1 Uniform circuits

One rationale for working with circuits was modeling computation using graphs, which we like to work with. In non-uniform circuits, we abstracted out some computational aspects. Can we interpolate the two rationales? That is, model computation as graphs, but disallow non-uniformity?

The study of uniform circuits isn't as extensive as that of non-uniform circuits, but it is well-established, dating back at least to the 1980's. There are many natural definitions, I'll give examples.

Definition 1 (uniform circuits)**.** *Consider a circuit family* $\{C_n : \{0,1\}^n \to \{0,1\}\}$ *of size* $s(n)$*. We say that* $\{C_n\}$ *is:*

- $\mathcal{P}$**-uniform:** *There is an algorithm that gets input* 1^n*, runs in time* $\mathrm{poly}(s)$ *and prints* C_n*.*
- **Logspace-uniform.** *There is an algorithm that gets input* 1^n*, runs in space* $O(\log s)$ *and prints* C_n*.*
- **Polylogtime-uniform.** *There is an algorithm that gets input three indices* (w, u, v) *of gates in* C_n*, runs in time* $\mathrm{polylog}(s)$ *(i.e., polynomial time in its input length), and decides gates* (u, v) *feed into* w*.*[1]

These definitions are natural and useful. The framework of uniform circuits is also more flexible than how it may seem at first glance, allowing to capture several different interesting notions. I'll present two notions along with some results.

Notion 1: Strong circuits, very strict uniformity. Consider, for example, polynomial size circuit families that are polylogtime-uniform.

The rationale behind this notion is that we struggle with proving lower bounds for non-uniform circuits, so why don't we impose the strictest uniformity condition we can, prove lower bounds, and then work our way up to more relaxed notions of uniformity. For proving (say) $\mathcal{P} \neq \mathcal{NP}$, this approach doesn't give much hope:

Theorem 2. *Any* $L \in \mathcal{DTIME}[T]$ *can be solved in polylogtime-uniform size* $O(T^2)$*.*

Proof. The standard simulation of TMs by circuits (as in Cook-Levin) already gives this, because all parts of the circuit are wired according to the same constant-sized gadget (representing the transition function of the TM). ■

Nevertheless, the strict uniformity condition is still non-trivial (i.e., it cannot be assumed wlog), because the simulation in Theorem 2 breaks down when considering $\mathcal{P}$-uniform circuit families of bounded size, and in particular of bounded depth. First,

[1] As before, we are dealing with general circuits and assume that their fan-in is two. For polylogtime-uniformity, it is simplest to assume that all negations are at the bottom and that the circuit is layered (i.e., alternating $\wedge$ and $\vee$ layers), or alternatively require the uniformity algorithm to also decide gate types.

when trying to simulate such families, we need to pay for the runtime of the outer "$\mathcal{P}$-uniform" machine.[2] Secondly, for bounded-depth Boolean circuits, we don't know how to simulate them by polylogtime-uniform circuits while preserving depth.[3]

In particular, for some "frontier" classes bounded-depth circuits (such as $\mathcal{ACC}$ or $\mathcal{TC}^0$), this notion is interesting and can chart a useful path for progress. In fact, there is a line of works that gets better lower bounds than the ones we have for the non-uniform counterparts! For example:

Theorem 3 ([AG94])**.** *The permanent cannot be computed logtime-uniform $\mathcal{ACC}$ circuits of size 2^{n^ϵ}, for some $\epsilon > 0$.*[4]

Theorem 4 ([All99])**.** *The permanent cannot be computed by logtime-uniform $\mathcal{TC}^0$ circuits of quasipolynomial size (in fact even larger, but the exact bound is less straightforward).*

Fixed-polysize circuits, more relaxed uniformity. Consider, for example, $\mathcal{P}$-uniform circuit families of size n^k. Think of this as fixed-polytime computation with strong preprocessing (that depends on the input length but not on any specific input).

The main interesting thing about this notion is that it is more tractable than one might a-priori think. For example, observe that the naive proof of the time hierarchy fails for this notion – it's not immediately clear how to prove that time n^{100} is hard for $\mathcal{P}$-uniform circuits of size n^{50}, because the "$\mathcal{P}$-uniform" machine is allowed more time than n^{100}. Nevertheless, it turns out that this type of hierarchy result is true!

