Pseudorandom generators: Derandomizing BPP

October 3, 2023

1 Pseudorandom generator

Definition 1. A distribution R over $\{0,1\}^m$ is (S,ϵ)-*pseudorandom* if for every circuit C of size $\leq S$,

$$|Pr_{x\sim R}[C(x)=1] - Pr_{x\sim U_m}[C(x)=1]| < \epsilon.$$

Thus, to every small enough circuit, R looks close to being uniformly at random. Here, the first parameter S denotes the size of the circuit, and ϵ denotes the distinction tolerance.

Definition 2. Let $S : \mathbb{N} \to \mathbb{N}$ be some "stretch function". A E-time computable function $G : \{0,1\}^* \to \{0,1\}^*$ is called a *pseudorandom generator* if $|G(z)| = S(|z|)$ for every string z, and for every $\ell \in \mathbb{N}$, the distribution $G(U_\ell)$ is $(S(\ell)^3, 1/10)$-pseudorandom.

Here S denotes the stretch parameter, which also *relates polynomially* to the circuit size of the distinguishing circuit. Thus S plays two roles in the above definition.

Discussion on the parameters: The "stretch" acheived on any ℓ-length string is $S(\ell)$. The collection of such strings produced (a distribution supported on2^ℓmany strings out of $2^{S(\ell)}$many strings overall) is resilient against circuits of size $S(\ell)^3$.

The choices of the constants 3 and 1/10 are arbitrary, but are made fixed for easing the discussion below.

A degenerate case: Suppose G is the identity function, $G(z) = z$. Then the output distribution is U_ℓ. Is U_ℓ a $(\ell^3, 1/10)$-pseudorandom distribution? Yes, because the difference in Definition 1 will be 0 for circuits of any size.

2 Derandomization from Pseudorandom generators

We first see how we can use pseudorandom generators to derandomize BPP. In the next section, we see how to construct pseudorandom generators from average-case hard functions.

The idea in the present section is quite simple: brute-force enuemerate the entire range of the PRG, and then run the randomized algorithm on all generated pseudorandom seeds instead of the uniformly random coins.

Theorem 3. *If there is an $S(\ell)$-pseudorandom generator, then, for every polynomial-time computable function $\ell : \mathbb{N} \to \mathbb{N}$,* ***BPTIME***$(S(\ell(n)) \subseteq DTIME(2^{c\ell(n)})$.

Discussion of the parameters. Suppose we want the target to be $DTIME(n^c)$. Then clearly, $\ell(n) = \log(n)$. If $S(\ell(n)) = n^c$on the left, then this means that the stretch function S must be exponential, *i.e.* $S(k) = 2^k$. So BPP=P using the above result requires pseudorandom generators with exponential stretch. (It also follows that they must be indistinguishable for $S(\ell(n))^3 = n^{3c}$-sized circuits.)

Proof. Suppose $L \in$ **BPTIME**$(S(\ell(n))$. Then there is an algorithm $A : \{0,1\}^* \to \{0,1\}^*$that on input $x \in \{0,1\}^n$, runs in time $cS(\ell(n))$ for some constant c, and satisfies

$Pr_{r\sim\{0,1\}^{S(\ell(n))}}[A(x,r) = L(x)] \geq \frac{2}{3}$.

Consider the following deterministic algorithm B. On input x, over all $z \in \{0,1\}^{\ell(n)}$, it runs $A(x, G(z))$ and outputs their majority.

We claim that for all sufficiently large n, the fraction of z for which $A(x, G(z)) = L(x)$ is at least $2/3 - 0.1$. If this claim is true, then note that the algorithm B is always correct on x, hence $L \in DTIME(2^{c\ell(n)})$. Suppose the claim is false, and there is an infinitely many xs for which

$$Pr[A(x, G(z)) = L(x)] < 2/3 - 0.1.$$

Then, for infinitely many lengths, there are circuits which hardcode the appropriate x to do the following: on input r, output $A(x, r)$. Then

$$\left|\left[Pr_{z\sim\{0,1\}^{\ell(n)}}(A(x, G(z)) = L(x)\right] - \left[Pr_{r\sim U_{S(\ell(n))}}(A(x, r) = L(x)\right]\right| \geq 0.1,$$

which violates the assumption that G is a pseudorandom generator. □

We now see a construction of a simple pseudorandom generator.

Theorem 4. *Suppose there is a function in E with average-case hardness n^4. Then there is an $S(\ell)$-pseudorandom generator for $S(\ell) = \ell + 1$.*

That is, given any binary string "seed" z, the pseudorandom generator outputs a string of length one more than that of z. The set of output strings as z ranges over n-length strings will be a pseudorandom distribution over $\{0,1\}^{n+1}$.

Proof. Le $G(z) = z \circ f(z)$. Then we show that G is $((\ell+1)^3, 1/10)$-pseudorandom.

