Are all distributions easy?

Emanuele Viola[*]

November 5, 2009

Abstract

Complexity theory typically studies the complexity of computing a function $h(x) : \{0,1\}^n \to \{0,1\}^m$ of a given input x. We advocate the study of the complexity of generating the distribution $h(x)$ for uniform x, given random bits.

Our main results are:

- There are explicit AC^0 circuits of size $\mathrm{poly}(n)$ and depth $O(1)$ whose output distribution has statistical distance $1/2^n$ from the distribution $(X, \sum_i X_i) \in \{0,1\}^n \times \{0,1,\ldots,n\}$ for uniform $X \in \{0,1\}^n$, despite the inability of these circuits to compute $\sum_i x_i$ given x.

  Previous examples of this phenomenon apply to different distributions such as $(X, \sum_i X_i \mod 2) \in \{0,1\}^{n+1}$.

  We also prove a lower bound independent from n on the statistical distance between the output distribution of NC^0 circuits and the distribution $(X, \mathrm{majority}(X))$. We show that $1 - o(1)$ lower bounds for related distributions yield lower bounds for succinct data structures.

- Uniform randomized AC^0 circuits of $\mathrm{poly}(n)$ size and depth $d = O(1)$ with error ϵ can be simulated by uniform randomized circuits of $\mathrm{poly}(n)$ size and depth $d+1$ with error $\epsilon + o(1)$ using $\leq (\log n)^{O(\log\log n)}$ random bits.

  Previous derandomizations [Ajtai and Wigderson '85; Nisan '91] increase the depth by a constant factor, or else have poor seed length.

Given the right tools, the above results have technically simple proofs.

[*]Supported by NSF grant CCF-0845003. Email: `viola@ccs.neu.edu`

1 Introduction

Complexity theory, with some notable exceptions, typically studies the complexity of computing a function $h(x) : \{0,1\}^n \to \{0,1\}^m$ of a given input x. We advocate the study of the complexity of generating the output distribution $h(x)$ for random x, given random bits. This question can be studied for a variety of computational models. In this work we focus on small bounded-depth circuits with unbounded fan-in (AC^0) or bounded fan-in (NC^0).

An interesting example of a function h for which computing $h(x)$ is harder than generating its output distribution is $h(x) := (x, \mathrm{parity}(x))$, where $\mathrm{parity}(x) := \sum_i x_i \mod 2$. Whereas AC^0 circuits cannot compute parity (cf. [Hås87]), Babai [Bab87] and Boppana and Lagarias [BL87] show a simple NC^0 circuit C whose output distribution equals that of $(x, \sum_i x_i \mod 2)$ for random $x \in \{0,1\}^n$:

$$C(x_1, x_2, \ldots, x_n) := (x_1, x_1 + x_2, x_2 + x_3, \ldots, x_{n-1} + x_n,\ x_n). \tag{1}$$

Later, Impagliazzo and Naor [IN96] extend this to show that small AC^0 circuits can even generate $(x, f(x))$ for more complicated functions such as inner product, i.e., $f(x) := x_1 \cdot x_2 + x_3 \cdot x_4 + \cdots + x_{n-1} \cdot x_n$. They use this to construct cryptographic pseudorandom generators computable by poly-size AC^0 circuits based on the hardness of the subset sum problem, and similar techniques are useful in constructing depth-efficient generators based on other assumptions [AIK06, Vio05].

To our knowledge, no other example was known of a function such that generating $(x, f(x))$ is easier than computing $f(x)$. Also, all the above examples exploit the same cancelation properties of telescopic sums. In this work, we prove that small AC^0 circuits can generate $(x, f(x))$ for any symmetric function f, including for example the majority function, except for an exponentially small statistical distance. This is an immediate corollary of the following theorem, stating that one can generate $(x, \sum_i x_i) \in \{0,1\}^n \times \{0,1,\ldots,n\}$.

Theorem 1.1. *There are explicit* AC^0 *circuits* $C : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n \times \{0,1,\ldots,n\}$ *of size* $\mathrm{poly}(n)$ *and depth* $O(1)$ *whose output distribution has statistical distance* $\leq 2^{-n}$ *from the distribution* $(x, \sum_i x_i) \in \{0,1\}^n \times \{0,1,\ldots,n\}$ *for random* $x \in \{0,1\}^n$.

Theorem 1.1 has a short proof relying on the results by Matias and Vishkin [MV91] and Hagerup [Hag91] about depth-efficient generation of random permutations of $[n]$ (a stripped-down version of their result is presented in §A). Specifically, once a random permutation π of $[n]$ is generated, we only need to select $s \in \{0,1,\ldots,n\}$ binomially distributed, let $x \in \{0,1\}^n$ be the string where exactly the bits at position $\pi(i)$ for $i \leq s$ are set to 1, and output (x, s).

