CS 710: Complexity Theory Date: May 3rd, 2024 (revised on May 5th, 2024)

Term Project Report
Hardness vs Randomness: Review and Beyond

Instructor: Jin-Yi Cai Author: Hao Lin*

Contents

*Personal website: everhao.me

*Email: hao@cs.wisc.edu

1 Preamble

We have seen the elegance of the randomness through the MAXCUT problem, where the 1/2-approximation is simply a random assignment of the vertices; the undirected graph access problem (USTCONN) can be solved with randomized algorithm with arbitrarily small one-sided error in log space complexity [Ca03].

Not only a novel approach to solve problems with similar efficiency to deterministic approaches, it is usually more natural to come up with, which may be due to its elegance, whereas some deterministic algorithms are found by applying derandomization methods to the randomized one afterwards.

However, does every random algorithm have a derandomized counterpart where the time complexity does not vary significantly? Or say does BPP = P or BPP = EXP, or any in between? Unfortunately, we have so far only known that P $\subseteq$ BPP $\subseteq$ EXP, i.e. no proper containment is known on either side. Not every problem in BPP has found a deterministic polynomial algorithm (i.e. is also in P). It may take huge efforts to find a P machine even for a single problem, for instance, the primality test problem (PRIME), for which, Agrawal, Kayal, and Saxena discovered an unconditional deterministic polynomial-time algorithm in 2002 [AKS04], decades after the Miller-Rabin randomized primality test [Mi75; Ra80] is invented.

Understanding the power of randomness is of critical importance. Not only because it may provide insights on practical ways to invent efficient algorithms, but it may unveil the connections among other fields in computational complexity theory. From the paper in late 1980s, especially the well-known Hardness vs Randomness [NW94] written by Nisan and the recent Turing and Abel reward winner Wigderson, we learn some astonishing fundamental connections among efficiency of randomness, assumptions on the existence of hard problems, the circuit complexity, and even interactive proof system, polynomial hierarchy, and space-time trade-off. In this report, we will review this paper, deliver and prove some important results (may reorder the procedure), and also give some comments within and beyond it.

2 Overview of the Nisan-Wigderson PRG

The main claim, as suggested by their title, involves a trade-off between the existence of greater hardness and the power of randomness. One of the most impressing results leads to a collapse of randomness to determinism without performance penalty, up to the polynomial-time reduction. This gives great insights to the questions we have just concerned.

Theorem 1 *If there exists a language $L \in \mathrm{EXP}$, such that with hardness $2^{\varepsilon n}$ for some $\varepsilon > 0$, then* BPP = P.

We will soon define the hardness of a language, but it is related to the lower bound of the size of the circuit family.

The trade-off comes from similar claims that if the "weaker" the assumption of the lower bound, the smaller the gap we get for P and BPP; and vice versa.

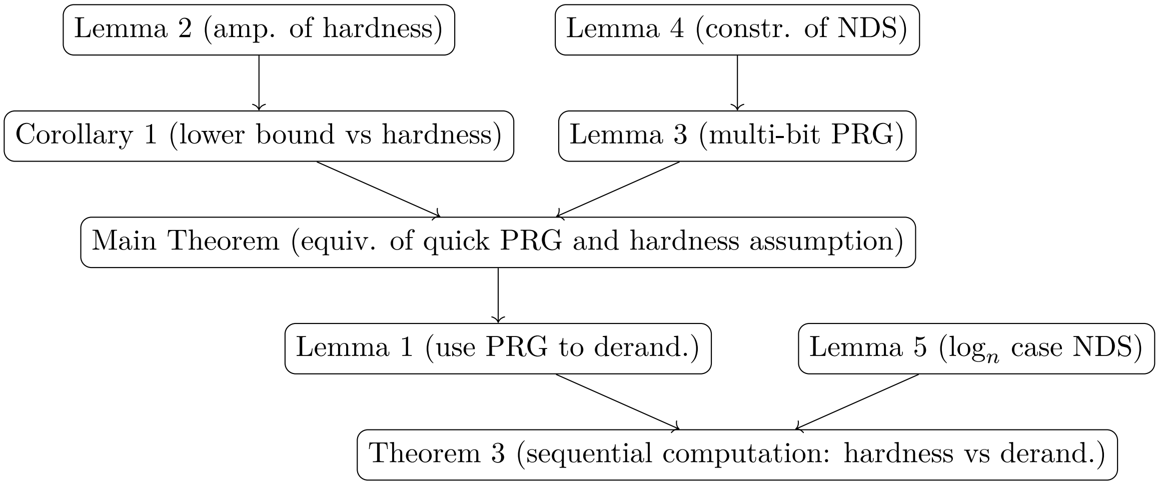
The main theorem in their paper puts the similar result in another way, probably in a technically more doable manner: unveiling the equivalence of circuit lower bound and efficiency of the pseudorandom generator (PRG).

