CS 880: Advanced Complexity Theory 3/14/2008

Lecture 21: Small-Bias Pseudorandom Generators

Instructor: Dieter van Melkebeek Scribe: Dalibor Zelený

Today we discuss small-bias pseudorandom generators. The goal of these generators is to fool linear functions, i.e., characters. We present one way to construct such pseudorandom generators, and talk about some applications of small-bias pseudorandom generators.

1 Pseudorandom Generators

We start by discussing the notion of a *pseudorandom generator* (PRG). Intuitively, the goal is to take a short purely random string called the *random seed* of length l, apply some function G_n to it, and obtain a longer *pseudorandom* string of length n. We would like G_n to be efficiently computable.

We design PRGs to *fool* certain tests, meaning that the tests being fooled can distinguish between a uniform distribution and a distribution induced by the PRG only with small probability. We formalize that in the following definition. We use the notation $x \leftarrow D$ to indicate that x comes from distribution D. We use U_n to denote the uniform distribution.

Definition 1. *A function $G_n : \{0,1\}^l \to \{0,1\}^n$* fools *test $T : \{0,1\}^n \to \{0,1\}$ to within ϵ if $|\Pr_{x \leftarrow U_n}[T(x)] - \Pr_{x \leftarrow G_n}[T(x)]| \leq \epsilon$.*

Our goal is to construct efficient PRGs with small seed length that fool all linear tests. The latter condition turns out to be equivalent to the PRG having small bias. We first introduce the notion of bias.

2 Bias

We start with the bias of a random variable. Afterwards, we give some examples. We also define ϵ-biased pseudorandom generators and state some of their properties.

Definition 2. *Consider a random variable that has two values, say $X : \Omega \to \{0,1\}$. We define the* bias *of X as* $\text{Bias}(X) = |\Pr[X = 0] - \Pr[X = 1]|$.

Notice that the bias of a function depends on the underlying probability distribution of the inputs. For example, consider the uniform distribution U_n. We compute the bias of the character χ_S when the input comes from the uniform distribution. We denote this by $\text{Bias}(\chi_S(U_n))$. It is easy to see that $\text{Bias}(\chi_\emptyset(U_n)) = 1$ because the character corresponding to the empty set is the constant function 1. In fact, $\text{Bias}(\chi_\emptyset(D)) = 1$ for any distribution D. For sets $S \neq \emptyset$, we get $\text{Bias}(\chi_S(U_n)) = 0$ because all the other characters are balanced and inputs come from the uniform distribution.

We extend to notion of bias to PRGs as follows.

Definition 3. *We say that a function $G_n : \{0,1\}^l \to \{0,1\}^n$ is an ϵ-*biased pseudorandom generator *(PRG) if for all nonempty subsets $S \subseteq [n]$ we have* $\text{Bias}(\chi_S(G_n)) \leq \epsilon$.

By the above observations, the requirement in Definition 3 is equivalent to the condition that $|\text{Bias}(\chi_S(U_n)) - \text{Bias}(\chi_S(G_n))| \leq \epsilon$ for every $S \subseteq [n]$.

We show that the notion of a small-bias PRG coincides with the notion of a PRG fooling all linear tests. A linear test over GF(2) entails taking the inner product of a random variable of length n with some fixed vector a of length n.

Proposition 1. *A PRG G_n fools all linear tests to within ϵ if and only if G_n is a (2ϵ)-biased PRG.*

Proof. For any random variable X of length $\{0,1\}^n$ and any $a \neq 0$ of length n, we have

$$\begin{aligned}
\Pr[\langle a, X\rangle = 0] - \Pr[\langle a, U_n\rangle = 0] &= \Pr[\langle a, X\rangle = 0] - \frac{1}{2} \\
&= \frac{1}{2}(\Pr[\langle a, X\rangle = 0] - \Pr[\langle a, X\rangle = 1]) \qquad (1)\\
&= \frac{1}{2}\text{E}[\chi_S(X)],
\end{aligned}$$

where $S = \{i \in [n] \mid a_i = 1\}$. In order to get (1), we view the $\frac{1}{2}$ from the previous line as $\frac{1}{2}(\Pr[\langle a, X\rangle = 0] + \Pr[\langle a, X\rangle = 1])$. □

