CS 6815: Pseudorandomness and Combinatorial Constructions **Fall 2022**

Lecture 3: Aug 30, 2022

Lecturer: Eshan Chattopadhyay *Scribe: Noam Ringach*

1 Introduction

We begin by recalling our definition of BPP:

Definition 1.1 (BPP)**.** *A language L is in* BPP *if there exists a polynomial time probabilistic Turing Machine (PTM) $\mathcal{A}$ such that for all possible inputs x we have that*

$$\Pr_{r\sim\{0,1\}^{\mathsf{poly}(|x|)}}[\mathcal{A}(x,r)=L(x)]\geq\frac{2}{3}.$$

In the last lecture, we saw that to increase the accuracy of our algorithm from at least $\frac{2}{3}$ to at least $1-\varepsilon$ (i.e., decreasing our errors from at most $\frac{1}{3}$ to at most ε), we can come up with a new PTM $\mathcal{B}$ defined as

Algorithm 1 Definition of $\mathcal{B}$ based on $\mathcal{A}$.

Require: Original PTM $\mathcal{A}$, error rate $\varepsilon>0$
  $t\leftarrow\log\frac{1}{\varepsilon}$
  for $i\in[t]$ **do**
    Sample $r^i\sim\{0,1\}^{\mathsf{poly}(|x|)}$
    $Y_i\leftarrow\mathcal{A}(x,r^i)$
  end for
  return $\mathrm{Maj}(Y_1,\ldots,Y_t)$

Furthermore, $\mathcal{B}$ has the following guarantees on accuracy, runtime, and randomness, where $T(n)=\mathsf{poly}(n)$ is the runtime of $\mathcal{A}$, $R(n)=\mathsf{poly}(|x|)$ is the randomness usage of $\mathcal{A}$, and c is an overhead constant from $\mathcal{B}$.

Algorithm	Accuracy	Runtime	Randomness
$\mathcal{B}$	$1-\varepsilon$	$c\cdot T(n)\cdot\log\frac{1}{\varepsilon}$	$c\cdot R(n)\log\frac{1}{\varepsilon}$

Table 1: Accuracy, runtime, and randomness usage of $\mathcal{B}$.

In this lecture, we explore the following question.

Question 1. *Can we construct another algorithm that uses less randomness than $\mathcal{B}$ to achieve the same $1-\varepsilon$ error rate?*

2 Pairwise Independence

In the process of constructing such an algorithm, we will need to introduce a variation on the independence of random variables (RVs) called *pairwise independence*.

Definition 2.1 (Pairwise Independence)**.** *A collection of RVs $X_1, X_2, \ldots, X_N$ on the domain $\mathcal{X}$ is said to be pairwise independent if the following two conditions are satisfied:*

1. *For all $i, j \in [N]$ such that $i \neq j$, we have that X_i and X_j are independent.*
2. *Each X_i is uniform on $\mathcal{X}$.*

We can easily construct a straightforward example of three pairwise independent RVs using the XOR function.

Example 2.2. *Say that we uniformly sample $x_1, x_2 \sim \{0,1\} = \mathbb{F}_2$ and define $x_3 = x_1 \oplus x_2$. Then $\{x_1, x_2, x_3\}$ is a collection of pairwise independent RVs.*

Proof. To see that x_1, x_2, x_3 are pairwise independent, all we need to show is that $\{x_1, x_3\}$ and $\{x_2, x_3\}$ are independent, since $\{x_1, x_2\}$ are independent by assumption. We will show that $\{x_2, x_3\}$ are independent, and the logic for $\{x_1, x_3\}$ being independent will be identical.

