Math 268 2012-10-30

Yao's next bit test

NW (Nisan-Wiederson) PRNG from "hard boolean" function

f function every circuit c of size $s(n)$ $f: \{0,1\}^n \to \{0,1\}$

$P_{\text{random input } x}(x = f(x)) \le \frac{1}{2} + \frac{1}{s(n)}$

Query: ~~How likely~~ If $s(n) \gg 0$, this (f) looks a lot like a fair coin flip algorithm. How likely is it that such an f exists.

We want a distribution S on strings $S = \langle s_1, \dots, s_k \rangle$ that is "random-looking" and k is small. We want to take a probabilistic algorithm $A(x, r)$ random input of length $\le T$, use elts of S for r, and still get random output, i.e.,

$$\left| \underset{i \in [k]}{\text{Prob}} (A(x, s_i)) - \underset{\substack{r \in \{0,1\}^n \\ \text{uniform}}}{\text{Prob}} (A(x,r)) \right| \le 1/T, \text{ for every}$$

T-step algorithm A and input x. Equivalently, for every circuit c, $|c| \le T$, $|\text{Prob}_i(c(s_i)) - \text{Prob}(c(r))| \le \frac{\varepsilon}{T}$. Thus, in time (time to construct S) + k poly (T), we can deterministically simulate A.

(The first such sets were due to Yao, with a cryptographic assumptions, largely for concerns of speed. For our purposes, however, derandomization can take longer.)

Notation given S, $T_{\varepsilon} := |\text{Prob}(c(s_i)) - \text{Prob}(c(r))| \le \varepsilon$.

Let $s \in S$, $s = s_1 \cdots s_T$. A next-bit circuit for S at position i is a circuit c which, given $s_1, \dots, s_{i-1}$, $c(s_1 \cdots s_{i-1})$ is a guess for s_i ~~with~~ $c(i) = \underset{s \in S}{\text{Prob}} (c(s_1, \dots, s_{i-1}) = s_i) - 1/2$. Yao showed that there is no successful next-bit test $\Leftrightarrow$ S is pseudorandom, using what is now called a hybrid argument.

Let $S = S_T$, $R \xleftarrow{} S_0$ uniform random

S_i: pick $s_1, \dots, s_i \in S$ (as in S), $r_{i+1}, \dots, r_T \in R$.

Suppose $c(S) - c(R) > \varepsilon$. Then

$$c(S_T) - c(S_{T-1}) + c(S_{T-1}) \cdots - c(S_0) > \varepsilon, \text{ so } \exists i \; c(S_i) - c(S_{i-1}) > \varepsilon/T.$$

We'll get a next-bit test for i! ①

S_{i-1} $s_1 \cdots s_{i-1} r_i \cdots r_T$

S_i $s_1 \cdots s_{i-1} s_i r_{i+1} \cdots r_T$

$c'(s_1 \cdots s_{i-1})$ Pick $r_i, \ldots, r_T \in_u \{0,1\}^n$. If $c(s_1 \cdots s_{i-1} r_i \cdots r_T) = 1$, output r_i. Otherwise output $1 - r_i$.

With prob $1/2$, set $r_i = s_i$. With prob $1/2$, set $r_i = 1 - s_i$.

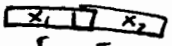
$\delta = c(\$ s_i)$ $\quad \delta' = c(s_1 \ldots s_{i-1} (1-s_i) r_{i+1} \cdots r_T)$

$c(s_{i+1}) = 1/2\,\delta + 1/2\,\delta' < \delta - \varepsilon/T$. So $\delta > \delta' + 2\varepsilon/T$.

Overall success $1/2\,\delta + 1/2 - 1/2\,\delta' \geq \frac{1}{2} + \varepsilon/T$

Computational independence of x_i's. I.e., we want it so that no matter what we know about $x_1, x_2, \ldots, x_{i-1}$ and $f(x_1), \ldots, f(x_{i-1})$ doesn't help predict $f(x_i)$ by more than $T' < T$.

Example $\quad 2^{\frac{T}{2}} - \log T'$ bits



Suppose $c(x_2, g(x_1)) = f(x_2)$, with probability δ.

① Fix the bits in $x_1 - x_2$ to maintain probability.

② Make a table of size T' of all possible $x_1, g(x_1)$ ⇐

③ Look up actual value of x_1 in table, $g(x_1)$, output $c(x_2, g(x_1))$

Now for the real version. Form of construction:

m: total bits to sample $z_1, \ldots, z_m$

For each i, $A_i \subseteq \{1, \ldots, m\}$, $|A_i| = n$, $A_i = \{a_{i1}, \ldots, a_{in}\}$, $x_i = z_{a_{i1}} \cdots z_{a_{in}}$. We insist for $i \neq j$ $|A_i \cap A_j| \leq \ell$ $(\log T)$, i.e., A_i's form an ℓ-design (although this is abuse of terminology)

Given $g_1(x_1), \ldots, g_{i-1}(x_{i-1})$, it is still hard to predict $f(x_i)$: if not, $c'(x_i, g_1(x_1), \ldots, g_{i-1}(x_{i-1}))$ predicts $f(x_i)$ $1/2 + \varepsilon$.

(contiguous for convenience)

① Fix the bits in $\overline{A_i}$ to maintain advantage. ~~For each~~

② For each x_j, $\leq 2^\ell$ possible values. Give a table of size 2^ℓ for each

③ C simulates C'. Look up each x_j in the table, Feed it to c'.

So we only need to worry about constructing A_i, and we want n small.

Let P be a prime, $2n \geq p \geq n$. Consider $\forall f(x) \in \mathbb{Z}_p[x]$. Set $m = pn$. Consider graph of $g(x)$, $\deg(g) = d \leq \ell$.

$A_g = \{(i, g(i)) : 1 \leq i \leq n\}$.

$A_g \cap A_{g'} = \{(i, g(i)) : (g - g')(i) = 0\}$ $\quad |A_g \cap A_{g'}| \leq \ell$.

$T \leq p^\ell$, so $\ell = \frac{\log T}{\log n}$.

Thm Say $\exists f \in EXP$; f_n is $s(n)$-hard for $s(n) = n^{\omega(1)}$. Then $\forall \varepsilon > 0$ $BPP \subseteq DTIME(2^{n^\varepsilon})$.

If say we want to simulate deterministically a probabilistic algorithm that takes time T. $n = T^\ell$, f_n is $T^{\omega(1)}$-hard.

We construct $A_1, \ldots, A_T$ with size of pairwise intersection $\leq \frac{\log(T)}{\log(n)} = \frac{1}{\delta}$.

$m = n^2 = T^{2\delta}$. For every $z_1, \ldots, z_n$, we compute $z = f(x_1) \cdots f(x_T)$.

$2^{n^\delta} \cdot T \cdot 2^m \leq$ subexponential in T