**CS59200-PRS: Pseudorandomness** Aug 27, 2025

# Lecture 2: Pairwise and $k$-wise Independence

*Lecturer: Wei Zhan* *Scribe: Justin Zhang*

## 1 Recap

Last lecture, we introduced the definition of pseudorandomness.

**Definition 1** ($\varepsilon$-fooling)**.** *Let $\mathcal{D}$ be a target distribution and $\mathcal{S}$ be the source distribution. We say function $G$ is pseudorandom against a set $\mathcal{A} = \{A : \operatorname{supp}(\mathcal{D}) \to \{0,1\}\}$ if there exists $\varepsilon > 0$ such that for all $A \in \mathcal{A}$,*

$$\left| \mathop{\mathbb{E}}_{s \sim \mathcal{S}}[A(G(s))] - \mathop{\mathbb{E}}_{\gamma \sim \mathcal{D}}[A(G(\gamma))] \right| \leq \varepsilon.$$

*Then, we say that function $G$ $\varepsilon$-fools the adversaries $\mathcal{A}$.*

In particular, when the target distribution $\mathcal{D}$ is the uniform distribution over $\{0,1\}^n$ and the source distribution $\mathcal{S}$ is the uniform distribution over $\{0,1\}^\ell$, for $\ell := \ell(n) < n$, we say function $G$ is a *pseudorandom generator* (PRG) against functions/adversaries $\mathcal{A}$. To summarize the bounds found on the minimum seed-length found in last lecture:

- when there is one adversary, i.e., $|\mathcal{A}| = 1$, the seed length is $\ell \geq \lceil \log(1/\varepsilon) \rceil - 1$;
- when the adversary class is all boolean functions, $\mathcal{A} = \{A : \{0,1\}^n \to \{0,1\}\}$, the seed length is necessarily at least the target length, $\ell = n$.
- For a generic, unknown set of adversaries $\mathcal{A}$, there exists a random construction with seed length $\ell = \log\log(|\mathcal{A}|) + 2\log(1/\varepsilon) + O(1)$ that $\varepsilon$-fools every $A \in \mathcal{A}$. No explicit constructions are known.
- If we consider only poly-time adversaries $\mathcal{A} = \mathsf{P}$, with $\varepsilon = \operatorname{negl}(n)$, then $\ell = \omega(\log n)$.

Remark: the random construction bound for generic adversaries can almost be guesses from the first two bounds. The intuition is that the unknown set of adversaries 'falls somewhere between' one adversary and all adversaries, so a PRG fooling such a set must have seed length lower bounded by both cases. Hence, the first bound contributes the $\log(1/\varepsilon)$ factor, and under the observation that there are $2^{2^n}$ boolean functions, the second bound contributes a $\log\log(|\mathcal{A}|) \leq n$ factor.

# 2 MAX-CUT

At the end of last lecture, we introduced the random MAX-CUT algorithm that approximated to $1/2$ of the max cut. Essentially, we take a random labeling $\ell : V \to \{0, 1\}$. By (linearity of) expectation, the cut is at least size $|E|/2 \geq \text{MAX-CUT}/2$ :

$$\mathbb{E}[|C|] = \sum_{(i,j)\in E} \mathbb{E}[1_{\ell(i)\neq\ell(j)}] = |E|/2$$

The problem is that there are $2^n$ possible cuts, so we want a pseudorandom labeling with seed length $O(\log n)$ fooling the algorithm $A(G = (V, E), r)$ defined as the function computing the fraction of the cut i.e.,

$$A(G, r) := \frac{1}{|E|} \sum_{(i,j)\in E} 1_{\ell(i)\neq\ell(j)},$$

which we want to fool with some error $\varepsilon$. In particular, for $\varepsilon = 1/n^2$, since the number of edges in the cut is an integer and there are at most $\binom{n}{2}$ edges, this would imply that by going over the all $2^{O(\log n)} = \text{poly}(n)$ possibilities of the pseudorandom labeling, we definitely would found one cut that has size at least $|E|/2$.

We can interpret the family of adversaries as a single fixed adversary parameterized for a particular graph $G$, and we can immediately use the single-adversary bound. However, recall that the function was constructed by knowing the truth-table of $A$ (here the value table of $A$ since $A$ is real-valued), and it is not efficiently computable. If instead, we interpret the family of adversaries as an adversary per all graphs $G$, we can similarly use the prior bound with $|\mathcal{A}| = 2^{\binom{n}{2}}$. However, we will run into a similar problem of efficient explicit computation as the function was randomly constructed.