We can view $\text{Bias}(\chi_S(G_n))$ as $|\text{E}_{x\leftarrow G_n}[\chi_S(x)]|$ because the range of χ_S is $\{-1, 1\}$. We also have

$$\text{E}_{x\leftarrow G_n}[\chi_S(x)] = \sum_x \Pr[G_n = x]\chi_S(x). \qquad (2)$$

View $\Pr[G_n = x]$ as a function g_n from $\{0,1\}^n \to [0,1]$ that indicates the probability whether a given string x is picked from the distribution G_n. We define a similar function u_n for the uniform distribution. Now we can rewrite the right-hand side of (2) as

$$2^n \text{E}_x[g_n(x)\chi_S(x)] = 2^n \widehat{g_n}(S), \qquad (3)$$

so if the bias is at most ϵ, we have

$$|\widehat{g_n}(S)| \leq \frac{\epsilon}{2^n} \qquad \text{whenever } S \neq \emptyset. \qquad (4)$$

We can now compute how far is the distribution induced by the small-bias PRG G_n from the uniform distribution. First we consider the 2-norm as our measure of distance. Notice that $\widehat{u_n}(\emptyset) = 1$, $\widehat{u_n}(S) = 0$ for all $S \neq \emptyset$, and $\widehat{g}(\emptyset) = 1$. We express both g_n and u_n using a Fourier expansion and then bound each term of the summation using (4) to get

$$\|g_n - u_n\|_2^2 = \text{E}_x\left[\sum_S |(\widehat{g_n}(S) - \widehat{u_n}(S))\,\chi_S(x)|^2\right] = \text{E}_x\left[\sum_{S\neq\emptyset} |\widehat{g_n}(S)|^2\right] \leq (2^n - 1)\frac{\epsilon^2}{2^{2n}},$$

which gives us

$$\|g_n - u_n\|_2 \leq \frac{\epsilon}{2^n} \cdot \sqrt{2^n - 1}. \qquad (5)$$

We see that in 2-norm, the distribution produced by the small-bias PRG is not far from the uniform distribution. We can also use (6) and properties of norms to bound

$$\|g_n - u_n\|_1 \leq \|g_n - u_n\|_2 \leq \frac{\epsilon}{2^n} \cdot \sqrt{2^n - 1}. \qquad (6)$$

This means that for any event A, $|\Pr_{x\leftarrow G_n}[x \in A] - \Pr_{x\leftarrow U_n}[x \in A]| \leq \epsilon \cdot \sqrt{2^n - 1}$.

Next, we construct one particular small-bias PRG whose seed length is logarithmic in n/ϵ.

3 Construction of a Small-Bias Pseudorandom Generator

Consider a string of length l both as a vector in $(\mathrm{GF}(2))^l$ and an element of $\mathrm{GF}(2^l)$. We use both these representations to construct a small-bias PRG. We represent our PRG by a function $G_n : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^n$ that maps strings of length $2l$ to strings of length n. We view the input as two vectors x and y, both of length l. To get the i-th bit of the pseudorandom sequence, we compute $\langle x^{i-1}, y\rangle$, the inner product of the i-th power of x in $\mathrm{GF}(2^l)$ viewed as a vector in $(\mathrm{GF}(2))^l$ with a string $y \in (\mathrm{GF}(2))^l$. The values of i range from 1 to n. We show that this is indeed a small-bias PRG.

Claim 1. *The PRG we just described has bias $\frac{n-1}{2^l}$.*

Proof. We compute the bias of the function $\chi_S(G_n)$ when its input strings x and y come from the uniform distribution. We have

$$\begin{aligned}
\mathrm{Bias}(\chi_S(G_n)) &= \left|\mathrm{E}_{x,y}\left[\chi_S\left(\left(\langle x^{i-1}, y\rangle\right)_{i=1}^{n}\right)\right]\right| && (7)\\
&= \left|\mathrm{E}_{x,y}\left[\prod_{i\in S}(-1)^{\langle x^{i-1}, y\rangle}\right]\right| && (8)\\
&= \left|\mathrm{E}_{x}\left[\mathrm{E}_{y}\left[(-1)^{\langle p_S(x), y\rangle}\right]\right]\right| && (9)\\
&\leq \frac{n-1}{2^l} && (10)
\end{aligned}$$

