CS 880: Pseudorandomness and Derandomization 2/4/2013

Lecture 4: k-Wise Uniform Generators

Instructors: Holger Dell and Dieter van Melkebeek Scribe: Alexi Brooks

In this lecture we develop our first nontrivial pseudorandom generator (PRG). Recall that a PRG for a class $\mathcal{A}$ is a collection of mappings $G = (G_r)_{r\in\mathbb{N}}$ where $G_r : \{0,1\}^{\ell(r)} \to \{0,1\}^r$ such that for every algorithm $A \in \mathcal{A}$ and almost every input x on which A uses, say, r random bits, the behavior of $A(x,\rho)$ is about the same when ρ is picked from the uniform distribution U_r and when it is picked from $G_r(U_{\ell(r)})$. For a given class $\mathcal{A}$, we create a PRG by exploiting the fact that algorithms in $\mathcal{A}$ only use their randomness in a certain limited way. Another way of putting this is to say that the "good behavior" of the algorithms in $\mathcal{A}$ only relies on certain properties of the distribution of random bits ρ. In this lecture, the property we consider is k-wise uniformity. For an arbitrary alphabet Σ, the notion is defined as follows.

Definition 1 (k-Wise Uniformity). *A distribution D on Σ^r is said to be k-wise uniform over Σ if, for all choices of k distinct positions $1 \le i_1 < i_2 < \ldots < i_k \le r$, the distribution $D|_{i_1,\ldots,i_k}$ is uniform on Σ^k.*

The notation $D|_{i_1,\ldots,i_k}$ denotes the marginal distribution of D on the positions $i_1,\ldots,i_k$, that is, the distribution obtained by picking a sample $\rho \doteq \rho_1 \ldots \rho_r$ from D and outputting $\rho_{i_1} \ldots \rho_{i_k}$.

We are mostly interested in the binary alphabet $\Sigma = \{0,1\}$ and will develop perfectly k-wise uniform bit distributions that can be generated from $\ell(r) = O(k \log r)$ truly random bits. In terms of the parameters of the PRG, this means that we achieve logarithmic seed length and zero error, which is unusually good. The zero error means that algorithms that only require k-wise uniformity of their random bit sequence behave *exactly* the same on the uniform distribution and on the pseudorandom one.

We will first motivate the special case of pairwise uniform bits ($k = 2$ and $\Sigma = \{0,1\}$), describe a simple construction, and observe a connection with error correcting codes (ECCs) that holds in the general case. We will then use that connection to construct perfect k-wise uniform generators over $\Sigma = \mathbb{F}_q$ that produce r elements from a seed of $O(k \cdot \log_q(r))$ elements, which in particular yield the promised k-wise uniform bit generators with seed length $O(k \log r)$.

1 Pairwise Uniformity

For motivation, we start with an application of pairwise uniformity: a randomized algorithm for maximum cut whose "good behavior" only depends on the pairwise uniformity of the random source's underlying distribution.

1.1 Application: Maximum Cut

We are given an undirected graph $G = (V,E)$. Our goal is to find a partition (S,T) of V such that the set $E(S,T)$ of edges that cross between S and T is as large as possible. That is, we are looking for sets $S,T \subseteq V$ with $S \cap T = \emptyset$ and $S \cup T = V$ such that the number $|E(S,T)|$ of edges that have one endpoint in S and one endpoint in T is maximized.

The corresponding Minimum Cut problem can be solved in polynomial time without any application of randomness, for example, by using linear programming. Maximum Cut, on the other hand, is known to be NP-hard [PT95]. The best polynomial-time approximation algorithm currently known yields an approximation factor $\alpha \doteq \frac{2}{\pi} \min_{0<\theta<\pi} \frac{\theta}{1-\cos\theta} \approx 0.878$, and it uses semidefinite programming [GW95]. It turns out that achieving a constant approximation factor larger than α is "unique games hard", which means that solving so-called unique games reduces to this approximation problem [KKMO07]. Whether solving unique games is NP-hard remains open; a positive answer is known as the unique games conjecture.

In the following, we present two algorithms that yield a cut $E(S,T)$ such that

$$|E(S,T)| \geq |E|/2\,, \tag{1}$$

and therefore guarantee an approximation factor of $1/2$.

The first algorithm is a simple sequential algorithm that follows the greedy paradigm.

1. Start with $S = T = \emptyset$.
2. Go over all vertices in an arbitrary order, say in lexicographical order.
3. If the vertex v under consideration has more neighbors in the current set S than in the current set T, then add v to T; otherwise, add v to S.