The construction of a quick PRG based on hardness assumption gives a lot more results than the BPP = P collapse:

- Parallel extension to the equivalence between circuit lower bounds of PSPACE and space efficiency of random parallel computation
- Randomized constant depth circuit extension, whose lower bound is proved, gives the $RAC^0 \in$ poly-log -space unconditionally
- Random oracles gives probabilistic analogy to P, NP, and up to the whole PH: almost-P = BPP, almost-NP = AM, almost-PH = PH
- Efficient simulation of time by space is achieved if efficient derandomization fails; and vice versa

2.1 Proof Hierarchy of Main Results

We give the big picture of the proof for the main theorem and also Theorem 3.


Lemma 2 (amp. of hardness)
Lemma 4 (constr. of NDS)
Corollary 1 (lower bound vs hardness)
Lemma 3 (multi-bit PRG)
Main Theorem (equiv. of quick PRG and hardness assumption)
Lemma 1 (use PRG to derand.)
Lemma 5 ($\log_n$ case NDS)
Theorem 3 (sequential computation: hardness vs derand.)

3 Main Results: Generator and its Sequential Computations

The existence of true randomness is still a question. Even there exists true randomness, we still need to resort to pseudorandom generator for various reasons including reproducibility. Different from true generator, pseudorandom generator generates the same sequence if the input (or say the seed) is the same. However, it holds the property to fool some small circuits that pretends to be true generator, which is purely random, or say generates uniformly distributed sequences.

Definition 1 **(Pseudorandom Generator (PRG))** *$G = \{G_:\{0,1\}^{l(n)} \to \{0,1\}^n\}$, denoted by $G : l \to n$, is called a pseudorandom generator if for any circuit C of size n:*

$$\Big|\Pr[C(y) = 1] - \Pr[C(G(x)) = 1]\Big| < 1/n$$

where y is chosen uniformly in $\{0,1\}^n$, and x in $\{0,1\}^l$.

G is said to be quick if $G \in \mathrm{DTIME}[2^{O(l)}]$.

Remark 1 It seems weird the first glance that we refer "quick" to an exponentially running time PRG. However, the Lemma 1 says it does not affect the time complexity for derandomization, so we are safe to apply it for most results that need or is related to derandomization.

We define a special kind of pseudorandom generator that only generates one extra bit, called extender.

Definition 2 **(Extender)** *$G = \{G_:\{0,1\}^l \to \{0,1\}^{l+1}\}$, is called an n-extender if for any circuit C of size n:*

$$\Big|\Pr[C(y) = 1] - \Pr[C(G(x)) = 1]\Big| < 1/n$$

where y is chosen uniformly in $\{0,1\}^{l+1}$, and x in $\{0,1\}^l$.

Similarly, G is said to be quick if $G \in \mathrm{DTIME}[2^{O(l)}]$.

3.1 Quick PRG for Derandomization

Lemma 1 *If there exists a quick pseudorandom generator $G : l(n) \to n$, then for any time constructible bound $t = t(n)$, we have $\mathrm{RTIME}[t] \subseteq \mathrm{DTIME}[2^{O(l(t^2))}]$.*

Proof. We have learned in class that every Turing machine with path length t can be simulated by a circuit of size t^2 using configuration graph of this Turing machine.

To replace a random algorithm who takes $O(t)$ true random bits, we use our quick PRG G that takes seed of size $l(t^2)$ and run in $2^{O(l(t^2))}$ time. Notice that the output of G can fool the circuit of size t^2, and hence any algorithm running in time t as we stated above.

Furthermore, we simulate the random algorithm by iterating over all possible seeds and use our quick PRG as above for each seed. Then simply take the majority vote. Since the number of seeds is of exponential size, the overall complexity is preserved. □

Remark 2 The relaxation on the exponential running time of PRG is mainly due to there are exponential seeds to run before taking a majority vote, which dominates the complexity. This provides a good support and motivation for us to investigate the more flexible PRG and gives more interesting results in potential.