Equation (7) comes from the definition of bias. The argument to χ_S is a string of length n obtained by the concatenation of $\langle x^{i-1}, y\rangle$ for all i between 1 and n. We use the definition of χ_S to rewrite the quantity inside of the expectation as a product, which yields (8). Now define $p_S(x) = \sum_{i\in S} x^{i-1}$. This modification yields (9) by linearity of the inner product. When x is a root of $p_S(x)$, the inner expectation (over y) is 1 because the exponent of (-1) in (9) is the inner product of an all-zero vector with y, which gives zero, so $(-1)^{\langle p_S(x), y\rangle} = 1$ regardless of y. For all other x, we get $(-1)^{\langle p_S(x), y\rangle} = -1$ and $(-1)^{\langle p_S(x), y\rangle} = 1$ with equal probability. Since p_S is a nonzero polynomial of degree at most $n-1$, it has at most $n-1$ roots, which makes the inner expectation evaluate to 1 for $n-1$ values of x that are roots of p_S, and to zero for all the other values of x, so the outer expectation is at most $(n-1)/2^l$. This gives us (10).

We want the bias to be at most ϵ, so pick $l \approx \log \frac{n}{\epsilon}$, which gives us a seed length of about $2\log\frac{n}{\epsilon}$. For such value of l, G_n has bias $\frac{n-1}{2^l}$. □

The number of possible seeds of the PRG we constructed is $(n/\epsilon)^2$. Thus, in order to derandomize algorithms that use this PRG, we need to cycle through $(n/\epsilon)^2$ seeds. This derandomization is efficient provided that we have an irreducible polynomial of degree l or some other representation of $\mathrm{GF}(2^l)$ that can be used efficiently. Irreducible polynomials over $\mathrm{GF}(2)$ are known for many degrees. If we don't know one, we can cycle through all polynomials of degree l with coefficients in $\mathrm{GF}(2)$ in order to find one that is irreducible over $\mathrm{GF}(2)$.

Suppose we have an algorithm whose running time is t and that uses r random bits. Then we can run the derandomized version of this algorithm using time $t\cdot\left(\frac{r}{\epsilon}\right)^2$. In particular, if a randomized algorithm runs in polynomial time, its derandomized version will also run in polynomial time.

4 Applications of Small-Bias Pseudorandom Generators

We mention a few applications of small-bias PRGs. Our first two applications relate to material presented in earlier lectures. We show how to approximate Fourier coefficients using a deterministic algorithm, and then apply this approximation to learning various concept classes. Our third application is in hardness of approximation. We show that the problem MAX-QE cannot be approximated within a factor of $1/2 + 1/\text{poly}$ or better unless $\text{P} = \text{NP}$.

4.1 Deterministically Approximating Fourier Coefficients

When we discussed learning, we saw sampling as a means of approximating Fourier coefficients of a function whose Fourier spectrum is concentrated on a known small set $\mathcal{S}$ of subsets of $[n]$. We derandomize this approximation. This will make the approximation use membership queries in place of labeled samples.

We approximate $\widehat{f}(S) = E_{x \leftarrow U_n}[f(x)\chi_S(x)]$ by $E_{x \leftarrow G_n}[f(x)\chi_S(x)]$. We are interested in the difference between the actual Fourier coefficient for a particular set S and our approximation. We have

$$\begin{aligned}
\left|\widehat{f}(S) - \mathrm{E}_{x\leftarrow G_n}\left[(f(x)\chi_S(x)\right]\right| &= \left|\widehat{f}(S) - \mathrm{E}_{x\leftarrow G_n}\left[\left(\sum_{T\subseteq[n]}\widehat{f}(T)\chi_T(x)\right)\chi_S(x)\right]\right| && (11)\\
&= \left|\widehat{f}(S) - \sum_{T\subseteq[n]}\widehat{f}(T)\mathrm{E}_{x\leftarrow G_n}\left[\chi_{S\triangle T}(x)\right]\right| && (12)\\
&\leq \sum_{T\subseteq[n]-S}\left|\widehat{f}(T)\right|\left|\mathrm{E}_{x\leftarrow G_n}\left[\chi_{S\triangle T}(x)\right]\right| && (13)\\
&\leq \sum_{T\subseteq[n]-S}\left|\widehat{f}(T)\right|\cdot\epsilon && (14)
\end{aligned}$$