The algorithm maintains the following invariant: The current cut $E(S,T)$ contains at least half of all edges of the subgraph $G[S \cup T]$ that is induced by the vertices considered so far. The invariant holds initially because the initial subgraph is empty. Whenever we consider a vertex v in a subsequent step, we decide whether to put v in S or in T. This decision is made in such a way that at least half of the edges between v and all previously considered vertices are cut. At the end of the algorithm, the invariant guarantees (1).

The greedy algorithm above is very efficient – when given the graph G in adjacency list representation, it can be implemented in time $O(|V|+|E|)$. It cannot, however, be run in parallel. Each choice made depends on every choice made before it. We now present a randomized algorithm that overcomes this deficiency by making all of its decisions in parallel. In expectation, the algorithm achieves the same quality guarantee, that is,

$$\mathbb{E}\Big[|E(S,T)|\Big] \geq |E|/2\,. \tag{2}$$

In this expression, S and T are random variables that represent the output of the algorithm; they depend on the internal randomness of the algorithm. For each $v \in V$, let X_v be a random bit, i.e., a sample from U_1. We define S and T as follows:

$$\begin{aligned} S &\doteq \big\{v \in V \,:\, X_v = 0\big\} \text{ and} \\ T &\doteq \big\{v \in V \,:\, X_v = 1\big\}\,. \end{aligned}$$

In essence, the algorithm assign vertices to S and T completely at random. Any given edge $(u,v) \in E$ gets cut if and only if $X_u \neq X_v$. If $X_u = b$, there is a precisely 50% chance that $X_v = b$, and a corresponding 50% chance that $X_v \neq b$. In other words, there is a 50% chance that the edge (u,v) is cut. Since this holds for every edge, (2) follows as we will see below.

We make use of indicator variables and the linearity of expectation. For any event defined by a predicate P, let $I[P]$ denote the random variable that takes the value 1 if the event happens (P holds), and 0 otherwise. We can write

$$|E(S,T)| = \sum_{e=(u,v)\in E} I[X_u \neq X_v].$$

This is just a different way of expressing that the cut size is equal to the number of edges that cross between S and T. By linearity of expectation, we then have

$$\mathbb{E}\Big[|E(S,T)|\Big] = \sum_{e=(u,v)\in E} \mathbb{E}\Big[I[X_u \neq X_v]\Big].$$

The expectation of an indicator variable equals the probability of the underlying event, so $\mathbb{E}\big[I[X_u \neq X_v]\big] = \Pr[X_u \neq X_v]$. Moreover, by the above argument,

$$\Pr[X_u \neq X_v] = \frac{1}{2}\,. \tag{3}$$

It follows that (2) holds.

Note that the predicate $X_u \neq X_v$ only depends on two of the $n \doteq |V|$ random variables. Thus, if we pick $(X_v)_{v\in V}$ from a distribution on $\{0,1\}^n$ that is only pairwise uniform (and in particular, not necessarily U_n), then (3) still holds and the rest of the analysis also carries through. Thus, the conclusion (2) holds as long as the distribution of $(X_v)_{v\in V}$ over $\{0,1\}^n$ is pairwise uniform. Potential correlations between the variables X_v do not affect the expected cut size as long as the variables are pairwise independent and individually uniformly distributed.[1]

Now, if (2) holds, then there has to be at least one $(x_v)_{v\in V}$ in the support of the distribution $(X_v)_{v\in V}$ for which the resulting partition (S,T) satisfies $|E(S,T)| \geq |E|/2$. As we will see in the next section, there exists a pairwise uniform distribution on $\{0,1\}^n$ that can be generated efficiently (in parallel) from a seed of $O(\log n)$ bits. Trying all possible seeds, computing the resulting cut size $|E(S,T)|$, and selecting one that gives the largest cut size, can then all be done efficiently in parallel, which results in an efficient parallel deterministic algorithm for finding a cut of size at least $|E|/2$.