3.2 Hardness

Definition 3 **((ε, S)-hard)** *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. We say that f is (ε, S)-hard if for any circuit C of size S,*

$$\left| \Pr[C(x) = f(x)] - 1/2 \right| < \varepsilon/2$$

where x is chosen uniformly at random in $\{0,1\}^n$.

Yao [Ya82] shows the closeness of approximation can be amplified by applying parity function to multiple copies of f.

Lemma 2 **(Yao)** *Let $f_1, f_2, \ldots, f_k$ be (ε, S)-hard functions, then for any $\delta > 0$, the function $f(x_1, x_2, \ldots, x_k)$ defined by*

$$f(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k} f_i(x_i) \,(\mathrm{mod} 2)$$

is $(\varepsilon^k + \delta, \delta^2(1-\varepsilon)^2 S)$-hard.

The kind of hardness we require

Definition 4 *Let $f : \{0,1\}^* \to \{0,1\}$ be a boolean function. We say that f cannot be approximated by circuits of size $s(n)$ if for some constant k, all large enough n, and all circuits C_n of size $s(n)$:*

$$\Pr[C_n(x) \neq f(x)] > n^{-k}$$

where x is chosen uniformly in $\{0,1\}^n$.

This hardness assumption is "weak" in the sense that it allows non-negligible fraction of error. The Yao's lemma 2 helps amplified the result.

Definition 5 **(hardness of f)** *Let $f : \{0,1\}^* \to \{0,1\}$ be a boolean function, and f_m be the restriction of f to strings of length m. The hardness of f at m, $H_f(m)$ is defined to the maximum integer h_m such that f_m is $(1/h_m, h_m)$-hard.*

Corollary 1 *Let $s(m)$ be any function such that $m \leq s(m) \leq 2^m$; if there exists a function $f \in \mathrm{EXP}$ that cannot be approximated by circuit of size $s(m)$, then for some $c > 0$, there exists a function $f' \in \mathrm{EXP}$ that has hardness $H_{f'}(m) \geq s(m^c)$.*

Remark 3 The corollary is used to formally relate "hardness" of a function to the lower bound of the circuit.

3.3 Main Lemma: Quick PRG with Nearly Disjoint Sets

It is easy to construct an extender given a "hard" function by simply computing this function once, which is hard enough to fool the small circuit. However, the difficulty comes from how to generate multiple bits given our seeds. The idea is to generate each bit with a subset of the seeds, while these subsets are mutually nearly disjoint.

Definition 6 *A collection of sets $\{S_1, \ldots, S_n\}$, where $S_i \subseteq \{1, \ldots, l\}$ is called a (k, m)-design if:*

(1) For all i,

$$|S_i| = m;$$

(2) For all $i \neq j$,

$$|S_i \cap S_j| \leq k.$$

An $n \times l$ 0-1 matrix is called a (k, m)-design if the collection of its n rows, interpreted as subsets of $\{1, \ldots, l\}$, is a (k, m)-design.

Definition 7 (`n-bit PRG`) *Let A be an $n \times l$ 0-1 matrix, let f be a boolean function, and let $x = (x_1 \cdots x_l)$ be a boolean string. Denote by $f_A(x)$ the n bit vector of bits computed by applying the function f to the subsets of the x's denoted by the n different rows of A.*

Our pseudorandom generator expands the seed x of length l to the pseudorandom string $f_A(x)$ of length n. The quality of our PRG is assured by the following lemma:

Lemma 3 *Let m, n, l be boolean integers; let f be a boolean function, $f : \{0,1\}^m \to \{0,1\}$, such that $H_f(m) \geq n^2$; and let A be a boolean $n \times l$ matrix which is a $(\log n, m)$-design. Then $G : l \to n$ given by $G(x) = f_A(x)$ is a pseudorandom generator.*

Remark 4 There are n subsets of seeds, each of size m, with an overlap of at most $\log n$, and the function f can fool circuit of size n^2 on size m input. $\log n < m < l$