We get (11) from the Fourier expansion of $f(x)$. We move the expectation in (11) inside and observe that $\chi_S\chi_T$ is the same as $\chi_{S\triangle T}$ where $\triangle$ denotes the symmetric difference of two sets. This yields (12). When $S = T$, $S\triangle T = \emptyset$, so the character $\chi_{S\triangle T}$ is the constant function, and the term in the summation in (12) with $\widehat{f}(T)$ for $T = S$ cancels with the $\widehat{f}(S)$. This observation and the triangle inequality give us (13). Since G_n is a small-bias generator and $S \neq T$ in (13), $|\mathrm{E}_{x\leftarrow G_n}[\chi_{S\triangle T}(x)]| \leq \epsilon$ for all relevant T. This gives us an upper bound (14) on the error in our approximation of $\widehat{f}(S)$ using G_n instead of the uniform distribution in the definition of $\widehat{f}(S)$.

If we want the upper bound in (14) to be at most δ, we pick $\epsilon \leq \delta/(\sum_{T\subseteq[n]-S}|\widehat{f}(T)|)$. Now we cycle through all random seeds of length $2\log(n/\epsilon)$ and get an approximation of $\widehat{f}(S)$ within a factor of δ in time $\text{poly}(n, \frac{1}{\delta}, \sum_{T\subseteq[n]}|\widehat{f}(T)|)$ using membership queries.

4.2 Learning

With the derandomization from the last section at hand, we can derandomize the learning algorithms we saw earlier in the course. These modified algorithms learn from membership queries.

Recall that we designed learning algorithms for classes of concepts whose Fourier spectra were concentrated on some set $\mathcal{S}$ of subsets of $[n]$. We described two ways of learning those concepts.

When we knew the set $\mathcal{S}$ exactly, we gave a learning algorithm that learned only from labeled samples. In the case when $\mathcal{S}$ was unknown, we assumed a bound M on the size of $\mathcal{S}$, and used a combination of labeled samples and membership queries to learn the concepts.

First let's focus on learning from samples. Recall that we learned a concept c by learning an approximation f such that $\widehat{f}(S) = 0$ for all $S \notin \mathcal{S}$, and $\widehat{f}(S) = \widehat{c}(S)$ for all $S \in \mathcal{S}$. We further approximated the Fourier coefficients $\widehat{f}$ by $\widehat{g}$. For each $S \in \mathcal{S}$, we looked at $f(x)\chi_S(x)$ for a certain amount of samples x, and averaged them to get $\widehat{g}(S)$. We set $\widehat{g}(S) = 0$ for all $S \notin \mathcal{S}$. This gave us a possibly non-Boolean function g. Finally, we argued that rounding g to the nearest Boolean function h yielded a Boolean function that wasn't too far from c in terms of the squared 2-norm.

Tracing through the algorithm, we can replace some of the steps and design a deterministic algorithm that approximately learns c using membership queries. We can approximate the Fourier coefficients $\widehat{f}$ using the algorithm described in the previous section. Since the algorithm is deterministic, it uses membership queries instead of labeled samples. If we approximate each $\widehat{f}(S)$ within $\delta/|\mathcal{S}|$ in order to get a function g, we have $\|c - g\|_2^2 \leq 2\delta$ using an argument similar to the one done earlier in class. We can approximate each Fourier coefficient in time $\mathrm{poly}(n, |\mathcal{S}|, \frac{1}{\delta}, \sum_T |\widehat{c}(T)|)$, so we find g (and thus h) deterministically in time $\mathrm{poly}(n, |\mathcal{S}|, \frac{1}{\delta}, \sum_T |\widehat{c}(T)|)$ since we need to approximate $|\mathcal{S}|$ coefficients. Thus, we get the following theorem.

Theorem 1. *We can approximately learn concept classes $\mathcal{C}$ with Fourier spectra concentrated on a known set $\mathcal{S}$ of subsets of $[n]$ in deterministic time $\mathrm{poly}(n, |\mathcal{S}|, \frac{1}{\delta}, \sum_T |\widehat{c}(T)|)$ using membership queries.*

In the setting where the set $\mathcal{S}$ was not known, we knew an upper bound M on the size of $\mathcal{S}$. We described a list decoding algorithm for the Hadamard code which used a combination of labeled samples from the uniform distribution and membership queries. We approximated the set $\mathcal{S}$ using a set $\mathcal{S}'$ consisting of all sets S whose Fourier coefficients were larger than a threshold τ. We approximated $\mathcal{S}'$ by a set $\mathcal{S}''$ which we obtained by expanding a binary tree with node labels representing prefixes of characteristic sequences of sets. Let v be a prefix. We defined $\mathcal{S}_v = \{S \mid S \cap \{1, \ldots, |v|\} = v\}$. For a node labeled v, we approximated $\sum_{S \in \mathcal{S}_v} (\widehat{c}(S))^2$. In the end, with high probability, we had a list of subsets of $[n]$ that contained all subsets whose squared Fourier coefficients were greater than τ, but didn't contain any subsets of $[n]$ whose squared Fourier coefficients were less than $\tau/2$.