Theorem 5 ([SW13])**.** *$\mathcal{P}$ hard for $\mathcal{P}$-uniform circuits of size n^k.*

Proof. Assume otherwise. Let $L \in \mathcal{P}$ hard for time n^{100k} with n^ϵ advice. Then L has $\{C_n\}$ of size n^k. The DCL of $\{C_n\}$ in $\mathcal{P}$-uniform n^k, so padded one also in $\mathcal{P}$-uniform n^k. In particular there is a description of C_n of size $< n^\epsilon$. Given the description as advice, run it to compute L, all in time n^{2k}. ■

An analogous result also holds for space-bounded machines (i.e., logspace-uniform circuits that are evaluable in bounded space), and it turns out that this type of result is tightly connected to the $\mathcal{BPL} = \mathcal{L}$ question; for example, proving a scaled-up result for space implies derandomization of $\mathcal{BPSPACE}[O(n)]$ (see [DPT24; DPT+25; PT25]).

[2] To see the impossibility result, consider a $\mathcal{P}$-uniform circuit family in which the "$\mathcal{P}$-uniform" machine computes a unary function hard for time $\text{poly}(T)$, and hard-wires the outcome into the description of the circuit.

[3] I say "Boolean" because this is actually possible in the arithmetic setting, wherein we have a classical depth reduction, and it can be carried out while preserving uniformity (see [AKT25]).

[4] Note that in terms of input length, the permanent is computable in time $2^{O(\sqrt{n})}$ (since the permanent of an $m \times m$ matrix is computable in time $2^{O(m)}$).

2 An algorithmic approach to lower bounds for uniform circuits

We'll see a recent algorithmic approach to lower bounds for uniform circuits, introduced recently by Santhanam [San23].

Definition 6. *An algorithm A is a* non-trivial sampling algorithm *if there is $k \in \mathbb{N}$ such that, given $C\colon \{0,1\}^n \to \{0,1\}$ of arbitrary polysize with acceptance probability $\geq 2/3$, the algorithm $A(C)$ runs in space n^k, tosses n^k coins, and there is some $y \in C^{-1}(1)$ such that $\Pr[A(C) = y] \geq n^{10}/2^n$.*

The crucial point in the definition is that A's resource bound n^k does not grow with its input size (i.e., with the size of C); indeed, it is not even clear that A can simulate C. Without this restriction, the samlping task seems clearly doable: for example, assuming that $prBPP = prP$, we can find a satisfying assignment for C in time (and space) $\text{poly}(|C|)$, and solve the sampling problem with success bound 1 rather than only $n^{10}/2^n$. With the restriction of n^k space and coins, it's not even immediately that the task is doable, let alone that it's a feasible path to proving lower bounds.

Fortunately, the problem that A needs to solve is indeed relaxed enough. Let's first see evidence that non-trivial sampling is feasible, and then see that non-trivial sampling implies lower bounds for uniform circuits.

2.1 Non-trivial sampling is feasible

Assuming cryptography, the task is doable for general circuits, and in fact with an algorithm meeting stricter requirements.

Theorem 7. *Assuming non-uniformly secure OWFs, there is a non-trivial sampling algorithm.*

Proof. Print a random output of a cryptographic PRG with stretch $n^\epsilon \to n$, running in fixed polytime n^k. For every polysize circuit with high acceptance probability, there is some string in the output-set of the PRG that the circuit accepts, and each string in the output-set is printed with probability $2^{-o(n)} \gg n^{10}/2^n$. ■

Note that the algorithm above ran in time n^k (rather than only space n^k), did not even look at its input (let alone simulate it), and had larger success probability than what is needed (and a stronger cryptographic assumption yields higher success probability). The point in the proof above is that cryptography allows us to get PRGs that run in time smaller than their adversaries.

For restricted circuit classes, the sampling task is unconditionally solvable. In particular, the task is doable for circuits that can be evaluated in sublinear space.