The circuits in Theorem 1.1 have two shortcomings over those of the previous examples (1) and [IN96]. First, they use many random bits, while (1) and [IN96] do not waste randomness, which is useful for constructing generators. Also, the circuits in Theorem 1.1 only generate a distribution that is close to the desired distribution, while in previous examples the distributions are equal. We do not know if these shortcomings can be overcome. More generally, we raise the challenge of proving lower bounds for generating distributions.

Challenge 1.2. *Exhibit an explicit map $h : \{0,1\}^{\text{poly}(n)} \to \{0,1\}^n$ such that the output distribution $h(X) \in \{0,1\}^n$ cannot be generated by $\text{poly}(n)$-size AC^0 circuits given random bits as inputs.*

Current lower-bounding techniques appear unable to tackle questions such as Challenge 1.2 (which, to our knowledge, is open even for DNFs). Note for example that standard "hard functions" f such as parity, inner product, $\mod 3$, and majority as we have seen all have the property that $(x, f(x))$ can be (almost) exactly generated by small AC^0 circuits. This suggests that our understanding of even these simple models is incomplete, and that pursuing the above direction may yield new proof techniques.

More concretely, the next section provides a motivation related to data structures lower bounds.

Motivation: Data structures lower bounds. Succinct data structures aim to compress data using a number of bits close to the information-theoretic minimum while at the same time supporting interesting queries (see, e.g., [Păt08] and the pointers there). For concreteness, we focus on the so-called membership problem, which asks to store a subset x of $[n]$ of size ℓ (think of x as an n-bit string of weight ℓ) in $\lceil \log_2 \binom{n}{\ell} \rceil + r$ bits, where r is as small as possible, while being able to answer the query "is i in x" by reading few bits of the data structure. We note that the bounds on this central problem are not tight [BMRS02, Pag01a, Pag01b, Păt08, Vio09]. For example, when $\ell := n/4$ we are unaware of lower bounds.

We observe that a lower bound for generating a distribution somewhat close to the uniform distribution on subsets of size ℓ yields a data structure lower bound.

Claim 1.2.1. *Suppose one can store subsets x of $[n]$ of size $n/4$ in $m := \left\lceil \log_2 \binom{n}{n/4} \right\rceil + r$ bits, while answering "is i in x" by non-adaptively reading q bits of the data structure. Then there is a circuit $C : \{0,1\}^m \to \{0,1\}^n$ whose output distribution has statistical distance at most $1 - 2^{-r-1}$ from the uniform distribution over subsets of $[n]$ of size $n/4$, and such that each output bit of C depends on only q input bits.*

Proof. The i-th output bit of C is the algorithm answering "is i in x." Think of feeding C random bits. With probability $\binom{n}{n/4}/2^{\lceil \log_2 \binom{n}{n/4} \rceil + r} \geq 1/2^{r+1}$ the input is uniform over encodings of subsets of $[n]$ of size $n/4$, in which case the statistical distance is 0. If we distinguish in every other case, the distance is at most $1 - 1/2^{r+1}$. □

Similar considerations extend to adaptive bit-probes and cell probes. The latter case is where memory is divided in cells of $\log n$ bits each. We note that q cell probes can be simulated by an $n^{O(q)}$-size DNF, thus sufficiently strong lower bounds for generating these distributions by DNFs yield lower bounds in the cell-probe model. We note that the proof of Theorem 1.1 also shows that small AC^0 circuits can generate the uniform distribution over sets of size ℓ with error at most $1/2^n$, for any given ℓ. But these circuits both have large depth > 2 and use many random bits, two reasons why this is not an obstacle to the above data structure application. While one could prove lower bounds for data structures without using this approach, Claim 1.2.1 appears to suggest an uncharted direction, along which we now report some partial progress.

For distributions whose output bits depend only on d input bits (i.e., generated by NC^0 circuits) we are able to obtain statistical distance $1/2^{O(d)}$ which is independent from n. Improving this lower bound to distance close to 1 would have data structures consequences via Claim 1.2.1 (for simplicity the next theorem is stated for sets of size $n/2$ rather than $n/4$ as in Claim 1.2.1).