Take any $a, b \in \mathbb{F}_2$. Then

$$\Pr_{x_2, x_3}[x_3 = a \wedge x_2 = b] = \Pr_{x_3}[x_3 = a \mid x_2 = b] \Pr_{x_2}[x_2 = b].$$

Of course, $\Pr_{x_2}[x_2 = b] = \frac{1}{2}$ by definition. Additionally, we see that $\Pr_{x_3}[x_3 = a \mid x_2 = b] = \frac{1}{2}$ because for any fixed $x_2 = b$, there is exactly one value of x_1 so that $x_1 \oplus x_2 = a$ and one value so that $x_1 \oplus x_2 \neq a$. Putting this together, we get that

$$\begin{aligned}\Pr_{x_2, x_3}[x_3 = a \wedge x_2 = b] &= \Pr_{x_3}[x_3 = a \mid x_2 = b] \Pr_{x_2}[x_2 = b] \\ &= \frac{1}{2} \cdot \frac{1}{2} \\ &= \Pr_{x_3}[x_3 = a] \Pr_{x_2}[x_2 = b],\end{aligned}$$

meaning that x_3 and x_2 are indeed independent. □

We can generalize this example to create even more pairwise independent RVs.

Example 2.3. *For any uniformly sampled $x_1, \ldots, x_r \sim \mathbb{F}_2$, take any non-empty subset $S \subseteq [r]$ and define*

$$Z_S = \bigoplus_{i \in S} x_i.$$

Then the collection $\mathcal{Z} = \{Z_S\}_{S \subseteq [r]}$ is a collection of $2^r - 1$ pairwise independent RVs.

Proof. First, we easily see that $|\mathcal{Z}| = 2^r - 1$ because there are 2^r possible subsets of $[r]$ and we are excluding the empty subset, so we have $2^r - 1$ non-empty subsets over which we are indexing (i.e., there are only $2^r - 1$ non-trivial ways we can take XOR over $x_1, \ldots, x_r$).

Second, we will show that $\mathcal{Z}$ is a collection of pairwise independent RVs. Consider any $S, U \subseteq [r]$ such that $S \neq U$ and any $a, b \in \mathbb{F}_2$. Then we have

$$\Pr_{x_1, \ldots, x_r \sim \mathbb{F}_2}[Z_S = a \wedge Z_U = b] = \Pr_{x_1, \ldots, x_r \sim \mathbb{F}_2}[Z_S = a \mid Z_U = b] \Pr_{x_1, \ldots, x_r \sim \mathbb{F}_2}[Z_u = b],$$

and all we need is for $\Pr_{x_1, \ldots, x_r \sim \mathbb{F}_2}[Z_S = a \mid Z_U = b] = \Pr_{x_1, \ldots, x_r \sim \mathbb{F}_2}[Z_S = a]$. Similarly as in Example 2.2, since $S \neq U$ there are some x_i's (namely, $\{x_i\}_{i \in S \setminus U}$) that change the value of Z_S

regardless of the value of Z_U. Hence, because the $\{x_i\}_{i\in S\setminus U}$ can change the value of Z_S to 0 or 1 equally likely, we see that the value of Z_U does not influence the result of Z_S, giving us that $\Pr_{x_1,\dots,x_r\sim\mathbb{F}_2}[Z_S = a \mid Z_U = b] = \Pr_{x_1,\dots,x_r\sim\mathbb{F}_2}[Z_S = a]$, so

$$\begin{aligned}\Pr_{x_1,\dots,x_r\sim\mathbb{F}_2}[Z_S = a \wedge Z_U = b] &= \Pr_{x_1,\dots,x_r\sim\mathbb{F}_2}[Z_S = a \mid Z_U = b] \Pr_{x_1,\dots,x_r\sim\mathbb{F}_2}[Z_u = b] \\ &= \Pr_{x_1,\dots,x_r\sim\mathbb{F}_2}[Z_S = a] \Pr_{x_1,\dots,x_r\sim\mathbb{F}_2}[Z_u = b],\end{aligned}$$

as desired. □

While this example may seem like a forced generalization of the previous one, it amazingly allows us to efficiently[1] generate n pairwise independent variables on $\mathbb{F}_2$ from $r = log(1 + n)$ uniform bits. However, we aren't creating randomness from nothing, since we quickly notice that if we consider $(Z_1, \dots, Z_n)$ as an element of $\mathbb{F}_2^n$, then the support generated by our example only has $2^r = 2^{\log(1+n)} = n + 1$ values in it (since this is how many possible inputs our algorithm takes), yet $|\mathbb{F}_2^n| = 2^n$, so our support is exponentially smaller than that of a uniform distribution over $\mathbb{F}_2^n$. This sparsity of our pairwise independent distribution on $\mathbb{F}_2^n$ illustrates how much stronger regular independence is.