Theorem 8. *There is a non-trivial sampling algorithm for branching programs.*

Proof. Given a BP, sample n^{20} strings and output the lex-first accepted one. This algorithm runs in space $O(n)$,[5] and uses n^{21} coins. Let S be the set of $2^n/n^{15}$ accepted strings that are lex-first in $\{0,1\}^n$. Then, this algorithm hits S wp at least $2/3$, and whenever it hits S it outputs some string in S (because these are the lex-first accepted strings in $\{0,1\}^n$). Thus, some $s \in S$ is output wp at least $(2/3)/|S| > n^{10}/2^n$. ∎

2.2 Non-trivial sampling implies lower bounds for uniform circuits

The paper [San23] has several variations on the theme "non-trivial sampling ⇒ lower bounds for uniform circuits". We will focus on a particularly clean result statement.

Theorem 9 ([San23])**.** *If there exists a non-trivial sampling algorithm for general circuits, then, $\mathcal{P} \neq \mathcal{PSPACE}$. Moreover, a non-trivial sampling algorithm for a class $\mathcal{C}$ implies that $\mathcal{PSPACE}$ doesn't have logspace-uniform $\mathcal{C}$-circuits.*

Note that Theorem 9 does not directly mention uniform circuits, but the proof goes through uniform circuits. Also, other statements in [San23] refer to uniform circuits more directly. Moreover, they allow, for example, to get lower bounds even in $\mathcal{NP}$ if we consider a variation on the sampling problem (in which we get a "succint" version of C as input), and in this case the sampler is even allowed a $\mathcal{PH}$ oracle.

Proof. The intuition is that if a string is sampled with overly high probability (i.e., more than 2^{-n}), then we can *compress* this string; that is, the string has a description of less-than-maximal size. In particular, incompressible strings cannot be sampled with non-trivial probability. If $\mathcal{P} = \mathcal{PSPACE}$ then we can decide whether a string is incompressible (for an appropriate notion of "incompressible") by a highly uniform circuit. But in this case, the sampler will sample an output of this circuit with non-trivial probability, and we will be able to compress it – a contradiction. Details follow.

A notion of compressibility, and a coding theorem. First we'll need an appropriate notion of "compressibility". We'll use a computationally bounded version of K-complexity. Think of the K-complexity when restricted to programs running in small space, and let's throw randomness into the mix.

Definition 10. *We say that $y \in \{0,1\}^n$ has ρ-confidence psK^t complexity m if*

$$\Pr_{r \in \{0,1\}^{t(n)}} [\exists P \text{ of length} \leq m \text{ st } P(r) = y \text{ in space } t(n)] \geq \rho \,.$$

The use of randomness here isn't trivial, because the program P may depend on the randomness r. Nevertheless, this is still a reasonable notion. In particular, as usual, a random n-bit string has $2/3$-confidence psK^t complexity at least $n - C$, regardless of t.[6] Also note that we can decide whether or not a string has ρ-confidence psK^{n^k} at least m in $\mathcal{PSPACE}$ (by enumerating over coins and over programs).

[5] It samples n^{20} strings one-by-one, while continuously storing the lex-least accepted string so far.

[6] This follows by a counting argument over pairs (p, r); each non-"yes" instance eliminates a program p and a $1/3$-fraction of r's.

The following lemma asserts that strings sampled with non-trivial probability $\gg 2^{-n}$ can be compressed, for the above notion of compressibility.

Lemma 11. *If there's an algorithm S running in space n^k and with n^k coins that samples $y \in \{0,1\}^n$ wp $\geq p$, then the 2/3-confidence $psK^{n^{O(k)}}$ of y is at most $\log(1/p) + 3\log(n)$.*

Proof. Consider a random $h\colon \{0,1\}^{\log(1/p)+O(1)} \to \{0,1\}^{n^k}$. We say that h is good for y if $y = S(h(x))$ for some x. Then

$$\Pr[h \text{ is not good}] = (1-p)^{O(1/p)} \leq .01 \ .$$

Let us now think of the description of h as a truth-table of size $O(n^k/p)$. The argument above says that a random truth-table is good for y wp $\geq .99$. We want to argue that a *pseudorandom* truth-table is also good for y. To do this, we decide if a truth-table is good by a very simple procedure: In particular, there is a depth-3 circuit $C = C_y$ of size $N = 2^{n^{O(k)}}$ that gets a truth-table and accepts iff h is good for y.[7] We can thus use strongly-explicit PRGs for DNFs:

Theorem 12 ([Nis91])**.** *There is a PRG G for depth-3 circuits of size N with error .01 and seed length $\text{polylog}(N)$ such that given s and i we can compute $G(s)_i$ in time $\text{polylog}(N)$.*

We instantiate this PRG with $N = 2^{n^{O(k)}}$ and with output length $O(n^k/p) \ll N$, and for every seed $s \in \{0,1\}^{n^{O(k)}}$, denote by $h_s(\cdot)$ the function described by the string $G(s)_1, ..., G(s)_N$. Since a random N-bit string is good whp and this is a PRG, wp at least .98 over choice of s there is x such that $S(h_s(x)) = y$.