Theorem 1.3. *Let n be even and $C : \{0,1\}^{d \cdot n} \to \{0,1\}^n$ be a function such that each output bit C_i of C depends on only d inputs bits. Then the output distribution of C (for random input) has statistical distance at least $1/2^{O(d)}$ from the uniform distribution over (characteristic vectors of) sets $x \subseteq [n]$ of size $n/2$.*

Proof. Suppose the output distribution of C is 4-wise independent. Then by standard anti-concentration inequalities such as the Paley-Zygmund inequality (alternatively, ask for larger independence and use [DGJ+09]), $\Pr_x[\sum_i C_i(x) > n/2] \geq \Omega(1)$. The statistical test which checks if the output bits sum to $n/2$ proves the claim in this case.

Otherwise, there are 4 output bits of C that are not uniformly distributed over $\{0,1\}^4$. Since these 4 bits depend on at most $4d$ input bits, there must be an output combination that has probability at least $1/2^4 + 1/2^{4d}$. But, when the output is a uniform set of size $n/2$, this output combination of the 4 bits has probability at most

$$\frac{1}{2} \cdot \frac{n/2}{n-1} \cdot \frac{n/2}{n-2} \cdot \frac{n/2}{n-3} \leq \frac{1}{2^4}(1 + o(1)).$$

So, just looking at these four bits, we get statistical distance $1/2^{O(d)}$. □

It is possible to push the above techniques to obtain a similar lower bound for generating $(x, \mathrm{majority(x)}) \in \{0,1\}^{n+1}$, where $\mathrm{majority}(x) = 1 \Leftrightarrow \sum_i x_i \geq n/2$.

Theorem 1.4. *Let n be odd and $C : \{0,1\}^{d \cdot n} \to \{0,1\}^{n+1}$ be a function such that each output bit C_i of C depends on only d inputs bits. Then the output distribution of C (for random input) has statistical distance at least $1/2^{O(d)}$ from $(x, \mathrm{majority(x)}) \in \{0,1\}^{n+1}$ for uniform $x \in \{0,1\}^n$.*

Pseudorandom generators The ability to generate a distribution efficiently has obvious applications in pseudorandomness which we now elaborate upon. The ultimate goal of derandomization of algorithms is to remove, or reduce, the amount of randomness used by a randomized algorithm while incurring the *least possible* overhead in other resources, such as time. Typically, this is achieved by substituting the needed random bits with the output of a pseudorandom generator. There are two types of generators. Cryptographic generators [BM84, Yao82] (a.k.a. Blum-Micali-Yao) use less resources than the algorithm to be derandomized. In fact, computing these generators can even be done in the restricted circuit class NC^0 [AIK06]. However, unconditional instantiations of these generators are rare, and in particular we are unaware of any unconditional cryptographic generator with large stretch, a key feature for derandomization. By contrast, Nisan-Wigderson generators [NW94] use more resources than the algorithm to be derandomized, and this looser notion of efficiency allows for more unconditional results [Nis91, NW94, LVW93, Vio07]. Moreover, all of these results yield generators with large, superpolynomial stretch.

In particular, Nisan [Nis91] shows a generator that fools small AC^0 circuits of depth d with exponential stretch, or seed length $\log^{O(d)} n$. As mentioned above, this generator uses more resources than the circuits to be derandomized. Specifically, it computes the parity function on $\geq \log^d n$ bits, which requires AC^0 circuits that have either depth $\geq d$ or superpolynomial size. Thus, if one insists on polynomial-size circuits, the derandomized circuit, consisting of the circuit computing the generator and the original circuit, has depth at least twice that of the original circuit. This constant factor blow-up in depth appears necessary for Nisan-Wigderson constructions.

In this work we present a derandomization which only blows up the depth by 1, and uses a number of random bits close to Nisan's (an improvement in the tools we use would let us match the number of random bits in Nisan's result).

Theorem 1.5 (Depth-efficient derandomization of AC^0)**.** *The following holds for every d. Let $f : \{0,1\}^* \to \{0,1\}^*$ be computable by uniform randomized AC^0 circuits of $\mathrm{poly}(n)$-size and depth d with error ϵ. Then f is computable by uniform randomized circuits of $\mathrm{poly}(n)$-size and depth $d+1$ with error $\epsilon + o(1)$ using $\leq (\log n)^{O(\log\log n)}$ random bits.*

Theorem 1.5 is proved by exhibiting a generator whose output looks random to small AC^0 circuits, and yet each of its output bits can be computed by a DNF, i.e., a depth-2 circuit (of size $n^{O(d)}$). Some evidence that such a generator may exist comes from Example (1), which implies a generator mapping $n-1$ bits to n bits that can be shown to look random to AC^0 circuits, and yet each output bit just depends on 2 inputs bits. However, the seed length of this generator is very poor, and it is not clear how to improve on it. Intuitively, one would like to be able to generate the output distribution of Nisan's generator [Nis91] more efficiently than shown in [Nis91]. We were not able to do so, and we raise this as another challenge.

