CS 59000 CTT Current Topics in Theoretical CS Dec 6, 2012

Lecture 32

Lecturer: Elena Grigorescu *Scribe: Purnima, Bhaskar and Nader*

1 Introduction

Throughout this course we have seen that having access to random bits could be very helpful in designing very efficient algorithms. In general, many problems for which we do not know if they have an efficient deterministic algorithm have efficient randomized algorithms. Randomness and 'pseudorandomness' are important in applications from cryptography and complexity theory (eg. pseudorandom generators, randomness extractors, expander graphs and error-correcting codes) (see [1] for a comprehensive survey).

However, pure randomness is an expensive resource as it is hard to find even in nature. In this lecture we will talk about two questions: 1) Is randomness really necessary in computation? (or, can we decrease the amount of randomness necessary?) and 2) How can we obtain true randomness?

Removing randomness from algorithms (the process of derandomization) is hard in general and brings up fundamental computational questions. Some famous problems in complexity (eg. primality testing, or undirected connectivity) took a few decades to be fully derandomized. In this lecture we'll see a derandomization technique that uses again the pseudorandom properties of expander graphs. Also, to address the 2nd question above we will introduce objects called extractors, that can extract true random bits from weak sources of randomness.

2 Pseudorandomness through expander graphs

We will show how to amplify the success probability of a randomized algorithm without increasing the number of random bits used.

Let L be a language for which there exist a randomized algorithm A as follow:

1. if $x \in L$ then $\Pr_{r \leftarrow \{0,1\}^n}[A(x,r) = 1] = 1$
2. if $x \notin L$ then $\Pr_{r \leftarrow \{0,1\}^n}[A(x,r) = 1] \leq \frac{1}{4}$

(where $r \leftarrow \{0,1\}^n$ means that r is uniform from $\{0,1\}^n$.) Note that this algorithm uses n random bits and has error of at most $\frac{1}{4}$. We now want to obtain an algorithm with error $\leq \delta$ (for some arbitrarily small δ) i.e if $x \notin L$ then $\Pr[A(x,r) = 1] \leq \delta$.

An immediate solution is to use the following algorithm:

Algorithm 1

1. run Algorithm A for $O(\log \frac{1}{\delta})$ times.
2. output the AND of all the bits output.

Obviously, Algorithm 2 requires $O(\log \frac{1}{\delta}) \cdot n$ many random bits and has error of at most δ.

It turns out that we can achieve δ error *without* any additional random bit (ie. with only n random bits). Algorithm 2 below achieving these parameters is due to Karp, Pippinger and Sipser.

Let G be a $N = 2^n \times 2^n$ $L \cup R$ d-left regular $(\frac{N}{2}, \gamma d)$-expander (so every set $S \subseteq L$ of size $|S| \leq N/2$ is s.t. $|\Gamma(S)| \geq \gamma d|S|$). We will choose the values of d, γ as functions of δ at the end of the analysis. Also, we can assume that for any $r \in \{0,1\}^n$, $r \in \Gamma(r)$. The idea is to view both the left and right hand side vertices of G as the elements of $\{0,1\}^n$. While in Algorithm 2 the success probability was amplified by running A a few times on independent random strings, in the new algorithm the success will be amplified by running A on strings that are correlated, yet preserve some sort of pseudorandom properties as encountered in the neighborhoods of vertices belonging to expander graphs.

Algorithm 2

Algorithm A'':

1. Pick $r \leftarrow \{0,1\}^n$ uniformly at random
2. Let $\Gamma(r) = \{r_1, r_2, \cdots, r_d\}$ the set of neighbors of r
3. Run $A(x, r_i)$ for all $i \in [d]$
4. Output the AND of all runs.

It is clear that A'' only uses n bits of randomness. One can build such large expanders so that given r its neighborhood can be computed explicitly in $poly(n)$ time without having to write down the whole graph G which would take an exponential amount of time.

Claim 1 *A'' has at most δ fraction of errors, i.e. if $x \in L$, $\Pr_r[A''(x,r) = 1] \leq \delta$.*

Proof Assume $x \notin L$. Let $Bad_x = \{r \in \{0,1\}^n, A(x,r) = 1\}$, so $|Bad_x| \leq \frac{N}{4}$. Let $B = \{r \mid \Gamma(r) \subset Bad_x\}$. We want to show that $|B| \leq \delta N$

Claim 2 $|B| \leq \delta N$

Proof Assume $|B| > \delta N$. Since $r \in \Gamma(r)$ $|B| \leq |\Gamma(B)|$. Note that $\Gamma(B) \subset Bad_x$ so $|B| \leq |\Gamma(B)| \leq |Bad_x| \leq \frac{N}{4} \leq \frac{N}{2}$. Since G is an expander it follows that $\Rightarrow \frac{N}{4} \geq |\Gamma(B)| \geq \gamma d|B| > \gamma d\delta N$. If we chose $\gamma = \frac{5}{8}, d = \frac{2}{5\delta}$ then we would have shown $\frac{N}{4} > \gamma d\delta N = \frac{N}{4}$ (a contradiction!) ■■

3 Extracting randomness from biased sources

In this section we discuss the problem of extracting randomness from a biased source of randomness.