Recall that for a node labeled v, we had

$$\sum_{S \in \mathcal{S}_v} (\widehat{c}(S))^2 = \mathrm{E}_{x_1, y_1, z \leftarrow U_{n+|v|}} \left[c(x_1 z)\chi_v(x_1) c(y_1 z)\chi_v(y_1) \right], \tag{15}$$

where ab denotes the concatenation of strings a and b.

We slightly abuse notation now. Take a string ab, where a has length $|v|$ and b has length $n - |v|$. We append $n - |v|$ ones to a and prepend $|v|$ ones before b. Then we can view ab as a componentwise product of the two strings a and b (now of length n). Similarly, append $n - |v|$ ones after v to get a string of length n which represents the characteristic sequence of the same set as the original string v. Notice that with this change, the value of $\chi_v(a)$ is still the same, and so is the value of $\chi_T(ab)$. Moreover, we can now write $\chi_T(ab) = \chi_T(a)\chi_T(b)$ where $T \subseteq [n]$.

We use this abuse of notation together with the Fourier expansion of c, linearity of characters, and linearity of expectation to rewrite the right-hand side of (15) as

$$\begin{aligned}
&\mathrm{E}_{x_1,y_1,z\leftarrow U_{n+|v|}}\left[\left(\sum_T \widehat{c}(T)\chi_T(x_1z)\right)\chi_v(x_1)\left(\sum_U \widehat{c}(U)\chi_U(y_1z)\right)\chi_v(y_1)\right]\\
&=\sum_{T,U}\widehat{c}(T)\widehat{c}(U)\mathrm{E}_{x_1,y_1,z\leftarrow U_{n+|v|}}\left[\chi_T(x_1)\chi_T(z)\chi_v(x_1)\chi_U(y_1)\chi_U(z)\chi_v(y_1)\right]\\
&=\sum_{T,U}\widehat{c}(T)\widehat{c}(U)\mathrm{E}_{x_1\leftarrow U_{|v|}}\left[\chi_T(x_1)\chi_v(x_1)\mathrm{E}_{y_1\leftarrow U_{|v|}}\left[\chi_U(y_1)\chi_v(y_1)\mathrm{E}_{z\leftarrow U_{n-|v|}}\left[\chi_T(z)\chi_U(z)\right]\right]\right]\\
&=\sum_{T,U}\widehat{c}(T)\widehat{c}(U)\mathrm{E}_{x_1\leftarrow U_{|v|}}\left[\chi_{T\triangle v}(x_1)\right]\mathrm{E}_{y_1\leftarrow U_{|v|}}\left[\chi_{U\triangle v}(y_1)\right]\mathrm{E}_{z\leftarrow U_{n-|v|}}\left[\chi_{T\triangle U}(z)\right]. \qquad (16)
\end{aligned}$$

where the last line follows because x_1, y_1, and z are independent.

We can determine the quality of our approximation using an analysis similar to the analysis of our approximation of individual Fourier coefficients. Instead of drawing from the uniform distribution, we draw x_1, y_1, and z from the distribution induced by an ϵ-biased PRG. Then we find an upper bound on the absolute value of the difference of the left-hand side of (15) and the expression (16) with the uniform distribution replaced by the distribution induced by an ϵ-biased PRG.

Notice that if $T = U$ and $T \cap \{1,\ldots,|v|\} = v$, all the expectations in (16) become 1. The expectation over z is 1 because $T = U$, and the other two are one because in that case $\chi_T(x_1) = \chi_v(x_1)$ and $\chi_T(y_1) = \chi_v(y_1)$ since the support of x_1 and y_1 is a subset of $\{1,\ldots,|v|\}$. Thus if $T \in \mathcal{S}_v$, the term with $T = U$ and $T \cap \{1,\ldots,|v|\} = v$ cancels with some $S \in \mathcal{S}_v$ in the aforementioned absolute value.