For Theorem 1.5, we notice that the recent line of work by Bazzi, Razborov, and Braverman [Bra09] shows that any distribution that is $(k := \log^c n)$-wise independent looks random to small AC^0 circuits of depth d, for a certain constant $c = c(d) \geq d$.

We show how such distributions can be generated by DNFs. Although the constructions of k-wise independent distributions in [CG89, ABI86, GV04] all require iterated sums of k bits, which for $k := \log^c n$ is unfeasible in our setting, we give an alternative construction using the recent unique-neighbor expanders by Guruswami, Umans, and Vadhan [GUV09], resulting in a generator with seed length $s := (\log n)^{O(\log\log n)}$. Specifically, the results in [GUV09] give explicit bipartite expanders with n nodes on the left, s nodes on the right, and left degree $O(\log n)$ such that any subset of left-hand nodes of size $\leq k = \log^c n$ has at least one unique neighbor. Wed define the i-th output bit of our $(\log^c n)$-wise independent generator as the xor of the $O(\log n)$ neighbors of the left-hand node i in this graph, which can be computed by a poly(n)-size DNF. The unique neighbor property guarantees that the xor of any $t \leq k = \log^c n$ output bits will be unbiased. And this, by the so-called Vazirani xor lemma, implies that any k output bits are uniformly distributed over $\{0,1\}^k$, concluding our overview.

We note that the use of unique-neighbor expanders to construct distributions with bounded independence is not new. It appears for example in the work by Mossel, Shpilka, and Trevisan [MST06] which studies the complexity of small-bias distributions, and shows that there are such distributions with non-trivial stretch computable in NC^0.

More related work A result (we already mentioned briefly) by Applebaum, Ishai, Kushilevitz [AIK06] shows, under standard assumptions, that there are pseudorandom distributions computable by NC^0 circuits. Their result is obtained via a generic transformation that turns a distribution D into another "padded" distribution D' that is computable in NC^0 and at the same time maintains interesting properties, such as pseudorandomness (but not stretch). The techniques in [AIK06] do not seem to apply to distributions such as $(x, \sum_i x_i)$ (Theorem 1.1), and they destroy stretch, which prevents them from obtaining Theorem 1.5 (regardless of the stretch of the original generator, the techniques in [AIK06] always produce a generator with sublinear stretch).

Organization In §2 we prove theorems 1.1 and 1.4, while in §3 we prove Theorem 1.5. In §4 we conclude and summarize a few open problems.

2 Generating $(x, \sum_i x_i)$

In this section we prove theorems 1.1 and 1.4. We start with the first.

Theorem 1.1. (Restated.) *There are explicit* AC^0 *circuits* $C : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n \times \{0,1,\ldots,n\}$ *of size* $\mathrm{poly}(n)$ *and depth* $O(1)$ *whose output distribution has statistical distance* $\leq 2^{-n}$ *from the distribution* $(x, \sum_i x_i) \in \{0,1\}^n \times \{0,1,\ldots,n\}$ *for random* $x \in \{0,1\}^n$.

First, we note that the statistical distance can be made 2^{-n^c} for an arbitrary constant c at the price of having the size of the circuit be a polynomial depending on c (choose larger ℓ in §A). Second, an immediate corollary of Theorem 1.1 is the ability to generate $(x, f(x))$ for any symmetric function f, e.g., majority.

Corollary 2.1. *For every symmetric function* $f : \{0,1\}^n \to \{0,1\}$ *there are explicit* AC^0 *circuits* $C : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^{n+1}$ *of size* $\mathrm{poly}(n)$ *and depth* $O(1)$ *whose output distribution have statistical distance* $\leq 2^{-n}$ *from the distribution* $(x, f(x)) \in \{0,1\}^{n+1}$ *for random* $x \in \{0,1\}^n$.

The proof of Theorem 1.1 relies on the following result by Matias and Vishkin [MV91] and Hagerup [Hag91] about generating random permutations of $[n]$. We think of a permutation of $[n]$ as represented by an array $A[1..n] \in [n]^n$.

Lemma 2.2 ([MV91, Hag91])**.** *There are explicit* AC^0 *circuits* $C : \{0,1\}^{\mathrm{poly}(n)} \to [n]^n$ *of size* $\mathrm{poly}(n)$ *and depth* $O(1)$ *whose output distribution has statistical distance* $\leq 2^{-n}$ *from the uniform distribution over permutations of* $[n]$.

Lemma 2.2 is obtained in [MV91] and Hagerup [Hag91, Theorem 3.9] in the context of PRAMs. Those works also achieve a nearly optimal number of processors, which appears to make the proofs somewhat technical. In §A we present a proof that uses the ideas in [MV91, Hag91] but is simpler and sufficient for our purposes.