1.2 Construction

We now describe the *inner product generator* [LLW05], a simple construction of a pairwise independent bit generator $G_r : \{0,1\}^\ell \to \{0,1\}^r$ with $\ell = O(\log r)$. Given a seed $\sigma \in \{0,1\}^\ell$, we output the XOR of all nonempty subsets of the positions $[\ell] = \{1,\dots,\ell\}$:

$$G_r(\sigma) = \left(\bigoplus_{i:x_i=1} \sigma_i\right)_{x\in\{0,1\}^\ell\setminus\{0^\ell\}}, \tag{4}$$

where x is the index of the output bits and is interpreted as the characteristic vector of a nonempty subset of $[\ell]$ (and thus ranges over $\{0,1\}^\ell \setminus \{0^\ell\}$). This construction gives us $r = 2^\ell - 1$, where the -1 comes from removing the empty subset ($x = 0^\ell$) from consideration. An equivalent way of expressing (4) is in terms of the inner product: $G_r(\sigma) = (\langle x,\sigma\rangle)_{x\neq 0^t}$, where $\langle x,\sigma\rangle$ is the inner product of x and σ over $\mathbb{F}_2$.

[1] In fact, the distribution of the first variable does not matter.

Theorem 1. *The distribution $G_r(U_\ell)$, where $G_r : \{0,1\}^\ell \to \{0,1\}^r$ is defined by (4), is pairwise uniform over $\{0,1\}$.*

Proof. We want to show that, for any two distinct positions $x_1, x_2 \in \{0,1\}^\ell \setminus \{0^\ell\}$, the joint output distribution at those two positions is uniform on $\{0,1\}^2$ when σ is picked from the uniform distribution on $\{0,1\}^\ell$. That is, for every choice of $y_1, y_2 \in \{0,1\}$, we want to prove that

$$\Pr_{\sigma \leftarrow U_\ell} \left[\big(G_r(\sigma)\big)_{x_1} = y_1 \ \wedge \ \big(G_r(\sigma)\big)_{x_2} = y_2 \right] = \frac{1}{4} \,. \tag{5}$$

Given the way we construct G, we can write the predicate underlying (5) as

$$\begin{cases} \sum_{j=1}^{\ell} (x_1)_j \cdot \sigma_j &= y_1 \text{ and} \\ \sum_{j=1}^{\ell} (x_2)_j \cdot \sigma_j &= y_2 \,, \end{cases} \tag{6}$$

which we can view as a system of two linear equations over $\mathbb{F}_2$ in the ℓ variables $\sigma_1, \ldots, \sigma_\ell$, where the first equation has the coefficient vector x_1 and the second one x_2. Since neither x_1 nor x_2 are the zero vector, and they are distinct, the system has full rank over $\mathbb{F}_2$, which implies that the number of solutions is the same no matter what the right-hand sides y_1, y_2 are, namely $2^{\ell-2}$. Since we pick σ uniformly from $\{0,1\}^\ell$, the probability that it is one of the solutions is $2^{\ell-2}/2^\ell = 1/4$. This establishes (5). □

Note that we had to rule out $x = 0^\ell$ because $\langle 0^\ell, U_\ell \rangle$ is constant zero and thus not uniform. Alternately, we could allow the all-zero index but only use the first $\ell - 1$ bits as the index and XOR all components with the last seed bit σ_ℓ. This may be somewhat more elegant but only achieves $r = 2^{\ell-1}$.

The construction (4) generalizes to any alphabet Σ that forms a group, which yields a pairwise uniform generator $G_r : \Sigma^m \to \Sigma^r$ over Σ with $r = 2^m - 1$. This is because the reasoning about the system (6) of linear equations with coefficients in $\{0,1\}$ holds over any group. In particular, it works if we pick $\Sigma = \mathbb{F}_q$. For the special case of $\mathbb{F}_q$ with $q = 2^p$ for some positive integer p, we can view the elements of Σ as binary strings of length p, and obtain a pairwise uniform generator $G_r : \{0,1\}^{pm} \to \Sigma^r$ over Σ with $r = 2^m - 1$. In this case, we can interpret the seed as a $(p \times m)$-matrix T over $\mathbb{F}_2$ whose columns correspond to the representations of $\sigma_1, \ldots, \sigma_m$ as elements of $\mathbb{F}_2^p$; the positions x are non-zero vectors of length m over $\mathbb{F}_2$ and the xth position of G_r becomes the matrix-vector product Tx. This is because addition over $\mathbb{F}_q$ corresponds precisely to addition over $\mathbb{F}_2^p$. In the alternate construction where we allow the zero vector x but use an additional random element $b \in \Sigma$ as part of the seed, the xth position of G_r is given by $Tx + b$. The latter construction has seed length $p(m+1) = O(pm)$. We can improve the seed length to $2p + m - 1 = O(p + m)$ by imposing additional structure on T, namely that the entries are constant on the descending diagonals, that is, the value of the entries only depends on the difference of their row and column indices. Such matrices T are called Toeplitz matrices.

