Derandomizing Approximation Algorithms for Hard Counting Problems

Michael Luby
International Computer Science Institute, Berkeley, CA
and University of California at Berkeley *

TR-95-069

December 1995

*Research supported in part by National Science Foundation operating grant CCR-9304722 and NCR-9416101, and United States-Israel Binational Science Foundation grant No. 92-00226

1 Introduction

This paper is a (biased) survey of some work in derandomization of randomized algorithms. Perhaps the most famous open problem in Computer Science is whether or not **NP** is equal to **P**, i.e., are efficient non-deterministic algorithms more powerful than efficient deterministic algorithms. An analogous question is whether nor not **RP** is equal to **P**, i.e., are efficient randomized algorithms more powerful than efficient deterministic algorithms.

These two questions formally are quite similar, as the only difference between the definitions of **NP** and **RP** are that for an **NP** language L, there is only required to be one witness for each $x \in L$, whereas for **RP** a constant fraction of strings are required to be witnesses.

Nevertheless, it is my belief that **NP** is more powerful than **P**, and on the other hand **RP** is equal to **P**. Paradoxically, there is some hope that an eventual proof that shows that **NP** is not equal to **P** will be strong enough to also show that **RP** is equal to **P**.

Definition (function ensemble): Let $f : \{0,1\}^{t(n)} \rightarrow \{0,1\}^{\ell(n)}$ denote a *function ensemble*, where $t(n)$ and $\ell(n)$ are polynomial in n and where f with respect to n is a function mapping $\{0,1\}^{t(n)}$ to $\{0,1\}^{\ell(n)}$. ♣

Definition (P-time function ensemble): Call $f : \{0,1\}^{t(n)} \times \{0,1\}^{\ell(n)} \rightarrow \{0,1\}^{m(n)}$ is a $T(n)$-*time function ensemble* if f is a function ensemble and there is a Turing machine such that, for all $x \in \{0,1\}^{t(n)}$, for all $y \in \{0,1\}^{\ell(n)}$, $f(x,y)$ is computable in time $T(n)$. Call f is a **P**-*time function ensemble* if $T(n) = n^{\mathcal{O}(1)}$. ♣

Definition (language): Let $L : \{0,1\}^n \rightarrow \{0,1\}$ be a function ensemble. One can view L as a *language*, where, for all $x \in \{0,1\}^n$, $x \in L$ if $L(x) = 1$ and $x \notin L$ if $L(x) = 0$. ♣

In the following, definitions of a variety of complexity classes with respect to a language L as just defined are made.

Definition (P): Say that $L \in \mathbf{P}$ if L is a **P**-time function ensemble. ♣

Definition (NP): Say that $L \in \mathbf{NP}$ if there is a **P**-time function ensemble $f : \{0,1\}^n \times \{0,1\}^{\ell(n)} \rightarrow \{0,1\}$ such that for all $x \in \{0,1\}^n$,

$$\begin{array}{lll} x \in L & \text{implies} & \Pr_{Y \in_U \{0,1\}^{\ell(n)}}[f(x,Y) = 1] > 0. \\ x \notin L & \text{implies} & \Pr_{Y \in_U \{0,1\}^{\ell(n)}}[f(x,Y) = 1] = 0. \end{array}$$

♣

Definition (RP): Say that $L \in \mathbf{RP}$ if there is a constant $\epsilon > 0$ and a **P**-time function ensemble $f : \{0,1\}^n \times \{0,1\}^{\ell(n)} \rightarrow \{0,1\}$ such that for all $x \in \{0,1\}^n$,

$$\begin{array}{lll} x \in L & \text{implies} & \Pr_{Y \in_U \{0,1\}^{\ell(n)}}[f(x,Y) = 1] \geq \epsilon. \\ x \notin L & \text{implies} & \Pr_{Y \in_U \{0,1\}^{\ell(n)}}[f(x,Y) = 1] = 0. \end{array}$$

♣

There are a few philosophical reasons to want to show that $\mathbf{RP} = \mathbf{P}$, e.g., does randomization help in the general context of efficient computation? There are also some practical reasons, e.g., in practice if one uses a Monte Carlo algorithm then there should be great concern about where the source of random bits comes from. In practice, what one typically does is to take a very small number of random bits and stretch them into a large number using a simple pseudorandom generator, e.g., a linear congruential generator, and then use these bits as the source of randomness for the Monte Carlo simulation. The problem in practice is that there is no guarantee that these pseudorandom bits are random enough to make the Monte Carlo simulation behave as if though they were truly random. In practice sometimes the answer turns out to be that this is good enough, sometimes it has turned out to be not good enough, and in many cases the answer is not known. One of the motivations behind the work on derandomization of randomized algorithms is to put this entire question on a much firmer scientific footing, i.e., to be able to say authoritatively which pseudorandom generators can be provably used for which Monte Carlo algorithms.

1.1 Randomness and Pseudorandomness

The notion of randomness tests for a string evolved over time: from set-theoretic tests to enumerable [24, Kolmogorov], recursive and finally limited time tests. There were some preliminary works that helped motivate the concept of a pseudorandom generator including [38, Shamir].