Proof of Theorem 1.1. First, note for every $s \in \{0, \ldots, n\}$ there is a circuit C_s of size $\mathrm{poly}(n)$ and depth $O(1)$ whose output distribution is exponentially close to the uniform distribution over n-bit strings of weight s. To see this, run the circuit from Lemma 2.2 to produce array $A[1..n]$ containing a random permutation. The i-th output bit of C_s is set to 1 if and only if there is $j \leq s$ such that $A[j] = i$. In other words, we set to 1 exactly the first s elements of A. It is easy to see that this can be implemented using poly-size AC^0 circuits.

To generate $(x, \sum_i x_i)$, it remains to select the circuits C_s with the correct probability. To do this, recall that, given two n-bit integers a, b, we can efficiently determine if $a > b$ as follows (a_1 is the most significant digit):

$$a > b \Leftrightarrow (a_1 > b_1) \vee (a_1 = b_1 \wedge a_2 > b_2) \vee (a_1 = b_1 \wedge a_2 = b_2 \wedge a_3 > b_3) \ldots.$$

Now interpret n fresh random bits as an integer $z \in \{1, \ldots, 2^n\}$. Let circuit $D : \{0,1\}^n \to \{0, 1, \ldots, n\}$ output $s \in \{0, \ldots, n\}$ if and only if

$$\sum_{i0}^{s-1} \binom{n}{i} < z \leq \sum_{i=0}^{s} \binom{n}{i}.$$

To do this in parallel, we can for example construct circuits D_s each responsible of one output, and then take the OR of their outputs. By construction, the output distribution of D over uniform input equals the distribution $\sum_i X_i$ where X_i are independent uniform random bits.

The circuit claimed in the theorem first runs D to obtain s, then runs C_s to obtain $x \in \{0,1\}^n$, and outputs (x, s). □

We now prove a lower bound for NC^0 circuits.

Theorem 1.4. (Restated.) *Let* n *be odd and* $C : \{0,1\}^{d \cdot n} \to \{0,1\}^{n+1}$ *be a function such that each output bit* C_i *of* C *depends on only* d *inputs bits. Then the output distribution of* C *(for random input) has statistical distance at least* $1/2^{O(d)}$ *from* $(x, \mathrm{majority(x)}) \in \{0,1\}^{n+1}$ *for uniform* $x \in \{0,1\}^n$.

Proof. Let X be a random input to C, and let $C(X) = (Y, M)$ where $|Y| = n$, $|M| = 1$, and M is supposed to be the majority of Y. Let $R \in \{0,1\}^n$ be a uniform string.

First, $\Pr[M = 1] = 1/2$, for else the same argument as in the proof of Theorem 1.3 shows that $|\Pr[M = 1] - 1/2| \geq 1/2^d$, and, since $\Pr[\mathrm{majority}(R) = 1] = 1/2$, we get the desired statistical distance just looking at this bit. (Note that Pr[majority$(R) = 1] = 1/2$ because n is odd.)

Consider the distribution $Y' := Y|M = 1$, i.e., Y conditioned on the event $M = 1$. Let k be a large constant, independent from n, to be set later.

If Y' is k-wise independent, then by [DGJ+09] we have $\Pr[\text{majority}(Y') = 0] \geq \Pr[\text{majority}(R) = 0] - 1/4 = 1/4$, for a sufficiently large k. Therefore we obtain constant statistical distance via the test T that, given $(z, b), |z| = n, |b| = 1$, outputs 1 if and only if $b = 1$ and majority$(z) = 1$: on the one hand, $\Pr[(R, \text{majority}(R)) \in T] \geq 1/2$, while on the other hand $\Pr[(Y, M) \in T] = \Pr[(Y, M) \in T|M = 1] \cdot \Pr[M = 1] = \Pr[\text{majority}(Y') = 1] \cdot 1/2 \leq (1 - 1/4) \cdot 1/2$.

Otherwise, assume that Y' is not k-wise independent. So there are k bits $(Y'_{i_1}, Y'_{i_2}, \ldots, Y'_{i_k})$ whose distribution is not uniform over $\{0,1\}^k$. The distribution Y' is the same as the first n output bits of $C(X')$ where $X' := X|M = 1$ is the distribution on the input bits conditioned of yielding $M = 1$. Since $\Pr[M = 1] = 1/2$, X' is the uniform distribution over the input bits conditioned on the event that the d input bits M depends on are uniformly distributed over $2^d/2$ combinations. Even if the $k = O(1)$ bits $(Y'_{i_1}, Y'_{i_2}, \ldots, Y'_{i_k})$ depended on $k \cdot d$ uniform and independent bits and also on the d input bits M depends on, there must be an output combination $a \in \{0,1\}^k$ such that $\Pr[(Y'_{i_1}, Y'_{i_2}, \ldots, Y'_{i_k}) = a] \geq 1/2^k + 1/2^{k \cdot d} \cdot 2/2^d \geq 1/2^k + 1/2^{O(d)}$. On the other hand, a calculation shows that any $k = O(1)$ bits of the distribution $R|\text{majority}(R) = 1$ are equal to a with probability that approaches $1/2^k$ as n tends to infinity. So looking at these k bits we get distance $1/2^{O(d)}$. $\square$