Exercise 1. *Show that for any positive integers m and p, the following distribution is pairwise uniform over $\{0,1\}^p$: Pick a Toeplitz matrix $T \in \{0,1\}^{p \times m}$ and a vector $b \in \{0,1\}^p$ uniformly at random, and output $\big(Tx + b\big)_{x \in \{0,1\}^m}$.*

2 Connection with Error Correcting Codes

Let us return to our original simple construction of a pairwise uniform bit generator $G_r : \{0,1\}^\ell \to \{0,1\}^r$ where $r = 2^\ell - 1$ and $G_r(\sigma) = (\langle x, \sigma \rangle)_x$ with x ranging over $\{0,1\}^\ell \setminus \{0^\ell\}$. Note that G_r is a linear transformation, so we can write it as $G_r(\sigma) = M\sigma$, where M is a $(r \times \ell)$-matrix over $\mathbb{F}_2$. This means that we can interpret G_r as a linear code (provided M has full rank). In fact, M is the generator matrix of the punctured Hadamard code, so G_r is the punctured Hadamard code. Can we relate the requirement of pairwise uniformity to a property of the code?

The answer is yes. For G_r to be pairwise uniform, it is necessary and sufficient for any two rows of M to be linearly independent. We argued the sufficiency in the proof of Theorem 1. The necessity follows because two linearly dependent rows either contain at least one which is zero (in which case the corresponding component is constant), or else the two rows are identical (in which case the two components are always equal); in either case, the joint distribution of the corresponding components is not uniform. We saw something like this when we discussed the concept of "distance" for linear codes in Lecture 3. The distance of a linear binary code is equal to the minimum number of columns that are required to be picked in the parity check matrix to get a sum that is all zeros. Thus, G_r is pairwise uniform if and only if the distance of the binary code with parity check matrix M^T is at least 3. Since the latter code is the dual code of G_r, we have the following:

Proposition 2. *A linear binary code is a pairwise uniform bit generator if and only if the distance of its dual code is at least 3.*

In our case, the dual of the punctured Hadamard code is the Hamming code, which has distance exactly 3. This is another way to see why our simple construction yields a pairwise uniform generator.

Proposition 2 generalizes to k-wise uniformity for larger values of k and to alphabets $\Sigma = \mathbb{F}_q$ for q other than 2.

Proposition 3. *A linear code over $\mathbb{F}_q$ is a k-wise uniform generator over $\mathbb{F}_q$ if and only if the distance of its dual code is at least $k+1$.*

Proof. Let $\Sigma = \mathbb{F}_q$ and let $G_r : \Sigma^\ell \to \Sigma^r$ with $G_r(x) = Mx$, where M is an $(r \times \ell)$-matrix over $\mathbb{F}_q$. Let U denote the uniform distribution over Σ^ℓ. We prove that the following are equivalent:

(1) $G_r(U)$ is k-wise uniform over $\mathbb{F}_q$.

(2) Every k rows of M are linearly independent over $\mathbb{F}_q$.

(3) The code with parity-check matrix M^T has distance at least $k+1$.

(4) The dual code of G_r has distance at least $k+1$.

(1) $\iff$ (2): This follows from the argument in the proof of Theorem 1. Under the uniform distribution of the seeds, the output distribution at a given set $x_1, x_2, \ldots, x_k$ of k distinct positions is uniform over $\mathbb{F}_q$ if and only if the number of solutions to the system of k linear equations in the ℓ seed variables over $\mathbb{F}_q$ corresponding to (6) is independent of the right-hand side. The latter is the case if and only if the rows of the coefficient matrix are linearly independent. Since the coefficient matrix consists of the rows of M labeled by $x_1, x_2, \ldots, x_k$, the equivalence follows.

(2) $\iff$ (3): This is Proposition 3 from Lecture 3.

(3) $\iff$ (4): This is true because the code with parity check matrix M^T is the dual code of G_r.$\square$

3 k-Wise Uniformity

We now exploit the connection between k-wise uniformity and error correcting codes to construct a linear k-wise uniform generator $G_r : \Sigma^\ell \to \Sigma^r$ over $\Sigma = \mathbb{F}_q$ that has a short seed length ℓ.