Eventually, we will use this technique of blowing up randomness to help us de-randomize algorithms, but first we will generalize this construction from RVs on $\mathbb{F}_2$ to RVs on larger domains. To do this, we will introduce the useful notion of *universal hash functions*.

Definition 2.4 (Family of Universal Hash Functions)**.** *The collection $\mathcal{H} = \{h : [N] \to [M]\}$ is a family of universal hash functions (UHFs) if for all $x, y \in [N]$ and all $a, b \in [M]$ we have that*

1. $\Pr_{h\in\mathcal{H}}[h(x) = a] = \frac{1}{M}$
2. $\Pr_{h\in\mathcal{H}}[h(x) = a \wedge h(y) = b] = \frac{1}{M^2}$.

Immediately, we see that we can construct a collection of pairwise independent RVs simply by sampling $h \sim \mathcal{H}$ and taking $X_i = h(i)$ for $i \in [N]$, and that doing this sampling only requires $\log(|\mathcal{H}|)$ random bits (since this is how many bits it takes to enumerate the elements of $\mathcal{H}$).

Now, of course, the question that remains is how to construct a family of UHFs efficiently.

Example 2.5 (Construction of a family of UHFs)**.** *Let $N = 2^n$ for some $n \in \mathbb{N}$ and fix our base field as $\mathbb{F}_{2^n}$ (which, we recall, is isomorphic to $\mathbb{F}_2^n$), so $N = |\mathbb{F}_{2^n}|$. To remain consistent with our previous notation, it will be useful to view $\mathbb{F}_{2^n}$ as $[N]$. We then sample $c, d \sim \mathbb{F}_{2^n}$ and define a hash function by the linear expression*

$$\begin{aligned}h_{c,d} : [N] &\to [N] \\ x &\mapsto cx + d\end{aligned}$$

The collection $\mathcal{H} = \{h_{c,d}\}_{c,d\in\mathbb{F}_{2^n}}$ is a family of UHFs that can be sampled with $O(\log N)$ bits.

Proof. Take any $x, y, a, b \in \mathbb{F}_{2^n}$ such that $x \neq y$. Note that to sample from $\mathcal{H}$, all we need to do is to sample c and d from $\mathbb{F}_{2^n}$, which requires $n + n = 2n = 2\log N = O(\log N)$ bits. Thus, using the

[1]Here, by efficient we mean in terms of n. I.e., we have to compute n XOR values, each of which looks at $r = \log(1 + n)$ bits, which in total takes $n\log(1 + n) = O(\mathsf{poly}(n))$ time.

definition of $h_{c,d}$, we can compute

$$\begin{aligned}\Pr_{h\in\mathcal{H}}[h(x)=a\wedge h(y)=b] &= \Pr_{c,d\sim\mathbb{F}_{2^n}}[h_{c,d}(x)=a\wedge h_{c,d}(y)=b]\\ &= \Pr_{c,d\sim\mathbb{F}_{2^n}}[cx+d=a\wedge cy+d=b]\\ &= \Pr_{c,d\sim\mathbb{F}_{2^n}}\left[c=\frac{b-a}{y-x}\wedge d=\frac{bx-ay}{x-y}\right].\end{aligned}$$

But both $\frac{b-a}{y-x}$ and $\frac{bx-ay}{x-y}$ are just arbitrary, fixed elements of $\mathbb{F}_{2^n}$, so the probability of c and d hitting them is exactly $\frac{1}{N}$ and are independent events, giving us that

$$\begin{aligned}\Pr_{c,d\sim\mathbb{F}_{2^n}}[h_{c,d}(x)=a\wedge h_{c,d}(y)=b] &= \Pr_{c,d\sim\mathbb{F}_{2^n}}\left[c=\frac{b-a}{y-x}\wedge d=\frac{bx-ay}{x-y}\right]\\ &= \Pr_{c,d\sim\mathbb{F}_{2^n}}\left[c=\frac{b-a}{y-x}\right]\Pr_{c,d\sim\mathbb{F}_{2^n}}\left[d=\frac{bx-ay}{x-y}\right],\end{aligned}$$

giving us that $\mathcal{H}$ is indeed a family of UHFs. □

With this construction, we have shown the following claim.

