1 Design Construction for NW-PRG

Recall that last lecture we proved the following theorem with the assumption that some \((n, k)\)-design exists.

**Theorem 1.1.** Suppose \(L \in \text{DTIME}(T(n))\) is \((S, \epsilon)\)-hard, where \(L\) is a series of \((S, \epsilon)\)-hard functions for every input length. Then there exists \((S', \epsilon')\)-PRGs \(\{G_n : \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}\}_n\) computable in time \(m(n) \cdot T(n)\), where \(s' = s - O(m \cdot 2^k)\) and \(\epsilon' = m \cdot \epsilon\).

**Definition 1.2.** A \((n, k)\)-design in universe \([r]\) is a collection of sets \(S_1, S_2, \ldots, S_m \subseteq [r]\) where \(|S_i| = n\) and \(\forall i \neq j, |S_i \cap S_j| \leq k\).

We now show how to construct the design and how to set the parameters. First, fix field \(F = F_n\). For each polynomial \(P(x)\) on \(F\) of degree \(\leq k\), set \(S_p = \{(a, p(a)) : a \in F\}\).

**Claim 1.3.** For \(p \neq q\), \(|S_p \cap S_q| \leq k\)

The claim is proven by realizing that \(p - q\) has at most \(k\) roots, since they are polynomials of degree \(\leq k\). Therefore, \(p\) and \(q\) intersect at most \(k\) times. With this construction, \(r = n^2\) because both \(a\) and \(p(a)\) range from 1 to \(n\), and \(m = n^{k+1}\).

Next, fix small constant \(\delta > 0\), so that \(S = 2^\delta n\) and \(\epsilon = 2^{-\delta n}\). Can we take \(T(n) = \text{poly}(n)\)? In other words, is there a language \(L \in \text{DTIME}(\text{poly}(n))\) that is \((2^\delta n, 2^{-\delta n})\)-hard? We know that \(P \subset \text{P/poly}\), so there are no languages in \(P\) that cannot be approximated by an exponential sized circuit. Instead, we assume \(T(n) = 2^{O(n)}\), i.e. \(L \in \text{E}\).

Now, set \(k = \frac{n}{10 \log(n)}\). It follows that \(m = n^{k+1} = 2^{\frac{2n}{10 \log(n)}} = 2^{\frac{n}{10} \sqrt{n}}\).

To conclude, if we are given a \(L \in \text{E}\) that is \((2^\delta n, 2^{-\delta n})\)-hard, then we can produce a \((S', \epsilon')\)-PRG that takes in \(r = n^2\) bits and outputs exponential bits, where \(S' = 2^\delta n - 2^{\frac{n}{10} \sqrt{n}} = \Omega(2^\delta n)\) and \(\epsilon' = 2^{\frac{2n}{10} \sqrt{n}} \leq 2^{-\delta n}\). This implies any algorithm in \(\text{BPP}\) can be derandomized to a deterministic algorithm that runs in time \(n^{O(\log n)}\). We note that one can construct better designs that will yield \(\text{BPP} = \text{P}\) under the circuit lower bound assumptions against \(\text{E}\).

We now switch topics to a new type of proof system.

2 Interactive Proofs

2.1 \(\text{NP}\)

Now, let us revisit the setting of \(\text{NP}\) but through the lens of the verifier. Recall that \(\text{NP}\) is the set of decision problems verifiable by a deterministic Turing machine in polynomial time. An alternative way of viewing \(\text{NP}\) is as a simple interactive proof system, with a "prover" in addition to the verifier. On input \(x\), the all-powerful prover \(P\) comes up with some certificate for \(x, a_1\), and sends it to the verifier \(V\).

---

1
If $x \in L$, then some prover that can convince the verifier that $x$ is in $L$. If $x \notin L$, then no prover can convince the verifier that $x$ is in $L$. Formally, for $L \in \text{NP}$, there exists a verifier $V$ such that:

$$
\begin{align*}
\text{if } x \in L & \Rightarrow \exists P \text{ s.t. } V(x, a_1) = 1 \\
\text{if } x \notin L & \Rightarrow \forall P, V(x, a_1) = 0
\end{align*}
$$

2.2 Deterministic Interactive Proofs (dIP)

Now, let us define a proof system by extending the number of rounds of communication.

Definition 2.1. Define a complexity class $\text{dIP}[k(n)]$ where $n$ is the length of $x$, and $k(n)$ is the number of rounds of communication between the prover and verifier.