Hence, wp more than 2/3 over s, there is a small description (x, G, S, h, n) of a procedure $P = P_{x,G,S,h,n}$ such that $P(s) = y$ (i.e., $P(s)$ prints $S(h_s(x))$). The procedure P runs in space $n^k + Space(h) \leq n^{O(k)}$, and its description is is of size

$$|x| + \lceil \log n \rceil + O(1) < \log(1/p) + 2\log(n) + O(1) \ . \quad \square$$

The proof itself. Let $\Pi = \Pi_k$ be the promise-problem in which "yes" instances do not have 1/3-confidence psK^t complexity $n - C$, and "no" instances have 2/3-confidence psK^t complexity $n - C$, where $t = n^{O(k)}$.[8] Note that $\Pi \in prPSPACE$.

If $\mathcal{P} = PSPACE$, then Π can be decided in polynomial time, and hence by a polylogtime-uniform circuit family $\{C_n\}$ of polysize.

Let us perform the following mental experiment, for some large enough $n \in \mathbb{N}$. We run the sampler, and provide it virtual access to C_n. The sampler runs in space $n^k \cdot \text{polylog}(n) < n^{2k}$ and uses n^k random coins, and it samples some "yes" string of Π wp $n^{10}/2^n$. By Lemma 11, that string has 2/3-confidence psK^t complexity at most $n - 10 \cdot \log(n) + 3\log(n) < n - C$, a contradiction. ■

[7] The depth-3 checks (by brute-force) if there is an entry in the truth-table that equals coins leads S to output y. Note that the circuit does not need to implement S; the set of good coins is hard-wired.

[8] The O-notation hides the universal constant from Lemma 11. What we need from C is just that at least 0.9 of strings will be "yes" instances.

Remark 13. *The proof above is a simplified version of the proof that appears in [San23]. The latter also supports statements that hold for weaker circuit classes.*

References

[AG94] Eric Allender and Vivek Gore. "A uniform circuit lower bound for the permanent". In: *SIAM Journal on Computing* 23.5 (1994), pp. 1026–1049.

[AKT25] Robert Andrews, Deepanshu Kush, and Roei Tell. "Polynomial-time PIT from (almost) necessary assumptions". In: *Proc. 57th Annual ACM Symposium on Theory of Computing (STOC)*. [2025] ©2025, pp. 1087–1095.

[All99] Eric Allender. "The permanent requires large uniform threshold circuits". In: *Chicago Journal of Theoretical Computer Science* (1999), Article 7, 19.

[DPT24] Dean Doron, Edward Pyne, and Roei Tell. "Opening Up the Distinguisher: A Hardness to Randomness Approach for BPL = L that Uses Properties of BPL". In: *Proc. 56th Annual ACM Symposium on Theory of Computing (STOC)*. 2024.

[DPT+25] Dean Doron, Edward Pyne, Roei Tell, and Ryan Williams. "When Connectivity is Hard, Random Walks are Easy With Non-Determinism". In: *Proc. 57th Annual ACM Symposium on Theory of Computing (STOC)*. 2025.

[Nis91] Noam Nisan. "Pseudorandom bits for constant depth circuits". In: *Combinatorica* 11.1 (1991), pp. 63–70.

[PT25] Edward Pyne and Roei Tell. "Composing Low-Space Algorithms". In: *Electronic Colloquium on Computational Complexity: ECCC* (2025).

[San23] Rahul Santhanam. "An algorithmic approach to uniform lower bounds". In: *Proc. 38th Annual IEEE Conference on Computational Complexity (CCC)*. 2023, Art. No. 35, 26.

[SW13] Rahul Santhanam and R. Ryan Williams. "On medium-uniformity and circuit lower bounds". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. IEEE, 2013, pp. 15–23.