Proof. The proof is done by contradiction. Assume such G is not a PRG, then Ya shown that we are able to predict the i-th bit using a probabilistic circuit D that takes previous $i-1$ predicted bits $y_1, \ldots, y_{i-1}$ from $f_A(x)$:

$$\Pr[D_n(y_1, \ldots, y_{i-1}) = y_i] - 1/2 > 1/n^2$$

We want to preserve this bias but rewrite this circuit D to $D'(x_1, \ldots, x_m)$ of size no more than n^2, which will then violates the assumption $H_f(m) > n^2$. The trick is that each y_i depends at most $\log n$ bits of $x_1, \ldots, x_m$ by the definition of NDS that the intersection of subsets is bounded by $\log n$ from above. Therefore, each y_i can be described with the circuit of size exponential to the input size, and hence is of $O(n)$. Iterating over all y_i, we have transformed circuit D to circuit D' with size no more than n^2, but preserving the bias over $1/n^2$. □

3.4 Construction of Nearly Disjoint Sets

We want to make our PRG as simple as possible in the sense that we want the seed length l to be as short as possible. The trivial lower bound is m, while we show that a tight lower bound does not deviate from m much.

Lemma 4 *For all integers n and m, such that $\log n \leq m \leq n$, there exists an $n \times l$ matrix which is a $(\log n, m)$-design, where $l = O(m^2)$. Moreover, the matrix can be computed by Turing machine running in space $O(\log n)$.*

Proof. This is done by constructing subsets over polynomials on $GF(m)$. W.l.o.g. assume m is a prime and $l = m^2$ precisely, then can denote any element in $\{1, \ldots, l\}$ by a pair of numbers in $GF(m)$. Furthermore, we can use polynomial q with degree no more than $\log n$, and get a 1-on-1 correspondence between $\mathbb{Z}_{l+1}^*$ and $S_q = \{(a, q(a)) \mid a \in GF(m)\}$.

Notice that $|S_q| = m$, and $|S_q \cap S_{q'}| \leq \log n$ if $q \neq q'$ by the fundamental theorem of algebra. Also, we have at least n different S_q since $m^{\log n+1} \geq n$. □

Remark 5 l is bounded from above by (m^2), so the gap between seeds length and the subset size is not big.

We know m is bounded from below by $\Omega(\log n)$. For small $m = O(\log n)$, we can optimize the generator seed length l to be the same order of magnitude $O(\log n)$.

Lemma 5 *For all integers n and m, where $m = C \log n$, there exists an $n \times l$ matrix which is a $(\log n, m)$-design, where $l = O(C^2 \log n)$. Moreover, the matrix can be computed by Turing machine in time polynomial in n.*

Proof. The Algorithm 1 simply greedily finds subsets from $\{1, \ldots, l\}$ of size m, which does not violates the $\log n$ intersection criteria.

The algorithm runs in $O(n^2 \times l \times m)$ which is in polynomial time.

There is a guarantee to succeed in finding such NDS. Since the the expectation of joint elements for two random subsets of size m is $m^2/l = \log n$. The Chernoff bounds give a concentration for a constant factor (say deviate from the expectation for $\log n$) is bounded by the inverse of its exponential, which is $1/n$, which is exponentially small compared to the size of our subset.

Algorithm 1 Construction of the $(\log n, C \log n)$-design subsets of $\{1, \ldots, l\}$, where $l = C^2 \log n$

```
procedure ConstructNDS($n, m, l$)
    $\{S_0, S_1, \ldots, S_n\} \leftarrow \emptyset^{n+1}$
    for $k \leftarrow 1$ to $n$ do
        $S_k \leftarrow$ ConstructKthSubset($\{S_1, \ldots, S_k\}, m, l, k$)
    end for
    return $\{S_1, \ldots, S_n\}$
end procedure
procedure ConstructKthSubset($\{S_1, \ldots, S_k\}, m, l, k$)
    val $\leftarrow 1$
    for $i \leftarrow 1$ to $m$ do
        collision_test:
        for $j \leftarrow 1$ to $k$ do
            if $S_k \cup \{\text{val}\}$ collides with $S_j$ for more than $\log n$ elements then
                val $\leftarrow$ val $+ 1$
                goto collision_test
            end if
        end for
        $S_k \leftarrow S_k \cup \{\text{val}\}$
    end for
    return $S_k$
end procedure
```

□

Remark 6 A better seed length bound is given when $m = O(\log n)$. The smaller the subset is, the more optimization space for us to narrow the seed length.