[Note that all messages $a_1, a_2, \ldots, a_{k(n)}$ between prover and verifier are polynomial in the size of $x$.] Let $\text{out}_V(V, P)(x)$ stand for the final output of the verifier. We now have: For $L \in \text{dIP}[k(n)]$, there exists a verifier $V$ such that:

$$
\begin{align*}
\text{if } x \in L & \Rightarrow \exists P \text{ s.t. } \text{out}_V(V, P)(x) = 1 \\
\text{if } x \notin L & \Rightarrow \forall P, \text{out}_V(V, P)(x) = 0
\end{align*}
$$

Definition 2.2. $\text{dIP} = \bigcup_{c \in \mathbb{N}} \text{dIP}[n^c]$

Claim 2.3. $\text{dIP} = \text{NP}$ (Nothing is gained by allowing communication.)

Proof:

$\text{NP} \subseteq \text{dIP}$ because $\text{NP} = \text{dIP}[1]$.

$\text{dIP} \subseteq \text{NP}$: Using $\text{dIP}$ verifier $V$, construct $\text{NP}$ verifier $\tilde{V}$: On input $x$ and certificate $a_1, a_2, a_3, \ldots, a_{k(n)}$, $\tilde{V}$ checks that all odd numbered messages $a_i$ are consistent with what $V$ would output. In other words, $\tilde{V}$ checks that $a_1 = V(x), a_3 = V(x, a_1, a_2), a_5 = V(x, a_1, a_2, a_3, a_4)$, and so on. If the odd numbered messages are consistent, then, $\tilde{V}$ outputs $V(x, a_1, a_2, a_3, \ldots, a_{k(n)})$; If the odd numbered messages are not consistent, then $\tilde{V}$ outputs 0. We can see that if $V(x) = 1$, then such a certificate consisting of the the messages would exist, and $\tilde{V}(x) = 1$.

So if deterministic interactive proofs don’t work, what will?
2.3 (Probabilistic) Interactive Proofs

We modify the deterministic interactive proof system defined above:

L is in $IP[k(n)]$ if $\exists V$ such that:

\[
x \in L \Rightarrow \exists P \, s.t. \, Pr[out_V(V,P)(x) = 1] > \frac{2}{3} \quad \text{(Completeness)}
\]

\[
x \notin L \Rightarrow \forall P, \, Pr[out_V(V,P)(x) = 1] < \frac{1}{3} \quad \text{(Soundness)}
\]

Note that the randomness lies with the verifier and not the prover because there is always a best strategy for the prover to take to trick the verifier.

We note that the following is a landmark theorem in complexity theory.

**Theorem 2.4.** $IP = PSPACE$.

Subject to time constraints, we may see some part of the proof of the above theorem in a future lecture. For now, let us investigate the power of $IP$.

**Definition 2.5.** $GNI = \{\langle G_0, G_1 \rangle : G_0 \not\approx G_1 \}$, where $G_0 \approx G_1$ if $\exists \pi : V \rightarrow V$ such that $\pi(G_0) = G_1$.

**Claim 2.6.** Graph Non-Isomorphism (GNI) $\in IP$

Algorithm:

On input of $G_0, G_1$, $V$ flips a bit $b$, picks a random permutation $\pi$, and sends $\pi(G_b)$ to $P$. $P$ then sends back $b'$, representing whether it thinks that $G_b \approx G_0$ or $G_b \approx G_1$. Then, $V$ accepts if $b = b'$.

To prove that this algorithm works, we consider its completeness and soundness. Completeness: First, suppose $G_0 \not\approx G_1$. Then, the prover can distinguish whether $G_b$ is a permutation of $G_0$ or a permutation of $G_1$ and send the correct $b'$ back. In this case, $b' = b$ with probability 1 and $V$ will always accept. Soundness: Now, suppose $G_0 \approx G_1$. In this case, the prover cannot distinguish whether $G_b$ is a permutation of $G_0$ or $G_1$ since it is isomorphic to both, and can only guess. Thus, it sends back 0 or 1 with probability 0.5 each, so $b' = b$ with probability 0.5. Although the 0.5 acceptance probability for isomorphic graphs falls below the $\frac{1}{3}$ bound, the algorithm can be repeated over and over again to drive the probability down. Thus, we have shown that $GNI \in IP$.

Note that we have assumed that $V$’s coins are unknown to $P$. This is called a private randomness setting. However, as we will see in the next lecture, this relaxing this assumption to a public randomness setting does not affect the proof.