CS 744: Pseudorandomness Generators

Lecture 8: Basic Derandomization Techniques - III

January 21, 2015

1 Pairwise Independence

One way to derandomize is to assume that, instead of requiring that some number of strings are drawn at random such that they are mutually independent, we can work with *pairwise* independent random variables.

A standard example of pairwise independent random variables which are not mutually independent, is as follows. Consider two independent bits a and b and let the third bit be $c = a \oplus b$. Then a, b and c are pairwise independent. However, given any two variables, the third is completely determined. Hence they are not mutually independent.

We generated three pairwise independent bits from two purely random bits. We can see if we can generalize this method of generating some pairwise random bits from fewer purely random bits.

Lemma 1. *Let $a_1, \ldots, a_k$ be mutually independent random bits. For any $S \subseteq \{1, \ldots, k\}$, define the random variable A_S by*

$$A_S = \oplus_{i \in S} a_i.$$

Then the set of $2^k - 1$ random variables thus generated are pairwise independent and unbiased.

Proof. First we show that the random variables are unbiased. Let $S \subseteq \{1, \ldots, k\}$ be arbitrary. Then the probability that $A_S = 1$ is the probability that $\oplus_{i \in S} a_i = 1$. Since the random variables a_i are unbiased, they are each 1 with probability $1/2$. Hence the probability that their parity is 1 is $1/2$ as well.

Now, we show that A_S is independent of A_T when S and T are distinct subsets of $\{1, \ldots, k\}$. To see this, we have to show that the probability that $A_S = a$ and $A_T = b$ is $1/4$, for arbitrary bits a and b.

First, if S and T are disjoint, then $\oplus_{i \in S} a_i = \beta_1$ and $\oplus_{i \in T} a_i = \beta_2$ for bits β_1 and β_2 is equal to $1/4$, by the mutual independence of $a_1, \ldots, a_k$.

Now consider general S and T which are disjoint. The value A_S can be written as $A_{S \cap T} \oplus A_{S-T}$. Similarly, A_T is $A_{S \cap T} \oplus A_{T-S}$. It is clear that $A_{S \cap T}$ influences A_S and A_T in an equal way. Now, $A_S = \beta_1$ and $A_T = \beta_2$ if and only if $A_{S-T} = (\beta_1 \oplus A_{S \cap T})$ and $A_{T-S} = (\beta_2 \oplus A_{S \cap T})$. This probability is $1/4$ since $(S - T) \cap (T - S) = \emptyset$. □

This shows that is possible to generate n pairwise random variables from $\Theta(\log n)$ mutually random bits. Keeping the "enumeration" method of derandomization in mind, this says that if we can make an algorithm work with n *pairwise independent* bits, then we can derandomize the algorithm fully in polynomial time.

This is a substantial improvement over the previously discussed methods, since this is both feasible, and applicable to many problems like MAXCUT.

1.1 Pairwise independence hash functions

If we want pairwise independent values from some set of values other than just 0 or 1, we need stronger techniques than the one above if we still want to use only $\Theta(\log n)$ mutually independent ("pure random") bits.

Definition 2. *A multiset of functions $\mathcal{H} = \{h : [N] \to [M]\}$ is* pairwise independent *if the following conditions hold for a function h drawn uniformly at random from $\mathcal{H}$.*

1. *For every $x \in [N]$, the variable $h(x)$ is uniformly distributed in $[M]$.*
2. *For distinct x_1 and x_2 in the domain, the variables $h(x_1)$ and $h(x_2)$ are independent of each other.*

The above conditions are equivalent to saying that for distinct x_1 and x_2 in the domain and for every y_1 and y_2 in the range, the probability that $h(x_1) = y_1$ and $h(x_2) = y_2$ is $\frac{1}{M^2}$.

Note: The experiment in the definition is as follows. Fix an element x, and then draw a function h at random from $\mathcal{H}$. This is followed by examining $h(x)$. Thus the random sampling is not from the domain of the functions in the hash family. The random sampling is to pick a function from $\mathcal{H}$. Since $\mathcal{H}$ is a finite set, it is possible to sample uniformly at random from it.

We say that $\mathcal{H}$ is *explicit* if given the description of h and x, we can compute $h(x)$ in time $\text{poly}(\log N, \log M)$.

In the next lecture, we will construct an explicit pairwise independent hash family.