

Pairwise Independence and Derandomization

Michael Luby & Avi Wigderson

Foundations and Trends in Theoretical Computer Science 2005

- Introduction to Pairwise Independence
- Generating Pairwise Independent Bits
- Introduction to Derandomization
- Brute Force Derandomization
- Derandomization via Pairwise Independence

Pairwise Independent Events

- Consider the set of events A_1, \dots, A_n
- The events A_1, \dots, A_n are mutually independent if $Pr[A_1 \cap A_2 \dots \cap A_n] = Pr[A_1]Pr[A_2] \dots Pr[A_n]$
- The events A_i and A_j are pairwise independent if $Pr[A_i \cap A_j] = Pr[A_i]Pr[A_j]$ for all i, j

Example of Pairwise Independent Events

- Consider the set $\Omega = \{abc, acb, aaa, bac, bca, bbb, cab, cba, ccc\}$ and probability of any of them being picked is $\frac{1}{9}$.
- Let event A_k denote that the k^{th} letter is a . Clearly $Pr[A_k] = \frac{1}{3}$
- $Pr[A_1 \cap A_2] = Pr[A_2 \cap A_3] = Pr[A_3 \cap A_1] = \frac{1}{9}$
- $Pr[A_1 \cap A_2 \cap A_3] = \frac{1}{9} \neq Pr[A_1]Pr[A_2]Pr[A_3]$
- Thus, the events pairwise independent but not mutually independent

Pairwise Independence is weaker than Mutual Independence

Generating Pairwise Independent Bits

- We know how to generate mutually independent bits (Coin Flips)
- Can we use mutually independent bits to generate pairwise independent bits?

Construction of n pairwise independent random bits with uniform distribution from a small sample space

- For given n , let $t = \log(n + 1)$ and let $I = \{1, \dots, t\}$.
 - Let random bits b_1, \dots, b_t be obtained by tosses of a fair coin.
 - Let J_1, \dots, J_n be pairwise distinct nonempty subsets of I .
 - For $i = 1, \dots, n$ let $r_i = \bigoplus_{j \in J_i} b_j$.
-
- Observe that there are $2^t - 1$ pairwise distinct nonempty subsets of I

Generating Pairwise Independent Bits

Lemma: Verification of the construction

- The random bits r_1, \dots, r_n obtained by the construction above are pairwise independent and each bit is uniformly distributed.

Proof

- Fix any nonempty set $J \subseteq \{1, \dots, t\}$ and let $r = \bigoplus_{j \in J} b_j$
- For any fixed index $s \in J$ we have, $r = \bigoplus_{j \in J} b_j = (\bigoplus_{j \in J \setminus \{s\}} b_j) \oplus b_s$
- If the b_j 's have already been determined for all $j \in J$, then depending on the value of b_s , the value of r will either agree with or will differ from $\bigoplus_{j \in J, j \neq s} b_j$
- But b_s attains its two possible values with probability $\frac{1}{2}$ each and the b_j are mutually independent, hence r will be uniformly distributed in $\{0, 1\}$.

Generating Pairwise Independent Bits

Proof(continued)

- Consider any two set $S, T (S \neq T) \subseteq [t]$. We have:

$$R_S = R_{S \cap T} \oplus R_{S \setminus T}$$

$$R_T = R_{S \cap T} \oplus R_{T \setminus S}$$

- $R_{S \setminus T}, R_{S \cap T}$ and $R_{T \setminus S}$ are independent as they depend on disjoint subsets of the b_i 's
- At least two of these subsets are nonempty (because $S \neq T$). This implies that (R_S, R_T) each take value in $\{0, 1\}$ with probability $\frac{1}{4}$.
- The random bits generated are pairwise independent bits

Brute Force Derandomization

- Brute Force Derandomization refers to simulating a randomized algorithm for every possible value of its random source.
- Consider a randomized algorithm that runs for $poly(n)$ steps on all inputs of size n .
- The algorithm might use r random bits
- Once the random bits are known, each execution can be seen as a deterministic algorithm
- There are 2^r possible combinations of random bits
- Algorithm: Execute randomized algorithm for each possible combination
- Time Complexity of deterministic algorithm $2^r poly(n)$ steps
- Done? No!

Brute Force Derandomization

- We want efficient deterministic algorithm
- $2^r \text{poly}(n)$ must be $O(n^c)$ for some constant c
- $\Rightarrow r = O(\log n)$
- Thus, when $r = O(\log n)$, then even brute force derandomization works well
- First transform a randomized algorithm into one that uses only few random bits, then apply brute force derandomization