algorithms that are linear tests for C. Let $\mathcal{A}_k$ be the set of deterministic linear tests for C that make k queries, and define $\text{cost}(A, f)$ to be 1 if A accepts f and 0 otherwise.

Now, the idea shall be as follows. Yao's principle tells us that, for any distributions $\mathcal{D}$ on X_δ and $\mathcal{R}$ on $\mathcal{A}_k$,

$$\max_{x \in X_\delta} \text{cost}(R, x) \geq \min_{A \in \mathcal{A}_k} \text{cost}(A, \mathcal{D}).$$

Plugging in the definitions of the cost functions,

$$\begin{aligned} &\text{cost}(R, f) = \mathbb{E}_{A \leftarrow \mathcal{R}}(A, f) = \Pr[R \text{ accepts } f] \\ &\text{cost}(A, \mathcal{D}) = \mathbb{E}_{f \leftarrow \mathcal{D}}(A, f) = \Pr[A \text{ accepts } f] \\ &\implies \max_{f \in X_\delta} \Pr[R \text{ accepts } f] \geq \min_{A \in \mathcal{A}_k} \Pr_{f \leftarrow \mathcal{D}}[A \text{ accepts } f]. \end{aligned}$$

Thus, our approach shall be to find a distribution $\mathcal{D}$ that fools every deterministic algorithm that makes only k queries into accepting. This forces R to accept on the worst-case input in this set with large probability, which means that R cannot be a property tester.

To define the distribution, consider, for some H, code(H). Define H_{-i} to be H with the ith column removed. I.e., it is H with the ith constraint removed. Since the members of code(H_{-i}) need to satisfy fewer constraints, code$(H) \subseteq$ code(H_{-i}) for all i.

Define $B_i = \text{code}(H_{-i}) - \text{code}(H)$. B_i is the set of all bit strings that satisfy all constraints imposed by H except for the ith one. Finally, let

$$B = \bigcup_i B_i. \text{ (as a multiset.)}$$

Then $\mathcal{D}$ we define to be the distribution induced by a uniform choice over B. Equivalently, we can define $\mathcal{D}$ as the distribution induced by picking uniformly at random and i, then a random codeword from B_i.

To proceed, we'll need a few lemmata.

Lemma 11 *For c, d large enough, there exists δ such that with high probability over random (d, c) LDPC codes, every $f \in B$ is δ-far from C.*

Intuitively, this is because, as we saw in previous lectures, random (d, c)-graphs make good expanders, which make good codes, which are characterized by large minimum distance, which allows for large distance of each code B_i, which means that each element in B is far from code(H).

The following definition can be treated, for our purposes, as a black box. The first lemma says that, with high probability, a random parity check matrix has the defined property, and the next lemma says that, if a matrix has the property, then for any deterministic algorithm with query complexity k, the probability of the algorithm rejecting an input chosen according to $\mathcal{D}$ is small.

Definition 12 *H is called (k, ϵ')-local if for all $u \in \{0,1\}^n$ such that u is the sume of at least $\epsilon' m$ columns of H,*

$$|\text{supp}(u)| \geq k.$$

Lemma 13 *For any sufficiently large c, d, there exist constants γ, $\epsilon' > 0$, such that with high probability over (d, c) LDPC codes, H is (k, ϵ')-local, where $k = \gamma \cdot n$.*

Lemma 14 *If H is (k, ϵ')-local, then for all $A \in \mathcal{A}_k$,*

$$\Pr_{f \leftarrow \mathcal{D}}[A \text{ rejects } f] \leq 2\epsilon'.$$

Now, fix a linear test $A(\alpha_1, \alpha_2, \ldots, \alpha_t)$, $t = \gamma n$, and pick a random (d, c)-regular graph with matrix H. Then, by lemma 13, with high probability, H is (k, ϵ')-local for $k = \gamma n$.

By lemma 11, there exists some δ such that with high probability, $f \sim \mathcal{D}$ is δ-far from code(H).

By a union bound argument, with high probability, there exists H with both of these properties. Lemma 14 then tells us that $\Pr_{f \leftarrow \mathcal{D}}[A \text{ accepts } f] \geq 1 - 2\epsilon' = 1 - \epsilon$. An application of Yao's principle then tells us that

$$\max_{f \in X_\delta} \Pr[R \text{ accepts } f] \geq 1 - \epsilon,$$

for all R. Thus, any linear test must accept an input that is δ-far from code(H) with probability $\geq 1 - \epsilon$, as desired.

■

References

[1] Eli Ben-Sasson, Prahladh Harsha, and Sofya Raskhodnikova. Some 3cnf properties are hard to test. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, STOC '03, pages 345–354, New York, NY, USA, 2003. ACM.