3 Depth-efficient derandomization of AC^0

In this section we prove Theorem 1.5:

Theorem 1.5 (Depth-efficient derandomization of AC^0)**.** (Restated.) *The following holds for every d. Let $f : \{0,1\}^* \to \{0,1\}^*$ be computable by uniform randomized* AC^0 *circuits of* poly(n)*-size and depth d with error ϵ. Then f is computable by uniform randomized circuits of* poly(n)*-size and depth $d + 1$ with error $\epsilon + o(1)$ using $\leq (\log n)^{O(\log \log n)}$ random bits.*

First, we remark that previous derandomizations [AW89, Nis91] increase the depth by a constant factor, as opposed to adding 1 in our result, or else have poor seed length.

For our depth-efficient derandomization, we need the results by Bazzi, Razborov, and Braverman that polylog-wise distributions fool small AC^0 circuits.

Theorem 3.1 ([Bra09])**.** *For every d there is c such that the following holds for every m. Let C be a boolean circuit of depth d and size m. Let D be a $(\log^c m)$-wise distribution. Then*

$$|Pr_U[C(U) = 1] - Pr[C(D) = 1]| \leq 1/m,$$

where U is the uniform distribution.

We also need the following result by Guruswami, Umans, and Vadhan.

Theorem 3.2 (Theorem 3.3 in [GUV09])**.** *For any integers s, m, and q, where q is a power of 2, there are explicit graphs $G : [q]^s \times [q] \to [q]^{m+1}$ such that any set $S \subseteq [q]^s$ of size at most 2^m has at least $|S| \cdot (q - (s-1)m)$ neighbors.*

The following is our efficient constructions of k-wise independent distributions.

Theorem 3.3. *For every n and $d \leq \log n$ there is an explicit circuit $C : \{0,1\}^\ell \to \{0,1\}^n$ such that:*
(i) each output bit of C is a DNF of size $n^{O(d)}$,
(ii) the distribution $C(U)$ for random $U \in \{0,1\}^s$ is $(\log^d n)$-wise independent,
(iii) $\ell \leq (\log n)^{O(d \cdot \log\log n)}$.

Proof. Let c be a sufficiently large, even constant to be set later. Let q be a power of 2 such that

$$c \cdot d \cdot \log n \leq q \leq 2c \cdot d \cdot \log n.$$

Choose

$$s := \left\lceil \frac{\log n}{\log q} \right\rceil,$$

and

$$m := \left\lceil \frac{q}{4} \cdot \frac{\log q}{\log n} \right\rceil \leq \left\lceil \frac{2c \cdot d \log n}{4} \cdot \frac{\log q}{\log n} \right\rceil = \frac{c \cdot d \cdot \log q}{2}.$$

Let G be the graph given by Theorem 3.2 for the above parameters. The i-th output bit of C is the parity of the neighbors of the i-th left-hand node in G. First, note that the numbers of left-hand nodes in G is

$$q^s \geq q^{(\log n)/\log q} \geq n,$$

so the output length of C is as desired. Also, each left-hand node has $\leq 2c \cdot d \log n$ neighbors by our choice of q. The corresponding parity can be computed in brute-force by a DNF of size $n^{O(d)}$. This verifies (i).

The number of right-hand side nodes is the input length ℓ of C. Using our above bounds on m and q, and the assumption that $d \leq \log n$, we have

$$\ell = q^{m+1} \leq (2c \cdot d \cdot \log n)^{O(d \cdot \log q)} \leq (\log n)^{O(d \cdot \log\log n)}.$$

This verifies (iii).

To see (ii), it is enough to observe that every $S \subseteq [q]^s$ of size $|S| \leq \log^d n$ has at least one unique neighbor. This unique neighbor guarantees that the parity of the bits corresponding to S (which recall are each defined to be the parity of their neighbors) is unbiased. Therefore, by the so-called Vazirani XOR lemma (cf. [Gol97]) the output distribution is $(\log^d n)$-wise independent.