## 2.1 Pairwise Independence Gives An Explicit Solution

The crucial intuition is that the function computing the cut essentially only checks edge relations, so the randomness we use only needs to be 'local' to the edges. Specifically, we reduce the randomness we use by relaxing the use of independently random bits to *pairwise independent* random bits.

**Definition 2** (Pairwise Independence/Uniform)**.** *Random variables $X_1, \ldots, X_n$ are pairwise independent if each distinct pair $X_i$ and $X_j$ ($i \neq j \in [n]$), $X_i$ and $X_j$ are independent. Furthermore, if each $X_i$ is a uniformly random variable, then we say $X_1, \ldots, X_n$ are pairwise uniform.*

For intuition, we see how pairwise independence helps reduce randomness in a small example graph.

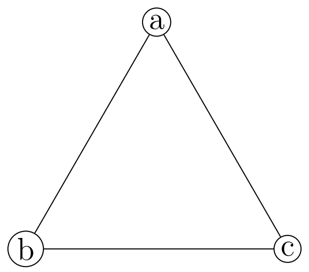

Figure 1: The complete graph on 3 vertices. A random labeling produces 3 random labels $a, b, c \in \{0,1\}$. We can reduce the amount of randomness used to 2 with the labels $a, b, a \oplus b$, whose corresponding random variables are pairwise independent.

*Example* (MAX-CUT on 3 Vertices). Consider the complete graph on 3 vertices pictured in Figure 1. By the random labeling $a, b, c \in \{0,1\}$, the expected number of cut edges is

$$\begin{aligned}\mathbb{E}[|C|] &= \sum_{(i,j)\in E} \mathbb{E}[1_{\ell(i)\neq\ell(j)}] \\ &= \Pr[a \neq b] + \Pr[b \neq c] + \Pr[c \neq a] \\ &= 3 \times (1/2).\end{aligned}$$

If instead, we used only drew two bits of randomness $a, b$ and used the labeling $a, b, a \oplus b$, observe that the expected value is still the same! This is because $a, b, a \oplus b$ is pairwise independent, so $\Pr[a \neq a \oplus b] = \Pr[b \neq a \oplus b] = 1/2$.

The same idea works for general graphs $G$, so we turn our attention to constructing pairwise uniform random variables.

## 3 Constructing Pairwise Uniform Random Variables

### 3.1 Pairwise Uniform Bits

To construct $n$ pairwise uniform bits, we use the following procedure: on a random seed $s \in \{0,1\}^{\log n}$, we think of $s$ as a vector in the vector space $\mathbb{F}_2^{\log n}$. We also identify the indices $i \in [n] = \{0, 1, \ldots, n-1\}$ with the vectors in $\mathbb{F}_2^{\log n}$ by binary expansion. Now we define the pairwise uniform bits be

$$X_i(s) := \langle s, i \rangle \bmod 2$$

for all $i \neq 0$, where $\langle \cdot, \cdot \rangle$ is the natural inner product. An interpretation of $X_i(s)$ is that we calculate the parity of $s$ at the indices indicated by the binary representation of $i$.

*Claim* 1. $X_1(s), \ldots, X_{n-1}(s)$ are pairwise uniform.

*Proof.* By construction, each of $X_1(s), \dots, X_{n-1}(s)$ is uniform over $\mathbb{F}_2$. We will show that for any $i \neq j, \Pr[X_i(s) = X_j(s)] = 1/2$, and apply uniformity to imply pairwise independence.

$$\begin{aligned}\Pr[X_i(s) = X_j(s)] &= \Pr[\langle s, i\rangle = \langle s, j\rangle] \\ &= \Pr[\langle s, i - j\rangle = 0] && \text{(bilinearity)} \\ &= 1/2.\end{aligned}$$

This, along with $\Pr[X_i(s) = 1] = \Pr[X_j(s) = 1] = 1/2$, is enough to solve the joint distribution of $X_i(s)$ and $X_j(s)$, and the unique solution is $\Pr[X_i(s) = b_i, X_j(s) = b_j] = 1/4$. That means $X_i(s)$ and $X_j(s)$ are also independent. □

## 3.2 Pairwise Uniform Hash Functions

Now, suppose we want pairwise uniform random variables over a larger set, such as bit strings $X_1, \dots, X_n \in \{0,1\}^m$. We can further think about pairwise uniform hash functions (parameterized by the seed $s \in \{0,1\}^\ell$) $h_s : \{0,1\}^n \to \{0,1\}^m$, where for all $x \neq x'$, $h_s(x), h_s(x')$ are independent. Note that this is equivalent to a $2^n$ sized set of $m$-long bit strings that are pairwise independent.

