U.C. Berkeley Handout N12
CS294: Pseudorandomness and Combinatorial Constructions October 6, 2005
Professor Luca Trevisan Scribe: Madhur Tulsiani

Notes for Lecture 12

1 Set-systems for the Nisan-Wigderson generator

We first continue the construction of the Nisan-Wigderson generator from the previous lecture. We assumed that is is possible to construct a large number of sets with small intersections in time polynomial in the number of sets. We now give an algorithm for constructing these sets. Formally, we prove the following

Lemma 1 *For every $n, \delta > 0$ it is possible to construct in time $2^{O(n)}$ a collection $S_1, \ldots S_N$ $(N = 2^{\delta n})$ such that*

- $\forall i \neq j \quad |S_i \cap S_j| \leq \delta n$
- $S_i \subseteq [t] \quad (t = \lceil \frac{e^2}{\delta} \rceil n)$

We first prove a Chernoff bound on the sum of independent Bernoulli variables which we shall require later.

Lemma 2 (Chernoff bound) *If $X_1, \ldots X_n$ are mutually independent 0/1 random variables and $\mathbf{Pr}[X_i = 1] = p_i$ then*

$$\mathbf{Pr}\left[\sum_{i=1}^{n} X_i > \alpha \sum_{i=1}^{n} p_i\right] = e^{(\sum_{i=1}^{n} p_i)(\alpha - 1 - \alpha \ln \alpha)}$$

Proof:

$$\begin{aligned}
\mathbf{Pr}\left[\sum_i X_i > \alpha \sum_i p_i\right] = \mathbf{Pr}\left[\alpha^{\sum_i X_i} > \alpha^{\alpha \sum_i p_i}\right] &\leq \frac{\mathbb{E}\left[\alpha^{\sum_i X_i}\right]}{\alpha^{\alpha \sum_i p_i}} \quad \text{(by Markov's inequality)} \\
&= \frac{\prod_i \mathbb{E}\left[\alpha^{X_i}\right]}{\alpha^{\alpha \sum_i p_i}} \quad \text{(as X}_i'\text{s are independent)} \\
&= \frac{\prod_i (1 - p_i + \alpha p_i)}{\alpha^{\alpha \sum_i p_i}} \\
&\leq \frac{\prod_i e^{(\alpha p_i - p_i)}}{\alpha^{\alpha \sum_i p_i}} \quad (1 + \mathrm{x} \leq \mathrm{e}^{\mathrm{x}} \text{ for } \mathrm{x} \geq 0) \\
&= \frac{e^{\sum_i \alpha p_i - p_i}}{(e^{\ln \alpha})^{\alpha \sum_i p_i}} = e^{(\sum_{i=1}^{n} p_i)(\alpha - 1 - \alpha \ln \alpha)}
\end{aligned}$$

□

To construct the required set-system, we divide the interval $\left[1, n\frac{e^2}{\delta}\right]$ into the n blocks $\left[1, \frac{e^2}{\delta}\right], \left[\frac{e^2}{\delta}+1, 2\frac{e^2}{\delta}\right], \ldots, \left[(n-1)\frac{e^2}{\delta}+1, n\frac{e^2}{\delta}\right]$. We call a set S structured if it contains exactly one element from each block. We now claim that the following algorithm constructs the required set-system in $2^{O(n)}$ time.

```
Set-System ()
  Take $S_1$ as an arbitrary structured set
  for $i = 2$ to $\delta n$
    Choose a structured set $S_i$ such that $|S_i \cap S_j| \leq \delta n \ \forall 1 \leq j < i$
```

The running time of the algorithm, even if it examines all possible sets at each stage, is $2^{O(n)}$. However, we need to prove that at each stage there exists such a structured set so that the algorithm can proceed. We claim

Lemma 3 *If $S_1, \ldots S_m$ $(m \leq 2^{\delta n})$ are structured sets, then there is a structured set S such that $|S \cap S_i| \leq \delta n \ \ \forall 1 \leq i \leq m$.*

PROOF: We begin by considering the probability of intersection of a random structured set S with a fixed set S_i. Let

$$X_j = \begin{cases} 1 & \text{if S and S}_\text{i} \text{ intersect in the jth block} \\ 0 & \text{otherwise} \end{cases}$$

Then $|S \cap S_i| = \sum_{j=1}^{n} X_j$ with $\mathbf{Pr}[X_j = 1] = \frac{\delta}{e^2}$. Using the Chernoff bound we have

$$\mathbf{Pr}[|S \cap S_i| > \delta n] \ = \ \mathbf{Pr}[\sum_j X_j > \delta n] \ \leq \ e^{\left(\frac{\delta n}{e^2}(e^2 - 1 - 2e^2)\right)} \ \leq \ e^{-\delta n} \ < \ 2^{-\delta n} < 1/m$$

Therefore the probability that there is at least one S_i such that $|S \cap S_i| > \delta n$ is less than 1, which proves the claim. $\square$

This completes the construction of the set-system and also concludes the proof of the Nisan-Wigderson result.