Claim 2.6. *There exists an efficient algorithm that takes $2\log N$ random bits and produces a pairwise independent distribution on $[N]$.*

Although we will not show it here, this can be generalized to hash functions with domain $[N]$ and range $[M]$ while requiring $\log N+\log M$ random bits. We will use this result shortly, so we state it here as a lemma.

Lemma 2.7. *There exists an efficient algorithm that takes $\log N+\log M$ random bits and produces a pairwise independent distribution on $[M]$.*[2]

In comparison, if we attempted the naïve approach of randomly picking a hash functions, one would require $N\log N$ bits. Again, this shows how far pairwise independence is from (complete) independence.

3 Pairwise Randomness and BPP

Now we circle back to the question we asked at the beginning of lecture, namely Question 1: Can we construct another algorithm that uses less randomness than $\mathcal{B}$ to achieve the same $1-\varepsilon$ error rate? Recall our definition of $\mathcal{B}$ in Algorithm 1, where we sample $r^1,\dots,r^t$ i.i.d from $\{0,1\}^{\mathsf{poly}(|x|)}$ and then take $\mathrm{Maj}(Y_1,\dots,Y_t)$ as our result where $Y_i=\mathcal{A}(x,r^i)$. After introducing pairwise independent RVs, a reasonable question would be what occurs when we take $r^1,\dots,r^t$ to be pairwise independent instead of independent. Call this modification of $\mathcal{B}$ with pairwise independent samples $\mathcal{B}_2$.

For consistency, since we can consider $\{0,1\}^{\mathsf{poly}(|x|)}$ as $\mathbb{F}_{2^{\mathsf{poly}(|x|)}}$, let $R=2^{\mathsf{poly}(|x|)}$ so that we may again view $\mathbb{F}_{2^{\mathsf{poly}(|x|)}}$ as $[R]$. With regular independence, to sample $r^1,\dots,r^t$ from $[R]$ would require $t\log R$ bits (since sampling each r^i takes $\log R$ bits). On the other hand, applying Lemma 2.7 with $N=t$ and $M=R$ gives us that we can sample pairwise independent $r^1,\dots,r^t$ from $[R]$ only using $\log t+\log R$ random bits, many fewer than $t\log R$.

[2]For those wondering why we cannot just make N small and use few random bits to get a pairwise independent distribution on $[M]$, recall that N is the number of elements that we have to be able to hash (i.e., the number of pairwise independent variables that we want in the end).

However, now that we have changed our sampling strategy for $r^1, \ldots, r^t \sim [R]$ from independent to pairwise independent, it is reasonable to expect the value of t (the number of pairwise independent variables that we wish to sample) to differ from that of Algorithm 1, where we had $t = \log \frac{1}{\varepsilon}$. To find a new value for t in $\mathcal{B}_2$, we will perform a similar analysis as the one we originally did for $\mathcal{B}$ and use a tail bound.

Take any $L \in \mathsf{BPP}$ and, without loss of generality, assume $x \in L$ (the proof in the other case is symmetrical. Recall from Algorithm 1 that $Y_i = \mathcal{A}(x, r^i)$ and define $Y = \sum_{i=1}^{t} Y_i$. Then, the probability that $\mathcal{B}_2$ is correct on x is

$$\Pr_{r^1,\ldots,r^t \sim [R]}[\mathcal{B}_2(x) = L(x)] = \Pr_{r^1,\ldots,r^t \sim [R]}\left[Y \geq \frac{t}{2}\right].$$