3.5 Main Theorem

Theorem 2 *For every function s, $l \leq s(l) \leq 2^l$, for some $c > 0$, the hardness statement is equivalent to the construction of our quick PRG in the sense:*

$$\exists f \in \text{EXP} \textit{ with hardness } s(l^c) \Longleftrightarrow \exists \textit{ quick pseudorandom generator } G : l \to s(l^c)$$

Proof. First we prove ($\Leftarrow$). It directly implies the existence of a quick $s(l^c)$-extender. Let G_l be such extender, we want to find functions to determine the problem of "Is y in the range of G_l?". An $f \in \text{EXP}$ can be found to compute this problem, and cannot be computed by the circuit of size $s(l^c)$ by definition. Even more, we want it not approximated by the circuit of same size, i.e. even cannot meet with a fraction of error. The work on random-self-reducibility by [Ba91] proves this. By Corollary 1, we concludes this direction.

Then we prove ($\Rightarrow$). Given $f \in \text{EXP}$ with hardness $s(l^c)$, we construct a quick PRG $G : l \to n$, where $n = s(m^{c/4})$. The NDS is constructed according to Lemma 4 where $l = O(m^2)$. Also by our choice of parameter n, we have $H_f(m) > n^2$, so by Lemma 3 we construct it. □

3.6 Sequential Derandomization

If we apply the quick PRG G to replace our true random bits, and perform the sequential derandomization according to Lemma 1, we have the following result:

Theorem 3 *If exists $f \in \text{EXP}$,*

(1) that cannot be approximated by polynomial size circuits,

(2) that cannot be approximated by circuit of size $2^{n^{\varepsilon}}$ for some $\varepsilon > 0$,

(3) with hardness $2^{\varepsilon n}$ for some $\varepsilon > 0$,

then,

(1) $\text{BPP} \subsetneq \text{EXP}$,

(2) $\text{BPP} \subseteq \text{DTIME}[2^{(\log n)^c}]$ *for some $c > 0$,*

(3) $\text{BPP} = \text{P}$

Proof. (1) implies $G : n^{\varepsilon} \to n$ for $\forall \varepsilon > 0$,

(2) implies $G : (\log n)^c \to n$ for some $c > 0$,

(3) implies $G : C \log n \to n$ for some $C > 0$, which construct NDS from Lemma 5 to construct an optimized PRG as in the main theorem.

Then apply Lemma 1 to perform the derandomization. □

4 Discussion

4.1 Related Work

The research on lower bounds for constant depth circuits is pioneered by Nisan [Ni91]. Since the lower bounds for AC^0 are known, we can give unconditional construction of the PRG based on the lower bound for parity function due to Hastard [Hå86].

Theorem 4 *For any family $\{C_n\}$ of circuits of depth d and size at most $2^{n^{1/(d+1)}}$, and for all large enough n,*

$$\left| \Pr[C_n(x) = \mathit{parity}(x)] - 1/2 \right| \leq 2^{-\Omega(n^{1/(d+1)})}\dagger$$

when x is chosen uniformly over all n-bit strings.

† There exists a typo in their paper, pointed by Jin-Yi Cai, where the bound should contain a negative sign according to Nisan's work [Ni91]. Specifically, apply his Lemma 8.6 condition (ii), and pick $s < n^{1/k}$, and $k = d + 1$.

Applying parity function to our construction of PRG, and this lower bound, without proof we get the result that

Theorem 5 *$RAC^0 \subseteq$ DSPACE[poly-log] and $RAC^0 \subseteq$ DTIME[$2^{\text{poly-log}}$]*

Recently, a similar promise-based result [CT22] is developed by Lijie Chen[‡].

4.2 Amplification

The power of amplification between polynomial and exponential is of great importance. A direct usage of the amplification from $O(\log n)$ bits seed to the $O(n)$ bits pseudorandom sequence, i.e. our quick PRG $G : C \log n \to n$ from Lemma 5, is to prove the Sipser-Lautemann theorem [Si83; La83] that BPP $\in \Sigma_2^p \cap \Pi_2^p$:

Theorem 6 *If there exists quick $G : C \log n \to n$, then we can simulate languages in* BPP*. To find and verify such G, we can utilize a Σ_2^p machine.*

Proof. The algorithm goes like this: (1) Nondeterministically guess a function on $O(\log n)$ bits with hardness $\Omega(n^2)$ (first alternation).