For any other combination of T and U, either $T \neq U$ or $T \cap \{1,\ldots,|v|\} \neq v$, so the character in at least one of the expectations in (16) doesn't correspond to the empty set, which means that at least one of the expectations will be bounded above by ϵ (and the remaining ones by 1) in absolute value. Then, after the cancellations described in the previous paragraph, we can use the triangle inequality to bound the aforementioned absolute value by

$$\sum_{T,U}|\widehat{c}(T)||\widehat{c}(U)|\epsilon = \epsilon\left(\sum_T |\widehat{c}(T)|\right)^2. \qquad (17)$$

The tree we create has at most n/τ nodes. If we want to get the correct tree with probability at least $1-\delta$, we need ϵ in (17) to be at most $\delta\tau/\big(n\big(\sum_T|\widehat{c}(T)|\big)^2\big)$.

In order to approximate (15), we need to use three of the PRGs and cycle through all possible combinations of their seeds. We have to do this because we pick x_1, y_1, and z independently. If we set $\tau = \delta$, the number of seeds for each of the PRGs will be polynomial in n, $\frac{1}{\delta}$, and $\big(\sum_T|\widehat{c}(T)|\big)^2$. We spend polynomial time on each triple of seeds when calculating the approximation, and the number of approximations we have to perform is $n/\tau = n/\delta$. Thus, we get the following theorem.

Theorem 2. *There is a deterministic algorithm that approximately list decodes the Hadamard code in time* $\mathrm{poly}(n, 1/\delta, \sum_T|\widehat{c}(T)|)$ *using membersihp queries.*

4.3 Learning Decision Trees

We can apply the active learning algorithm from the previous section to derandomize algorithms for learning decision trees we saw earlier in the course. Recall that a decision tree of depth d

has at most 4^d nonzero Fourier coefficients, each of which is a multiple of $1/2^d$. We can use the derandomized approximation of (15) with the error from (17) equal to $1/2^{2d+1}$ to create the tree exactly. The leaves of the tree represent the subsets of $[n]$ with nonzero Fourier coefficients. Now we know the set $\mathcal{S}$ on which the Fourier spectrum of the concept we are learning is concentrated. Thus, we can use the learning algorithm from Theorem 1 to learn the decision tree exactly if the algorithm approximates each Fourier coefficient within $1/2^{d+1}$. Since there are at most 4^d nonzero Fourier coefficients, we get the following theorem as a consequence of Theorem 2 and Theorem 1.

Theorem 3. *We can learn decision trees of depth d exactly using a deterministic algorithm that runs in time* $\mathrm{poly}(n, 4^d)$ *and uses membership queries.*

The term $1/\delta$ from the two previous theorems disappeared because δ from those two theorems is approximately $1/4^d$. The term $\sum_T |\widehat{c}(T)|$ disappears because it's bounded above by 2^d, which is less than 4^d. Hence, we can learn decision trees of logarithmic depth exactly using a deterministic algorithm that runs in polynomial time.

4.4 Inapproximability of MAX-QE

In the MAX-QE problem, we are given m quadratic equations over n unknowns. In the decision version of the problem, our goal is to find an assignment of values to the variables such that all the equations are satisfied. In the optimization version, our goal is to come up with an assignment that satisfies as many equations as possible. This problem is NP-hard because we can arithmetize 3-SAT using a polynomial number of quadratic equations. We show that we cannot approximate MAX-QE within a factor of $1/2 + 1/\mathrm{poly}$ or better unless $\mathrm{P} = \mathrm{NP}$.

It is not known whether this inapproximability bound is tight. We can trivially achieve a factor of 1/4, but we don't know of an algorithm that performs better than that.

Theorem 4. *We cannot approximate MAX-QE within a factor of* $1/2 + 1/\mathrm{poly}$ *or better unless* $\mathrm{P} = \mathrm{NP}$.

Proof. Consider an instance of MAX-QE over GF(2) generated by arithmetizing a 3-CNF formula. Recall that in order to arithmetize a clause using quadratic equations, we introduce a variable z_j for clause j and use equations $z_j = 1 - \tilde{x}_{j_1}\tilde{x}_{j_2}$ and $(1 - z_j)\tilde{x}_{j_3} = 0$, where $\tilde{x}_{j_i} = (1 - x_{j_i})$ when the clause contains x_{j_i} and $\tilde{x}_{j_i} = x_{j_i}$ when the clause contains $\overline{x_{j_i}}$.