Furthermore, by the linearity of expectation we see that

$$\begin{aligned}
\mathop{\mathbb{E}}_{r^1,\ldots,r^t \sim [R]}[Y] &= \mathop{\mathbb{E}}_{r^1,\ldots,r^t \sim [R]}\left[\sum_{i=1}^{t} Y_i\right] \\
&= \sum_{i=1}^{t} \mathop{\mathbb{E}}_{r^1,\ldots,r^t \sim [R]}[Y_i] \\
&\geq \frac{2}{3}t,
\end{aligned}$$

where the last line follows from our definition of BPP. We now know that we want to avoid the case that $Y < \frac{1}{2}$ and that $\mathbb{E}_{r^1,\ldots,r^t \sim [R]}[Y] \geq \frac{2}{3}t$, so we could reasonably ask for the probability of Y being within $\frac{t}{10}$ of its mean. In other words, we would like to upper bound[3]

$$\Pr_{r^1,\ldots,r^t \sim [R]}\left[\left|Y - \frac{2}{3}t\right| > \frac{t}{10}\right]. \tag{1}$$

Previously, for $\mathcal{B}$ we used the Chernoff bound to (1), but in this case that is no longer an option as the $r^1, \ldots, r^t \sim [R]$ are only pairwise independent. Instead, we will use Chebyshev's inequality, which requires us to compute $\mathrm{Var}(Y)$. Thankfully, since the Y_i's are pairwise independent (because the r^i's are), we have that $\mathrm{Cov}(Y_i, Y_j) = 0$ for $i \neq j$. Then sing the fact that each Y_i is a Bernoulli RV with $p = \frac{2}{3}$, meaning that $\mathrm{Var}(Y_i) = \frac{2}{3}\left(1 - \frac{2}{3}\right) = \frac{2}{9}$, we can compute

$$\begin{aligned}
\mathrm{Var}(Y) &= \mathrm{Var}\left(\sum_{i=1}^{t} Y_i\right) \\
&= \sum_{i=1}^{t} \mathrm{Var}(Y_i) + \sum_{i \neq j} \mathrm{Cov}(Y_i, Y_j) \\
&= \sum_{i=1}^{t} \mathrm{Var}(Y_i) \leq \frac{2t}{9}
\end{aligned}$$

We can now apply Chebyshev's inequality to (1) as

$$\Pr_{r^1,\ldots,r^t \sim [R]}\left[\left|Y - \frac{2}{3}t\right| > \frac{t}{10}\right] \leq \frac{100}{t^2} \cdot \frac{2t}{9} = \frac{c}{t}$$

[3]Upper bounding this value does also limit the probability that we are in the upper end of the tail, but this is an acceptable relaxation as being near the mean of Y is sufficient in our case.

Thus, to upper bound this value by ε we need $\frac{c}{t} \leq \varepsilon$, so we choose $t \geq \frac{c}{\varepsilon} = \Omega\left(\frac{1}{\varepsilon}\right)$, meaning that $\mathcal{B}_2$ repeats $\mathcal{A}$ on the order of $\Omega\left(\frac{1}{\varepsilon}\right)$ times and uses $\log t + \log R = \log \frac{1}{\varepsilon} + \log R + O(1)$ random bits. Putting this together with the facts about $\mathcal{B}$ in Table 1 gives us

Algorithm	Accuracy	Runtime	Randomness
$\mathcal{B}$	$1-\varepsilon$	$c \cdot T(n) \cdot \log \frac{1}{\varepsilon}$	$c \cdot R(n) \log \frac{1}{\varepsilon}$
$\mathcal{B}_2$	$1-\varepsilon$	$c' \cdot T(n) \cdot \frac{1}{\varepsilon}$	$R(n) + \log \frac{1}{\varepsilon} + O(1)$

Table 2: Accuracy, runtime, and randomness usage of $\mathcal{B}$ and $\mathcal{B}_2$.

In conclusion, we have answered a definitive YES to Question 1, showing that we can indeed use fewer random bits than $\mathcal{B}$ while achieving the same $1-\varepsilon$ error rate. Nevertheless, this was at the cost of an exponential increase in runtime. We can further explore this trade-off between runtime and randomness by defining $\mathcal{B}_k$ to be analogous to the algorithm that we have been using, but with the $r^1, \ldots, r^t$ used in it drawn k-wise independently (where we define k-wise independence as any collection of k RVs being independent, so pairwise independence is simply 2-wise independence). These $\mathcal{B}_k$'s will essentially interpolate between $\mathcal{B}$ and $\mathcal{B}_2$, and their runtime and randomness usage can be analyzed similarly as to what we have done so far, although higher moment methods are required as k increases.