6.895 Randomness and Computation	April 23, 2008

Lecture 20

Lecturer: Ronitt Rubinfeld (lecture given by Krzysztof Onak) *Scribe: Ankur Moitra*

1 Introduction

In this lecture we will complete the construction of Nisan's Pseudorandom Generator for space bounded computation. This pseudorandom generator can be used to convert $O(S(n)\log R(n))$ random bits into $R(n)$ bits that appear random to any algorithm that runs in space $S(n)$. We will also introduce two applications of this pseudorandom generator.

2 Matrix Norm Identities

Before constructing Nisan's Pseudorandom Generator for space bounded computation, we will need to define appropriate vector and matrix norms:

Definition 1 $\|x\| = \Sigma_i |x_i|$ *for* $x \in R^s$.

Definition 2 $\|M\| = \sup_{x\in R^s\setminus\{\mathbf{0}\}} \frac{\|xM\|}{\|x\|}$ *for* $M \in R^{s\times s}$

Using these definitions and the triangle inequality:

$\|M+N\| \le \|M\| + \|N\|$ for all $M, N \in R^{s\times s}$

$\|MN\| \le \|M\|\cdot\|N\|$ for all $M, N \in R^{s\times s}$

$\|M\| = \max_i \Sigma_j |M_{ij}|$ for all $M \in R^{s\times s}$

If $|M_{ij}| \le \epsilon$ for all i, j then $\|M\| \le s\epsilon$ for $M \in R^{s\times s}$

If M is a transition probability matrix, then $\|M\| = 1$

3 Construction of Nisan's Pseudorandom Generator

The Nisan Pseudorandom Generator is constructed by choosing a set $h_1, \ldots, h_l \in H$ of pairwise independent functions, and generating a seed y uniformly at random from the space $\{0,1\}^r$ and applying the following iterative procedure:

$G_0(y) = y$,

$G_{i+1}(y, h_1, \ldots, h_{i+1}) = G_i(y, h_1, \ldots, h_i) \circ G_i(h_{i+1}(y), h_1, \ldots, h_i)$,

where $\circ$ is the string concatenation operator.
Then $G_l(y, h_1, \ldots, h_l)$ generates a distribution on the space $\{0,1\}^{r2^l}$.
For example,

$$G_2(y, h_1, h_2) = y \circ h_1(y) \circ h_2(y) \circ h_1(h_2(y)).$$

Definition 3 U_n *is the uniform distribution on* $\{0,1\}^n$.

Definition 4 *If* $A, B \subseteq \{0,1\}^n$, *then* $h : \{0,1\}^n \to \{0,1\}^n$ *is* (ϵ, A, B)-independent *if*

$$\left|\Pr_{x\in\{0,1\}^n}[x \in A, h(x) \in B] - \Pr_{x,y\in\{0,1\}^n}[x \in A, y \in B]\right| \le \epsilon.$$

Last lecture we proved the Hash Mixing Lemma:

Lemma 5 *For $\epsilon = 2^{-r/3}$, for all $A, B \subset \{0,1\}^r$ and all but an ϵ fraction of $h \in H$, h is (ϵ, A, B)-independent.*

4 Transition Matrices generated by Space-Bounded Algorithms

Consider an algorithm $A(x, y)$ where x is the input string and y is the random bit string used by the algorithm. Suppose also that A runs in space $S(n)$ and that $S(n) = \Omega(\log n)$. Suppose also that A uses $R(|x|)$ random bits.

Then fix $x \in \{0,1\}^n$. Define T to be the number of possible configurations (intermediate computation steps) of A on x. $T = O(nS(n)) \cdot 2^{O(S(n))} = 2^{O(S(n))}$. The first factor follows because the algorithm is run on a two tape Turing Machine, and the first tape is the read-only input. Then there are n possible locations for the first head and $S(n)$ possible locations for the second head. The number of states is finite, and thus there are $O(nS(n))$ possible configurations for the state and location of both tape heads. Additionally, there are $O(1)$ possible choices for the symbol in each tape cell, and the constant hidden by the $O(\cdot)$ notation depends on the size of the tape alphabet.

Also, for any distribution D over $\{0,1\}^k$ define $Q(D)$ as the transition matrix corresponding to algorithm A run on input string x. Formally, for each y that can be generated by the distribution D, run A starting from state i on x using the random bits sequentially (to decide which outgoing transition to choose) until all random bits in y are used up. This defines a deterministic computation that results in a particular state. Sum the probabilities of all strings y that can be generated by D that result in transition from state i to state j.

Definition 6 *A sequence $h_1, \ldots, h_k \in H^k$ is* ϵ-good *if $\|Q(G_k(U_r, h_1, \ldots, h_k)) - Q(U_{r2^k})\| \le \epsilon$.*

Lemma 7 $\Pr[h_1, \ldots, h_k$ *is not* $(2^k - 1)T^2\epsilon$*-good*$] \le kT^3\epsilon$, *where $\epsilon = 2^{-r/3}$.*

Proof This lemma will be proven by induction. For the case $k = 0$ this is trivial because the distribution $G_0(U_r)$ is identical to the distribution U_r, and hence the transition matrices are identical.