Notice that if some assignment a satisfies all equations, then a satisfies any linear combination of those equations. On the other hand, suppose that a doesn't satisfy all equations. Then a satisfies exactly one half of the linear combinations of the equations because a linear combination is satisfied by a if and only if it consists of an even number of equations that a doesn't satisfy.

One would be tempted to conclude that this observation by itself gives us the inapproximability result we want. However, the number of linear combinations is exponential in the input size, so the result doesn't follow yet. We use a small-bias PRG to complete the proof.

Pick a linear combination $E(c) = \sum_j c_j E_j$ of equations where $c_j \in \{0, 1\}$ and $c = (c_1, c_2, \ldots, c_m)$. Let B_a be the set of equations E_j that are not satisfied by an assignment a to the variables. Then a satisfies $E(c)$ if and only if $\chi_{B_a}(c) = 1$ because in that case the number of equations not satisfied by a that are present in the linear combination is even.

Notice that

$$\begin{aligned}
\mathrm{E}_{c \leftarrow U_n}[\chi_{B_a}(c)] &= \Pr_{c \leftarrow U_n}[\chi_{B_a}(c) = 1] - \Pr_{c \leftarrow U_n}[\chi_{B_a}(c) = -1] \\
&= 2 \Pr_{c \leftarrow U_n}[\chi_{B_a}(c) = 1] - 1 \\
&= 2 \Pr_{c \leftarrow U_n}[a \text{ satisfies } E(c)] - 1,
\end{aligned}$$

which means that

$$\Pr_{c \leftarrow U_n}[a \text{ satisfies } E(c)] = \frac{1}{2} + \frac{1}{2}\mathrm{E}_{c \leftarrow U_n}[\chi_{B_a}(c)]. \tag{18}$$

Now let's replace U_n by our small-bias PRG G_n. Since G_n is a small-bias PRG, $\mathrm{E}_{x \leftarrow G_n}[\chi_{B_a}(c)] \leq \epsilon$, which means that

$$\Pr_{c \leftarrow G_n}[a \text{ satisfies } E(c)] = \frac{1}{2} + \frac{1}{2}\mathrm{E}_{c \leftarrow G_n}[\chi_{B_a}(c)] \leq \frac{1}{2} + \frac{\epsilon}{2}. \tag{19}$$

Suppose that we could approximate MAX-QE within a factor $\frac{1}{2} + \epsilon$ or better using some algorithm $\mathcal{A}$. Then consider an arithmetization of a 3-SAT formula φ in m clauses and n variables. This yields $2m$ quadratic equations (two for each clause) in $n + m$ variables. Consider a small-bias PRG G_n with bias ϵ that generates strings of length $2m$. We need the seed length to be $2\log(2m/\epsilon)$. We then cycle through all random seeds of G_n, get a total of $2^{2\log(2m/\epsilon)} = (2m/\epsilon)^2$ strings of length $2m$, and thus a total of $(2m/\epsilon)^2$ linear combinations of the equations formed during arithmetization of φ. We can do this as long as $\epsilon > \frac{1}{\text{poly}}$. Now we run the algorithm $\mathcal{A}$. We accept if $\mathcal{A}$ satisfies at least $\frac{1}{2} + \epsilon$ of the linear combinations thus created, and reject otherwise.

If $\varphi \in$ 3-SAT, $\mathcal{A}$ will produce an assignment to the variables of our system of equations that satisfies at least $\frac{1}{2} + \epsilon > \frac{1}{2} + \frac{\epsilon}{2}$ of the linear combinations, so we accept correctly by (19). On the other hand, if $\varphi \notin$ 3-SAT, (19) implies that $\mathcal{A}$ will produce an assignment that satisfies a fraction of less than $\frac{1}{2} + \epsilon$ of the linear combinations. Since $\mathcal{A}$ gives a $(\frac{1}{2} + \epsilon)$-approximation to MAX-QE, it's not possible that all linear combinations are satisfiable, so we reject correctly as well. This means that we can use $\mathcal{A}$ to decide satisfiability in polynomial time, which completes the proof. □

5 Closing Remarks

Sometimes it is sufficient that small-bias PRGs have small bias only for small nonempty sets. For such PRGs, we can do with even shorter seed lengths.

Next time we start discussing threshold phenomena.