By Proposition 3, this is equivalent to finding an $[r, r-\ell, k+1]_q$-code with small ℓ, i.e., we want to find a linear code over $\mathbb{F}_q$ with prescribed block length r and minimum distance $k+1$, and with as large a message length as possible. Reed–Solomon codes realize an optimal relationship between block length, message length, and distance, which for the prescribed block length and minimum distance give us a message length of $r+1-(k+1) = r-k$, and therefore yield $\ell = k$. However, they only exist for a block length that is equal to the alphabet size, i.e., for $r = q$. Thus, we can use the $[q, q-k]_q$ Reed–Solomon code. The actual k-wise uniform generator is the dual of this code, which, as we saw in Lecture 3, is the $[q, k]_q$ Reed–Solomon code. Thus, we conclude that the $[q, k]_q$ Reed–Solomon code is a k-wise uniform generator G_q over $\mathbb{F}_q$.

Recall that the $[q, k]_q$ Reed–Solomon code takes a message of k elements from $\mathbb{F}_q$, interprets them as successive coefficients of a univariate polynomial over $\mathbb{F}_q$, and encodes them as the value of that polynomial at all elements of $\mathbb{F}_q$. Spelled out this way, we have established the following result.

Lemma 4. *For every prime power q and integer $k \le q$, let $G : \mathbb{F}_q^k \to \mathbb{F}_q^q$ be defined by*

$$G(\sigma) = \left(\sum_{i=1}^{k} \sigma_i x^{i-1} \right)_{x \in \mathbb{F}_q}.$$

Then the distribution obtained by applying G to a uniform sample from $\mathbb{F}_q^k$ is k-wise uniform over $\mathbb{F}_q$.

The reason why the construction in Lemma 4 works is actually quite intuitive. Consider any k distinct positions $x_1, x_2, \ldots, x_k$ in the output distribution. The elements we get to see at those positions are the values of a random polynomial of degree at most $k-1$ over $\mathbb{F}_q$ at the points $x_1, x_2, \ldots, x_k$, where each of the k coefficients are chosen uniformly at random from $\mathbb{F}_q$. Even if we are told what the values of the polynomial are at all but the last point x_k, we have no clue about the value of the polynomial at x_k; each possible value in $\mathbb{F}_q$ is equally likely. This is equivalent to saying that the joint distribution at all k points is uniform over $\mathbb{F}_q^k$. Note, though, that if we were looking at one more position, say x_{k+1}, then that component is completely determined once we have seen the components at positions $x_1, x_2, \ldots, x_k$. This is because the values at k points uniquely determine the interpolating polynomial of degree at most $k-1$. Thus, the construction from Lemma 4 yields a distribution that is k-wise but not $(k+1)$-wise uniform.

A few remarks are in order. The generator uses a seed consisting of k elements of $\mathbb{F}_q$ and has output length $r = q$. What if we need an output length larger than q? We could take $\frac{r}{q}$ independent samples from G_q, but this naïve construction requires a seed consisting of $\frac{r}{q}k$ elements of $\mathbb{F}_q$. A better solution is to apply the construction from Lemma 4 over an extension field $\mathbb{F}_{q^p}$ for an appropriate choice of p. Viewing elements of $\mathbb{F}_{q^p}$ as sequences of p elements of $\mathbb{F}_q$, this construction yields a k-wise uniform generator[2] over $\mathbb{F}_q$ that takes a seed of pk elements and outputs (at most) pq^p elements. By setting $p = \log_q(r)$, the seed only consists of $\log_q(r)k$ elements of $\mathbb{F}_q$.

[2] In fact, the PRG has the following stronger pseudorandomness property: If we consider the natural partition of the output into blocks of p positions each, then the distribution is k-wise uniform over those blacks.

Theorem 5. *For every prime power q and positive integers r and $k \leq r$, there exists a polynomial-time computable k-wise uniform generator $G : \mathbb{F}_q^\ell \to \mathbb{F}_q^r$ with $\ell = k \cdot \lceil \log_q(r) \rceil$.*

In particular, for $q = 2$, the construction over the extension field yields a k-wise uniform bit generator with seed length $k \log(r)$, which is significantly better than the seed length $\frac{r}{2}k$ of the naïve construction.

References

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[KKMO07] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

[LLW05] Michael George Luby, Michael Luby, and Avi Wigderson. *Pairwise independence and derandomization*, volume 1 of *Foundations and Trends in Theoretical Computer Science*. 2005. `http://www.math.ias.edu/~avi/BOOKS/TCS003-1.ps`.

[PT95] S. Poljak and Z. Tuza. Maximum cuts and large bipartite subgraphs. *DIMACS Series*, 20:181–244, 1995.