A trivial construction uses seed length $mn$, when we just use $m$ independent copies of pairwise independent bits. We will show how to lower this multiplicative dependence on $m$ and $n$ to an additive one.

We try to simulate what we did for pairwise independent bits, by working over a larger finite field $\mathbb{F}_{2^m}$, where the corresponding field arithmetic is known to take time poly$(m)$ time. Our first attempt is to define the hash functions $h_s : \{0,1\}^n \to \{0,1\}^m$ as

$$h_s(x) := \langle s, x\rangle,$$

where $s, x \in \mathbb{F}_{2^m}^k$, with $k = \lceil n/m \rceil$. While uniformity holds by construction, this is not pairwise independent. For example, for every non-unit element $a \in \mathbb{F}_{2^m}$, $h_s(x)$ and $h_s(ax)$ are no longer independent since $h_s(ax) = a \cdot h_s(x)$. We resolve this by adding another term, defining $h_{s_0,s_1}(x) = \langle s_1, x\rangle + s_0$. Here $s_1 \in \mathbb{F}_{2^m}^k$ while $s_0 \in \mathbb{F}_{2^m}$.

*Claim* 2. $h_{s_0,s_1}(x)$ is a pairwise uniform hash function with seed length $km + m \leq n + 2m$.

*Proof.* By construction uniformity holds (specifically because $s_0$ is uniformly random in $\mathbb{F}_{2^m}$). We show pairwise independence:

$$\begin{aligned}\Pr[h_{s_0,s_1}(x) = y \wedge h_{s_0,s_1}(x') = y'] &= \Pr[\langle s_1, x\rangle + s_0 = y \wedge \langle s_1, x'\rangle + s_0 = y'] \\ &= \Pr[\langle s_1, x - x'\rangle = y - y' \wedge \langle s_1, x\rangle + s_0 = y] \\ &= \Pr[\langle s_1, x - x'\rangle = y - y'] \Pr[\langle s_1, x\rangle + s_0 = y \mid s_1] \\ &= (1/2^m) \times (1/2^m).\end{aligned}$$

The first term is $1/2^m$ because $x - x' \neq 0$, and on a chosen non-zero coordinate of $x - x'$, there is only one solution for $s_1$ after fixing all the other coordinates. The second term is because of the uniformity of $s_0$. □

## 4 Constructing $k$-wise Independent Hash Functions

We generalize pairwise independence to $k$-wise independence.

**Definition 3** ($k$-wise Independence)**.** *Random variables $X_1, \dots, X_n$ are $k$-wise independent if for all $1 \le i_1 < \cdots < i_k \le n$, $X_{i_1}, \dots, X_{i_k}$ are independent.*

Likewise, we can generalize the pairwise uniform hash function construction to a $k$-wise uniform hash function when the input and output has the same length. We replace the linear function $s_1 x + s_0$ with a degree $(k-1)$ polynomial, where the seed determines the $k$ coefficients,

$$h_s(x) = \sum_{i=1}^{k-1} s_i x^i = s_{k-1}x^{k-1} + \cdots + s_1 x + s_0.$$

Here $x$ and each $s_i$ is in $\mathbb{F}_{2^n}$, so seed length is $kn$ for hash functions $h_s : \{0,1\}^n \to \{0,1\}^n$.

The proof of $k$-wise uniformity follows a similar argument to its pairwise uniform counterpart, as we want to uniquely solve the seed given the conditions in a joint distribution. More specifically, we can think of the equations as a linear system on $s_0, \dots, s_{k-1}$, and we want to show

$$\Pr\left[\bigwedge_{i=1,\dots,k} h_s(x_i) = y_i\right] = (1/2^n)^k.$$

To see this, we writing out the linear system,

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^{k-1} \\ 1 & x_2 & \dots & x_2^{k-1} \\ & & \ddots & \\ 1 & x_k & \dots & x_k^{k-1} \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{k-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix},$$

which has a unique solution when $x_i$'s are distinct and thus the $k \times k$ Vandermonde matrix is non-singular.

*Remark.* Currently, the simplest way we know to construct $k$-wise independent bit strings is by taking the first $m$ bits of $k$-wise independent hash function outputs.