(2) Verify that it is indeed hard (second alternation), which is a coNP problem

(3) Use it as the "hard" f for our PRG

(4) Try all possible seeds, in total $O(n)$ as seed is of $O(\log n)$ length. □

4.3 Vast Connections

We have seen many amplification using polynomial manipulations to get an exponential change on certain quantity. Amplification is a powerful tool to enable us perform efficient algorithms and viable proofs, just as the Sipser-Lautemann theorem mentioned above. The mechanism behind this amplification is from our PRG. But where does this power come from our PRG?

It seems to me that the concentration theory is one of the roots of cause. The construction of our $\log n$ size compact PRG (Lemma 5) requires the Chernoff bound, where quantity over polynomial times summation is bounded by an exponential term.

The similar discovery is also laid in some of the proofs in complexity theory when dealing with probabilistic Turing machine (PTM). It is the isolation lemma helps connects the PTM classes with the polynomial hierarchy (e.g. the alternative proof for Theorem 6 using isolation lemma). As we know, the polynomial hierarchy (PH) is related to many other fields in complexity theory, including oracles (since we have two alternative equivalent definition of PH), first-order logic (the alternation

[‡] Lijie Chen was actively participating competitive programming test, dominating an era in Olympiad Informatics contests when I was also engaged in some series competition in China. It is interesting to see his name and work a decade later in theoretical computer science. Though he is way more clever and diligent than me, we all move forward to another vast and arousing world.

of our quantifiers), Turing machine (when boils down to the specific case in P and NP, especially the nondeterminism idea), circuits (as it can logic formula is easily mutually translated to boolean circuits), and interactive proof system (which can also be described with logics and oracles). It seems to me that isolation lemma is one of the amazing ways to bridges many seemingly unrelated fields together. What beneath the isolation lemma is the amplification of PTM classes, which uses Chernoff bound again.

The astonishing beauty for math and theoretical computer science for me is the elegance yet powerful results that connects many things at the first glance are not trivial or even weird, but turns out to share the common traits. Computational complexity is such a field.

5 Acknowledgement

Professor Jin-Yi Cai is a great professor who provides smooth lectures and valuable insights which are hard for me to catch up at the beginning but hence motivates me to dive deeper into the TCS world, and seems like an interesting person who jokes and smiles from time to time.

I also appreciate myself in choosing this theory class (and also UW–Madison) which finds myself a good way to enjoy my leisure time and overcome loneliness.

Also thanks to Jinlang, who gives me opportunities to teach her after-class, which both builds up my confidence and improves my understanding of the materials better.

Most importantly, thanks to my family members who have supported my study so far.

References

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. “PRIMES is in P”. In: *Annals of mathematics* (2004), pp. 781–793.

[Ba91] L Babai, L Fortnow, N Nisan, and A Wigderson. “BPP has weak subexponential simulations unless EXPTIME has publishable proofs”. In: *proceedings of structures in complexity theory*. 1991.

[Ca03] Jin-Yi Cai. *Lectures in Computational Complexity*. `https://pages.cs.wisc.edu/~jyc/710-draft-book.pdf`. [Online; accessed 03-May-2024]. 2003.

[CT22] Lijie Chen and Roei Tell. “Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise”. In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 125–136.

[Hå86] Johan Håstad. “Computational limitations for small depth circuits”. PhD thesis. Massachusetts Institute of Technology, 1986.

[La83] Clemens Lautemann. “BPP and the polynomial hierarchy”. In: *Information Processing Letters* 17.4 (1983), pp. 215–217.

[Mi75] Gary L Miller. “Riemann’s hypothesis and tests for primality”. In: *Proceedings of the seventh annual ACM symposium on Theory of computing*. 1975, pp. 234–239.

[Ni91] Noam Nisan. “Pseudorandom bits for constant depth circuits”. In: *Combinatorica* 11 (1991), pp. 63–70.

[NW94] Noam Nisan and Avi Wigderson. “Hardness vs randomness”. In: *Journal of computer and System Sciences* 49.2 (1994), pp. 149–167.

[Ra80] Michael O Rabin. “Probabilistic algorithm for testing primality”. In: *Journal of number theory* 12.1 (1980), pp. 128–138.

[Si83] Michael Sipser. “A complexity theoretic approach to randomness”. In: *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. 1983, pp. 330–335.

[Ya82] Andrew C Yao. “Theory and application of trapdoor functions”. In: *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*. IEEE. 1982, pp. 80–91.