Theorem 4 (Nisan-Wigderson) *Suppose that there exists a language L decidable in time $2^O(n)$ and a constant $\delta > 0$ such that L is $\left(2^{\delta n}, \frac{1}{2^{\delta n}}\right)-$hard on inputs of length n, then there is a generator $G : \{0,1\}^{O(\log N)} \to \{0,1\}^N$ computable in time poly(n) which is $\left(\Omega(N^2), \Omega(1/N)\right)-$pseudorandom.*

2 The Impagliazzo-Wigderson theorem

The above theorem assumes a language that is exponentially hard on average. A later result by Impaglaizzo and Wigderson shows that it is sufficient to find a language which is hard in the worst case i.e. it cannot be solved by subexponential circuits on all inputs. They show that this allows the construction of another language which is hard on average as required above.

Theorem 5 (Impagliazzo-Wigderson) *Suppose there exists a language L decidable in time $2^{O(n)}$ and a constant $\delta > 0$ such that L cannot be solved by circuits of size less than $2^{\delta n}$ then there also exits a language L' decidable in time $2^O(n)$ and a constant $\delta' > 0$ such that L' is $\left(2^{\delta' n}, \frac{1}{2^{\delta' n}}\right)$ − hard on inputs of length n.*

We shall try to prove the theorem by constructing the language L' and giving a reduction from L to L' such that an input of length n for L maps to an input of length $\Theta(n)$ for L'. We then try to construct a family of circuits for L using a family which solves L' on a significant fraction of inputs.

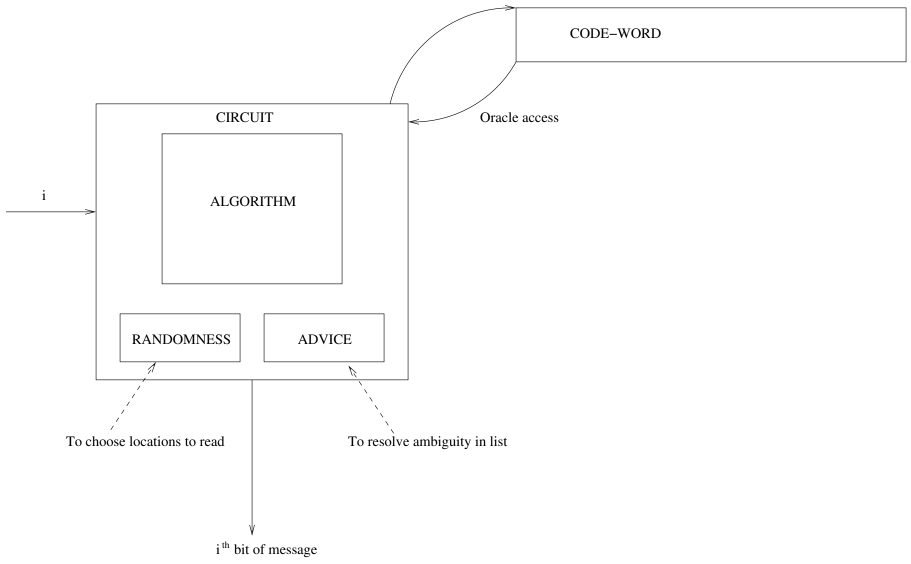

Figure 1: Circuit for deciding L using L'

We view the entire truth table of a function which decides L on inputs of length n as single string of length $N = 2^n$. We then construct the table for the language L' by applying a suitable error-correcting code. The output length of the code is required to be $2^{\Theta(n)} = poly(N)$. For those input lengths of L' that are not generated by this method, we just pick the truth table by an arbitrary convention (say accepting all strings of that length).