Indeed, note that the expansion property of the graph holds for sets of size up to

$$2^m \geq \exp\left(\frac{q}{4} \cdot \frac{\log q}{\log n}\right) \geq \exp\left(\frac{c \cdot d \cdot \log n}{4} \cdot \frac{\log\log n}{\log n}\right) \geq \exp\left(\frac{c \cdot d \cdot \log\log n}{4}\right)$$
$$\geq \log^d n,$$

for a sufficiently large c.

Hence, by the expansion property, S has at least $|S|(q-(s-1)m)$ neighbors. To guarantee a unique neighbor, we need more than $|S|q/2$ neighbors. Equivalently, $(s-1)m < q/2$. By our choice of s and m we have

$$(s-1)m \leq \left(\frac{\log n}{\log q} + 1 - 1\right) m = \frac{\log n}{\log q} \cdot \frac{q \log q}{4 \log n} = \frac{q}{4} < \frac{q}{2},$$

concluding the proof. □

Combining the above two theorems yields Theorem 1.5. Suitable improvements in theorems 3.2 and 3.1 would make the number of random bits in the conclusion of Theorem 1.5 match the state-of-the-art [Nis91].

We also draw a connection to the long-standing challenge of finding an algorithm that, given a DNF, computes an additive approximation to the number of satisfying assignments (see, e.g., [Tre04]). The above techniques show that the latter problem is polynomial-time reducible to the problem of computing an additive approximation to the number of satisfying assignments of a given depth-3 circuit with only $(\log n)^{O(\log\log n)}$ variables.

4 Conclusion

This paper makes a step towards a systematic study of the complexity of generating distributions. Many open problems remain. What is the complexity of generating $(x, \text{majority}(x)) \in \{0,1\}^{n+1}$? We have proved the existence of AC^0 distributions that are exponentially close, while NC^0 distributions are at distance $\epsilon > 0$ independent from n.

What is the complexity of generating the uniform distribution over sets of a fixed size $\ell \leq n$? Our techniques yield the same bounds for this problem too. A sufficiently strong lower bound for this problem would have interesting consequences for data structures.

Finally, what is the complexity of generating distributions that look random to small AC^0 circuits? We have shown how to efficiently generate k-wise independent distributions, and obtained a new derandomization of depth-d circuits by circuits of depth $d+1$ that use a number of random bits close to the best known polynomial-time derandomization [Nis91]. Still, the parameters can be improved. What is the complexity of generating the output distribution of Nisan's generator [Nis91]? Efficient generation of such distributions allows one to bypass the computational overhead that is intrinsic in the Nisan-Wigderson generators.

Acknowledgments. I am very grateful to Rajmohan Rajaraman and Ravi Sundaram for extensive collaboration on this project.

References

[ABI86] Noga Alon, László. Babai, and Alon Itai. A fast and simple randomized algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986. 4

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. 1, 4, 5

[AW89] Miklos Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constand-depth circuits. *Advances in Computing Research - Randomness and Computation*, 5:199–223, 1989. 7

[Bab87] László Babai. Random oracles separate PSPACE from the polynomial-time hierarchy. *Inform. Process. Lett.*, 26(1):51–53, 1987. 1

[BL87] Ravi B. Boppana and J. C. Lagarias. One-way functions and circuit complexity. *Inform. and Comput.*, 74(3):226–240, 1987. 1

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. on Computing*, 13(4):850–864, November 1984. 3

[BMRS02] Harry Buhrman, Peter Bro Miltersen, Jaikumar Radhakrishnan, and Venkatesh Srinivasan. Are bitvectors optimal? *SIAM J. Comput.*, 31(6):1723–1744, 2002. 2

[Bra09] Mark Braverman. Poly-logarithmic independence fools AC^0 circuits. In *24th Conference on Computational Complexity (CCC)*. IEEE, 2009. 4, 7

[CG89] Benny Chor and Oded Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, 1989. 4

[DGJ$^+$09] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. Bounded independence fools halfspaces. In *50th Symposium on Foundations of Computer Science (FOCS)*, 2009. 3, 7

[Gol97] Oded Goldreich. Three XOR lemmas — an exposition. Available from `http://www.wisdom.weizmann.ac.il/~oded/`, November 1997. 8

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), 2009. 4, 8

[GV04] Dan Gutfreund and Emanuele Viola. Fooling parity tests with parity gates. In *8th International Workshop on Randomization and Computation (RANDOM)*, pages 381–392. Springer, 2004. 4

[Hag91] Torben Hagerup. Fast parallel generation of random permutations. In *18th Colloquium on Automata, Languages and Programming (ICALP)*, pages 405–416, 1991. 1, 5, 6, 12