For $k > 0$ we will bound the probability that $h_1, \ldots, h_k$ is not $(2^k - 1)T^2\epsilon$-good by the probability that at least one of two particular events does not occur. Then in the case that both events occur we will show that these events, together with the matrix and vector norm inequalities that we stated earlier, imply that the condition is met and that $h_1, \ldots, h_k$ is $(2^k - 1)T^2\epsilon$-good.

Definition 8 $B_{ij}^{h_1,\ldots,h_{k-1}} = \{x \in \{0,1\}^r | G_{k-1}(x, h_1, \ldots, h_{k-1})$ *takes state i to state j*$\}$

The events are:

- **Event I:** $(h_1, \ldots, h_{k-1})$ is $(2^{k-1} - 1)T^2\epsilon$-good.
- **Event II:** For all triples of states (i, l, j), h_k is $(\epsilon, B_{il}^{h_1,\ldots,h_{k-1}}, B_{lj}^{h_1,\ldots,h_{k-1}})$ independent.

Then $\Pr[\neg\mathbf{I} \vee \neg\mathbf{II}] \le \Pr[\neg\mathbf{I}] + \Pr[\neg\mathbf{II}]$. Note that this probability is over both the random choices of the strings $x \in \{0,1\}^r$ and over the random choice of the function h_k. Using the inductive hypothesis, $\Pr[\neg\mathbf{I}] \le (k-1)T^3\epsilon$. Next, using the union bound over all possible triples of states, and the hash mixing lemma, $\Pr[\neg\mathbf{II}] \le T^3\epsilon$ because the probability is also over random choices for the function h_k. Therefore, $\Pr[\neg\mathbf{I} \vee \neg\mathbf{II}] \le \Pr[\neg\mathbf{I}] + \Pr[\neg\mathbf{II}] \le (k-1)T^3\epsilon + T^3\epsilon = kT^3\epsilon$.

It remains only to show that if both Events I and II occur, then $h_1, \ldots, h_k$ is $(2^k - 1)T^2\epsilon$-good. So suppose that both Events I and II occur. We have

$$\|Q(G_k(U_r, h_1, \ldots, h_k)) - Q(U_{r2^k})\| \le \|Q(G_k(U_r, h_1, \ldots, h_k)) - Q(G_{k-1}(U_r, h_1, \ldots, h_{k-1}))^2\|$$

$$+\|Q(G_{k-1}(U_r,h_1,\ldots,h_{k-1}))^2-Q(U_{r2^k})\|.$$

Then consider a particular element in the matrix $Q(G_k(U_r,h_1,\ldots,h_k))-Q(G_{k-1}(U_r,h_1,\ldots,h_{k-1}))^2$. The probability of transitioning from state i to state j is the sum over l of transitioning from i to l on the first half $r2^{k-1}$ random bits times the probability of transitioning from l to j on the second half $r2^{k-1}$ random bits.

By the definition of (ϵ,A,B)-independent, we know that for each l, the probability of transitioning from i to l and l to j under the distribution $G_k(U_r,h_1,\ldots,h_k)$ is at most an additive ϵ different than the probability of transitioning from i to l and l to j under the distribution

$$G_{k-1}(U_r,h_1,\ldots,h_{k-1})\circ G_{k-1}(U_r,h_1,\ldots,h_{k-1}).$$

The (i,j)-th entry of the matrix $Q(G_k(U_r,h_1,\ldots,h_k))-Q(G_{k-1}(U_r,h_1,\ldots,h_{k-1}))^2$ is a sum of

$$\Pr_{x\in\{0,1\}^r}[x\in B_{i,l}^{h_1,\ldots,h_{k-1}},h(x)\in B_{l,j}^{h_1,\ldots,h_{k-1}}]-\Pr_{x,y\in\{0,1\}^r}[x\in B_{i,l}^{h_1,\ldots,h_{k-1}},y\in B_{l,j}^{h_1,\ldots,h_{k-1}}]\in[-\epsilon,\epsilon]$$

over all l. Hence each entry of the matrix is in the range $[-T\epsilon,T\epsilon]$, and

$$\|Q(G_k(U_r,h_1,\ldots,h_k))-Q(G_{k-1}(U_r,h_1,\ldots,h_{k-1}))^2\|\le T\cdot T\epsilon=T^2\epsilon.$$