Suppose that we want to simulate a uniformly random bit using a biased coin. Let the coin be biased so that $\Pr[H] = 1/2+\delta$ and $\Pr[T] = 1/2-\delta$.

Von Neumann proposed a method to get a uniformly random bit using this biased coin as follows: flip the coin twice, if the result is HT then output 1, and if the result is TH output 0. Else do not output anything. Then in this case, $\Pr[1] = \Pr[0] = \Pr[H] \cdot \Pr[T] = (1/2 + \delta) \cdot (1/2 - \delta)$. The expected time to get a uniform random bit is then $1/(2(1/4 - \delta^2))$.

Also another method is to flip the biased coin n times and output 1 if the number of heads is even and 0 otherwise. In the special case when $n = 2$, note that $\Pr[1] = \Pr[HH] + \Pr[TT] =$

$(1/2+\delta)^2 + (1/2-\delta)^2 = \frac{1}{2} + 2\delta^2$. Inductively, after n coin flips, one can show that this probability is $\Pr[1] = \frac{1}{2} + \frac{1}{2}(-2\delta)^n$, which for large enough n is negligibly close to $1/2$. We can model this experiment in the form of a randomness extractor.

Definition 3 *(Deterministic extractor) An extractor* $Ext : \{0,1\}^n \to \{0,1\}^m$ *is a function which when applied to a random variable over* $\{0,1\}^n$ *outputs strings in* $\{0,1\}^m$ *whose probability distribution is close to the uniform distribution over* $\{0,1\}^m$ *(denoted* U_m*).*

Next we define a notion of distance between distributions.

Definition 4 *The statistical distance between two random variables taking values in the domain* D *(or two probability distributions)* X *and* Y *is as follows*

$$\Delta(X,Y) = (1/2)\sum_{a\in D} |\Pr[X=a] - \Pr[Y=a]|.$$

We say that an extractor is an ϵ-extractor if its statistical distance from the uniform distribution (i.e. $\Delta(X,Y)$) is at most ϵ.

Property of extractors Suppose that we have an algorithm $A(w,r)$ with the property that it has error δ when computing a function $f(w)$ (i.e. $\Pr_r[A(w,r) \neq f(w)] \leq \delta$ for all $w \in \{0,1\}^n$), when r is drawn from the uniform distribution on $\{0,1\}^m$. Let $Ext : \{0,1\}^n \to \{0,1\}^m$ be an ϵ- extractor for some class C of sources (distributions) on $\{0,1\}^n$. Then if we try to compute f by running A with the randomness provided by the distribution $Ext(X)$ where $X \in C$ it will have error $\delta + \epsilon$. We won't show this property, but only remark that it intuitively follows since the distribution of the strings r that are bad for A on w has at most an ϵ mass more than the uniform distribution.

Extractors based on min entropy We next move to discussing extractors based on the min entropy of a source of weak randomness. We can start by asking: How much randomness can we extract from a random binary variable?

Recall that the entropy of a random variable is a measure of the amount of randomness in the distribution of the random variable, measured in bits. It is intuitively clear that in to order to extract k bits of entropy the support of the weak (say, binary) source must be of size at least 2^k.

There are a few notions of entropy that are useful in different contexts. The one that we will use here is that of minimum entropy. Minimum Entropy is defined as $H_\infty(X) = \min_{a\in d} \log \frac{1}{\Pr[X=a]}$.

Definition 5 *A random variable is a* k*-source if* $H_\infty \geq k$*, i.e.* $\Pr[X=a] \leq 2^{-k}$.

Some very important examples of k sources are *flat* k-sources, where a random variable is uniformly distributed over a set $S \subseteq \{0,1\}^n$ and $|S| = 2^k$. By standard results in convex geometry it can be shown that *every* k-source is a convex combination of flat k-sources (i.e. $Y = \sum p_i X_i$ where $\sum p_i = 1$ and Y is a k-source and X_i's are flat k-sources. So, the study of k-sources reduces to the study of flat k-sources.

It can be shown that for any flat k-source there is an extractor that can extract almost all of its entropy.

Proposition 6 *$\forall n, k, m, \epsilon > 0$ and every flat k-source X, if a random function $Ext : \{0,1\}^n \to \{0,1\}^m$ with $m = k - \log(1/\epsilon) - O(1)$ is s.t. $Ext(X)$ is ϵ-close to U_m w.p. $1 - 2^{-\Omega(2^k \epsilon^2)}$.*

One would like an extractor to work for many kinds of sources. It turns out that a union bound applied to the above result does not actually imply the existence of one function that works for all k-flat sources. This is where a very important insight has been proposed: if we allow access to a small number of truly random bits together with the weak source of randomness then one can actually obtain an extractor that extracts almost all the entropy of many weak sources.

Definition 7 (Seeded Extractors) *A seeded extractor is $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that $\forall$ k-sources X, $Ext\ (X, U_d)$ is $\epsilon-$close to U_m.*

Using a union bound in Proposition 6 we get that there exists a good extractor for all k-flat sources.

Proposition 8 *$\forall n, k, \epsilon > 0$, $\exists$ a (k, ϵ)-extractor $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m = k + d - \Omega(\log(1/\epsilon))$ with $d = \log(n - k) + O(\log(1/\epsilon))$.*

References

[1] S. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 2012.