Thus the output given by the circuit for L' may be viewed as bits of a corrupted version of the codeword such that at most $1/2 + \Omega(N^\delta)$ fraction is corrupted. The problem of solving L on all inputs is now the problem of finding the original message as illustrated in Fig.1. There are however a few problems to be tackled:

1. The fraction of the codeword that may be corrupted is too large for unique decoding to be possible. Thus, a list-decoding giving all possible values will be required. This still leaves the question of choosing a candidate from the list to then decide the membership of L.

2. The time available to the decoding algorithm is only sublinear.This is so because we want to be able to produce the the circuit for L for any δ that L may be defined with.

However, note that we do not have to retrieve the whole message but just one specific bit of it in as we are interested in solving L only on one input at a time. Thus, the problem is to find a given bit of the original message in sublinear time. We shall design this code over the next few lectures. We shall first look at unique and list decoding of Reed-Solomon codes and then design codes which can be uniquely and also list-decoded in sublinear time.

3 Unique decoding of Reed-Solomon codes

We recall that Reed-Solomon codes are mappings of the form $C : \mathbb{F}^k \to \mathbb{F}^n$ where $\mathbb{F}$ is a finite field such that $|\mathbb{F}| > n$. For computing the code we first fix the field elements $a_1, \dots a_n$. The message is then viewed as a polynomial over the field and we identify the message $(M_0, \dots M_{k-1})$ with the polynomial $p_M(x) \equiv M_0 + M_1 x + \dots M_{k-1}x^{k-1}$. The mapping is then defined as:

$$C(M) \ := \ (p_M(a_1), \dots, p_M(a_n))$$

Thus the codewords corresponding to two different messages are the evaluations of two degree $k-1$ polynomials on n elements and hence, cannot agree at more than $k-1$ of them.

The decoding problem is then to find the polynomial $p()$ given $(y_1, \dots, y_n)$ which are the values of the polynomial at the (known) points $a_1, \dots, a_n$ with upto e errors.

We first define the set $I = \{i : p(a_i) \neq y_i\}$. Note that we do not actually know the elements of the set. We also define the polynomial $E(x) = \prod_{i \in I}(x - a_i)$. Thus, the zeroes of this polynomial are the $a_i's$ for which we have the erroneous values. Observe that $\forall 1 \leq i \leq n, \ E(a_i)y_i = E(a_i)p(a_i)$. We now give the following decoding algorithm:

```
Decode-Reed-Solomon $(y_1, \dots, y_n)$
  **if** there is a polynomial $p(x)$ such that $p(a_i) = y_i \ \ \forall 1 \leq i \leq n$
    output $p(x)$ and stop.
  **else**
    Find two polynomials $N(x)$ and $E(x)$ such that $N(a_i) = E(a_i)y_i \ \ \forall 1 \leq i \leq n$.
    Output $N(x)/E(x)$.
```

Notice that the running time of the above algorithm is essentially polynomial as the constraints can just be viewed as a system of linear equations in the coefficients of the polynomial which can be solved in polynomial time. We also know that this system does have at least one non-zero solution which is given by the polynomials $E(x)$ and $N(x) = E(x)p(x)$ as described above. We now prove the correctness of the algorithm.

Lemma 6 *Let $E'(x), N'(x)$ be any feasible solution to the above system of equations. Then*

$$\frac{N'(x)}{E'(x)} = p(x)$$

Proof: Let $E'(x), N'(x)$ be any solution to the system of linear equations and let $E(x), N(x)$ be as above. Then for all $1 \leq i \leq n$, we have

$$E'(a_i)N(a_i) = E'(a_i)E(a_i)y_i = (E'(a_i)y_i)E(a_i) = N'(a_i)E(a_i)$$

Thus, the polynomials $E'N$ and EN' agree on at least n points. However, $deg(E'), deg(E) \leq e$ and $deg(N'), deg(N) \leq e - k + 1$ and so $deg(E'N), deg(EN') \leq 2e - k + 1 < n$. Hence, the two polynomials must be identical. This gives

$$\begin{aligned} E'N(x) &= EN'(x) \qquad \text{(in the ring } \mathbb{F}\text{[x])} \\ \Rightarrow \frac{N(x)}{E(x)} &= \frac{N'(x)}{E'(x)} \qquad \text{(as rational functions in the field of quotients of } \mathbb{F}\text{[x])} \\ \Rightarrow \frac{N'(x)}{E'(x)} &= p(x) \end{aligned}$$

This proves the claim. $\square$