To bound the second term, define $M_1=Q(G_{k-1}(U_r,h_1,\ldots,h_{k-1}))$ and $M_2=Q(U_{r2^{k-1}})$. Then

$$\begin{aligned}\|Q(G_{k-1}(U_r,h_1,\ldots,h_{k-1}))^2-Q(U_{r2^k})\| &= \|Q(G_{k-1}(U_r,h_1,\ldots,h_{k-1}))^2-Q(U_{r2^{k-1}})^2\|\\ &= \|M_1^2-M_2^2\|\le\|M_1+M_2\|\cdot\|M_1-M_2\|\\ &\le (\|M_1\|+\|M_2\|)\cdot\|M_1-M_2\|\le 2\|M_1-M_2\|\end{aligned}$$

This last line follows because the matrices M_1,M_2 are transition matrices. Also because Event I occurred, we have

$$\|M_1-M_2\|\le(2^{k-1}-1)T^2\epsilon.$$

Eventually,

$$\|Q(G_k(U_r,h_1,\ldots,h_k))-Q(U_{r2^k})\|\le T^2\epsilon+2\cdot(2^{k-1}-1)T^2\epsilon=(2^k-1)T^2\epsilon.$$

And this completes the inductive proof. ■

5 Nisan's Pseudorandom Generator

Using the technical lemma proven in the previous section, we can prove Nisan's theorem on pseudorandom generators for space-bounded computation.

Theorem 9 *For any algorithm A that runs in $S(n)=\Omega(\log n)$ space and uses $R(n)$ random bits, there is a pseudorandom generator for A with parameter $\frac{1}{10}$ that uses $O(S(n)\log R(n))$ random bits and runs in $O(S(n)\log R(n))$ space.*

Proof Note that $l\le 2^l\le R(n)$ because we only require $R(n)$ random bits, and the distribution $G_l(U_r,h_1,\ldots,h_l)$ will be a distribution on $r2^l$ bits. Also $R(n)\le T$, if the algorithm is not permitted to loop. Recall that $\epsilon=2^{-r/3}$, and also recall that $T=2^{O(S(n))}$. Then we can choose $r=O(S(n))$ and achieve $\Pr[(h_1,\ldots,h_l)$ is not $\frac{1}{20}$ good$]\le\frac{1}{20}$.

Suppose that $(h_1,\ldots,h_l)$ is $\frac{1}{20}$ good. Then

$$
\begin{aligned}
&|\Pr[A \text{ accepts } y \leftarrow U_{R(n)}] - \Pr[A \text{ accepts } y \leftarrow G_l(U_r, h_1, \ldots, h_l)]| \leq \\
&\leq |1_{\text{start}} \cdot (Q(U_{R(n)}) - Q(G_l(U_r, h_1, \ldots, h_l))) \cdot 1_{\text{accept}}|
\end{aligned}
$$

where 1_{start} is the (horizontal) characteristic vector for the start configuration and 1_{accept} is the (vertical) characteristic vector for the set of accepting states. Note that $w = 1_{\text{start}} \cdot (Q(U_{R(n)}) - Q(G_l(U_r, h_1, \ldots, h_l)))$ is a row of the matrix $Q(U_{R(n)}) - Q(G_l(U_r, h_1, \ldots, h_l))$, and $\|w\| \leq \|Q(U_{R(n)}) - Q(G_l(U_r, h_1, \ldots, h_l))\| \leq \frac{1}{20}$. Furthermore, $|w \cdot 1_{\text{accept}}|$ is the sum of absolute values of some coordinates of w, so $|w \cdot 1_{\text{accept}}| \leq \|w\| \leq \frac{1}{20}$.

Summarizing, the probability that the algorithm accepts the input differs by at most $\frac{1}{20} + \frac{1}{20} = \frac{1}{10}$.

Consider now the number of random bits needed to run the pseudorandom generator. The seed requires $r = O(S(n))$ random bits, and generating the pairwise independent functions requires $O(lr) = O(S(n) \log R(n))$ random bits. Also, recall each of these functions can be calculated in $O(r)$ space which implies that the pseudorandom generator runs in $O(lr) = O(S(n) \log R(n))$ space, which is required to store the random bits. ■

6 Applications

Universal Traversal Sequences. Nisan gave an application of this pseudorandom generator to universal traversal sequences. Consider a sequence and a start node. This pair defines a walk if we consider all outgoing edges from a particular node as consistently numbered. Then beginning at the start node, choose the outgoing edge specified by the sequence. If no such edge exists (because the degree of the current node is too small), then the walk stays put. A universal traversal sequence is a sequence such that for all connected n-node graphs and for any start node s in the graph, the walk specified by the sequence and the start node will visit all nodes in the graph. The probabilistic method can be used to show the existence of universal traversal sequences of polynomial length. However derandomizing the construction of universal traversal sequences has remained an open problem. Nisan used this pseudorandom generator to deterministically construct universal traversal sequences of size $n^{O(\log n)}$ in $n^{O(\log n)}$ time.

Streaming. This pseudorandom generator has also been used extensively in streaming, where small space algorithms are often constructed by assuming access to a truly random function (a random hash function for instance), and using Nisan's pseudorandom generator to remove this assumption, incurring a small space penalty.