If there is a circuit C of size $2(\ell+1)^3 < \ell^4$such that

$$Pr_{z\sim\{0,1\}^n}[C(z_1 \dots z_n) = f(z)] > \frac{1}{2} + \frac{1}{20(\ell+1)} > \frac{1}{2} + \frac{1}{\ell^4},$$

then this violates the assumption that f is hard on average for circuits of size $< n^4$. □

---

3 Pseudorandom generators from strong average-case hardness

We describe two initial ideas for extending the observation in Theorem 4. We could consider the pseudorandom generator which extends z by two bits - first by using the first half of the string and evaluating a hard function on it, and then by using the second half of the string and evaluating a hard function on it. That is, for every $z \in \{0,1\}^\ell$,

$$G(z) = z_1 \dots z_{\ell/2} \circ f(z_1 \dots z_{\ell/2}) \circ z_{\ell/2+1} \dots z_n \circ f(z_{\ell/2+1} \dots z_\ell).$$

The analysis is similar as before, except that we have to note that $f(z_1 \dots z_{\ell/2})$ and $f(z_{\ell/2+1} \dots z_\ell)$ are independent, hence the first cannot help in predicting the second.

Can we generalize this observation, dividing z into a fixed number of blocks, and interspersing the values of f on those blocks? This will produce a psuedorandom string from a random z. However, the output string

is going to be only at most twice as long as the original string (since the extra number of bits will be equal to the number of blocks, and there are at most $|z|$ many blocks).

We need an *exponential stretch.*

This is where the Nisan-Wigderson generator comes into the picture. Some of the ideas are similar to the blockwise encoding. The major conceptual difference is that we will reuse bits in many blocks. Thus, the blocks will "overlap" significantly, and the overlapping blocks may not even occur adjacent to each other in z. Formally, the concept we use is the notion of a *combinatorial design.*

3.1 The Nisan-Wigderson Pseudorandom generator.

Definition 5. Let $\mathcal{I} = \{I_1, \ldots, I_m\}$be a family of subsets of the set of integers $\{1, 2, \ldots, \ell\}$ where every set in the family has exactly n integers. ($n \leq \ell$). Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function.

The Nisan-Wigderson pseudorandom generator [1] is the function $NW_{f,\mathcal{I}} : \{0,1\}^\ell \to \{0,1\}^m$ defined by

$$NW_{f,\mathcal{I}}(z) = f\left(z_{i_1}\right) \circ \cdots \circ f\left(z_{i_m}\right),$$

where z_{i_j} is the substring obtained by restricting z to the indices from I_j.

Discussion on the parameters: The maximum value of m is $\binom{\ell}{n}$. This value is maximal when n is $\ell/2$, where the number of subsets can be $\binom{\ell}{\ell/2} = 2^{\Theta(\ell)}$ by Stirling's approximation. So the generator can stretch an ℓ-length string to length $2^{\Theta(\ell)}$.

For the generator to produce pseudorandom outputs from random z, the following conditions must be met:

1. f must satisfy hardness assumptions,
2. the family of sets $\mathcal{I}$ must form a *combinatorial design*, defined below.

Definition 6. Let $d < n < \ell$. A (d, n, ℓ)-*combinatorial design* is a family of sets $\mathcal{I} = \{I_1, \ldots, I_m\}$of subsets of $\{1, 2, \ldots, \ell\}$,where each subset has exactly n elements, and two distinct sets have at most d common elements.

We now show that

1. combinatorial designs with $2^{d/10}$ subsets can be constructed in time $2^{O(\ell)}$time (Lemma 7)
2. If a function's average-case hardness is 2^{2d}, then the Nisan-Wigderson generator is $\left(\frac{H_a(f)}{10}, \frac{1}{10}\right)$-pseudorandom.

Here the stretch function is the number of subsets, which is $2^{d/10}$. The circuit size bound is $\frac{H_a(f)}{10} = \frac{2^{2d}}{10}$. Thus the stretch function and the circuit size are polynomially related.

Lemma 7. *There is an algorithm, which, given $\langle d, n, \ell\rangle$where $d < n$ and $10n^2/d < \ell$, runs for $2^{O(\ell)}$steps, and outputs an (d, n, ℓ)-combinatorial design containing $2^{d/10}$subsets of $\{1, 2, \ldots, \ell\}$.*

Proof. Start with $\emptyset$and keep adding sets to the collection as long as they satisfy the intersection condition and the cardinality condition, till we get $2^{d/10}$subsets. We have to argue that under the assumption that $d < n$ and $10n^2/d < \ell$, there are enough such sets.

We argue that if $\ell = 10n^2/d$ and $\{I_1, \ldots, I_m\}$ is a collection of n-element subsets of $\{1, 2, \ldots, \ell\}$where pairwise intersections of distinct sets have at most d elements, if $m < 2^{d/10}$, then there is an n-element subset I of $\{1, 2, \ldots, \ell\}$such that it has at most d common elements with the previously selected sets.

For this, instead of counting such sets, we use the probabilistic method. Select I at random by selecting n elements independently at random, with each element chosen to be in I with probability $2n/\ell$. Then the expected size of I is $2n$, and the expected size of any intersection, by independence, is $|I \cap I_j| = 2n^2/\ell < d/5$ for any $1 \leq j \leq m$. Hence by Chernoff bound,

$$Pr\left[|I| > n\right] > 0.9$$

$$Pr\left[|I \cap I_j| \geq d\right] \leq 0.5 \times 2^{d/10}$$

The probability that I will have at least n elements and have at most d elements in common with all the remaining elements, is at least 0.4. This completes the proof. □

References

[1] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.