[Hås87] Johan Håstad. *Computational limitations of small-depth circuits.* MIT Press, 1987. 1

[IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, Fall 1996. 1

[LVW93] Michael Luby, Boban Veličković, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *2nd Israeli Symposium on Theoretical Computer Science (ISTCS)*, pages 18–24, 1993. 4

[MST06] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On epsilon-biased generators in nc^0. *Random Struct. Algorithms*, 29(1):56–81, 2006. 5

[MV91] Yossi Matias and Uzi Vishkin. Converting high probability into nearly-constant time-with applications to parallel hashing (extended abstract). In *23rd ACM Symposium on Theory of Computing (STOC)*, pages 307–316, 1991. 1, 5, 6, 12

[Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991. 4, 7, 9

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Computer & Systems Sciences*, 49(2):149–167, 1994. 4

[Pag01a] Rasmus Pagh. Low redundancy in static dictionaries with constant query time. *SIAM J. Comput.*, 31(2):353–363, 2001. 2

[Pag01b] Rasmus Pagh. On the cell probe complexity of membership and perfect hashing. In *33rd Annual Symposium on Theory of Computing (STOC)*, pages 425–432. ACM, 2001. 2

[Păt08] Mihai Pătraşcu. Succincter. In *49th Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2008. 2

[Tre04] Luca Trevisan. A note on approximate counting for k-dnf. In *7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (RANDOM)*, volume 3122 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 2004. 9

[Vio05] Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *20th Annual Conference on Computational Complexity (CCC)*, pages 183–197. IEEE, 2005. 1

[Vio07] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM Journal on Computing*, 36(5):1387–1403, 2007. 4

[Vio09] Emanuele Viola. Bit-probe lower bounds for succinct data structures. In *41th Annual Symposium on the Theory of Computing (STOC)*. ACM, 2009. 2

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE. 3

A Generating random permutations in AC^0

In this section we give a simple proof of the following lemma.

Lemma 2.2 ([MV91, Hag91])**.** (Restated.) *There are explicit* AC^0 *circuits* $C : \{0,1\}^{\mathrm{poly}(n)} \to [n]^n$ *of size* $\mathrm{poly}(n)$ *and depth* $O(1)$ *whose output distribution has statistical distance* $\le 2^{-n}$ *from the uniform distribution over permutations of* $[n]$*.*

The main technique is known as "dart throwing:" we view the input random bits as random pointers $p_1, p_2, \ldots, p_n$ into $m \gg n$ cells. We then write i in the p_i-th cell (empty cells get "$*$"). If there are no collisions, the ordering of $[n]$ in the cells gives a random permutation of $[n]$. However, it is not clear how to explicitly write out this permutation using small depth, because to determine the image of i one needs to count how many cells before p_i are occupied, which cannot be done in small depth.

The key insight of Matias and Vishkin [MV91] (see also [Hag91]) is to view the cells as representing the permutation in a different format, one from which we can explicitly write out the permutation in small depth. The format is known as the canonical form for the cyclic notation. We now briefly review it closely following [MV91]. Just like the standard format, the alternative format represents a permutation via an array $A[1..n]$ whose entries contain all the elements $[n]$. However, rather than thinking of $A[i]$ as the image of i, we think of the entries of A as listing the cycles of the permutation in order. Each cycle is listed starting with its smallest element, and cycles are listed in decreasing order of the first element in the cycle. This format allows for computing the permutation efficiently: the image of i is the element to the right of i in A, unless the latter element is the beginning of a new cycle, in which case the image of i is the first element in the cycle containing i. Identifying the first element of a cycle is easy, because they are smallest than any element preceding them in A. One can now verify that computing the image of i can be done by circuits of size $\mathrm{poly}(n)$ and depth $O(1)$.

The benefit of this format is that it works even if the array A has $m \gg n$ cells, of which $m - n$ are empty and marked by "$*$."

To conclude the proof of the lemma, generate ℓ uniform and independent sets of pointers $p_1^i, \ldots, p_n^i$, $i = 1, \ldots, \ell$, where each pointer has range $[m]$ for m the smallest power of 2 larger than $2n^2$ (thus each pointer can be specified by $\log m$ bits).

If there exists i such that the pointers $p_1^i, \ldots, p_\ell^i$ are all distinct (i.e., there are no collisions), then run the above algorithm on the output corresponding to the first such i. This results in a random permutation.

Since the pointers are chosen independently, the probability that there is no such i is

$$\Pr[\forall i \exists j, k \le n : p_j^i = p_k^i] = \Pr[\exists j, k \le n : p_j^1 = p_k^1]^\ell \le (1/2)^\ell.$$

Choosing $\ell := n$ proves the lemma.