[9, Blum and Micali] introduce the fundamental concept of a pseudorandom generator that is useful for cryptographic (and other) applications, and gave it the significance it has today by providing the first provable construction of a pseudorandom generator based on the conjectured difficulty of a well-known and well-studied computational problem, the discrete log problem. [41, Yao] introduces the now standard definition of a pseudorandom generator, and shows an equivalence between the this definition and the next bit test introduced in [9, Blum and Micali]. The standard definition of a pseudorandom generator introduced by [41, Yao] is based on the fundamental concept of computational indistinguishability introduced previously in [18, Goldwasser and Micali]. [41, Yao] also shows how to construct a pseudorandom generator from any one-way permutation.

Another important observation of [41, Yao] is that a pseudorandom generator can be used to reduce the number of random bits needed for any probabilistic polynomial time algorithm, and this shows how to perform a deterministic simulation of any polynomial time probabilistic algorithm in subexponential time based on a pseudorandom generator. The results on deterministic simulation were subsequently generalized in [10, Boppana and Hirschfeld].

The robust notion of a pseudorandom generator, due to [9, Blum and Micali], [41, Yao], should be contrasted with the classical methods of generating random looking bits as described in, e.g., [23, Knuth]. In studies of classical methods, the output of the generator is considered good if it passes a particular set of standard statistical tests. The linear congruential generator is an example of a classical method for generating random looking bits that pass a variety of standard statistical tests. However, [11, Boyar] and [25, Krawczyk]

show that there is a polynomial time statistical test which the output from this generator does not pass.

2 Circuits

A convenient way to view computation is via boolean circuits.

Definition ($C_{\ell(n)}^{d(n),s(n)}$): For all $n \in \mathcal{N}$, let $C_{\ell(n)}^{d(n),s(n)}$ be the set of all circuits with $\ell(n)$ boolean input variables $z = \langle z_1, \ldots, z_{\ell(n)} \rangle$ of depth $d(n)$ with a total of at most $s(n)$ gates. A circuit $C \in C_{\ell(n)}^{d(n),s(n)}$ consists of "and"-gates and "or"-gates, where each gate is allowed unbounded fan-in. C consists of $d(n)$ levels of gates, where all gates at a given level are the same type. All the gates at level 1 have as inputs any mixture of variables and their negations. For all $i \in \{2, \ldots, d(n)\}$, all gates at level i receive their inputs from the gates at level $i-1$. The set of gates at level $d(n)$ are considered to be the output of C. ♣

For example, let $f : \{0,1\}^n \times \{0,1\}^{\ell(n)} \rightarrow \{0,1\}$ be the **P**-time function ensemble associated with an **RP** language L. One can think of f as a family of boolean circuits as follows. For each n, there are 2^n circuits in the family with $\ell(n)$-bit inputs, one circuit for each of the 2^n possible values of $x \in \{0,1\}^n$. The circuit C_x associated with the input x computes the value of $f(x,y)$ on input $y \in \{0,1\}^{\ell(n)}$. The circuit C_x can be derived from the description of f and from x. Since f is a **P**-time function ensemble, without loss of generality one can say that C_x consists of at most a polynomial number $p(n)$ of alternating levels of "and" and "or" gates, with a single output gate at the bottom level. The property that C_x has is that if $x \in L$ then $\Pr_Y[C_x(Y) = 1] \geq 1/2$ and if $x \notin L$ then $\Pr_Y[C_x(Y) = 1] = 0$, where $Y \in_R \{0,1\}^{\ell(n)}$. One can view this circuit family for L as a subfamily of $C_{\ell(n)}^{p(n),p(n)}$.

In terms of circuits, the approach to derandomization explored in this survey is to construct a pseudorandom generator g that can be used as the source of randomness for all $C \in \mathcal{C}$. This approach was initiated by [9, Blum and Micali] and [41, Yao].

Definition (ϵ**-pseudorandom generator for a circuit family** $\mathcal{C}$): Let $g : \{0,1\}^{r(n)} \rightarrow \{0,1\}^n$ be a **P**-time function ensemble, where $r(n) < n$. Let $\mathcal{C}$ be an infinite family of circuits, and let $C \in \mathcal{C}$ be a circuit with n inputs. The distinguishing probability of C for g is

$$\delta = |\Pr_Y[C(Y) = 1] - \Pr_Z[C(g(Z)) = 1]|,$$

where $Y \in_R \{0,1\}^n$ and $Z \in_R \{0,1\}^{r(n)}$, in which case g is said to ϵ-approximate C for any $\epsilon > \delta$. Say that g is a ϵ-pseudorandom generator for $\mathcal{C}$ if g ϵ-approximates C for all $C \in \mathcal{C}$. Say that $\mathcal{C}$ has time-success ratio $\mathbf{S}(n)$ for g if the minimum over all circuits $C \in \mathcal{C}$ with n inputs of the ratio of the number of gates in C divided by the distinguishing probability of C is at least $\mathbf{S}(n)$. ♣

Given g, the derandomization for the **RP** circuit family described above is straightforward: For any $x \in \{0,1\}^n$, one can approximate the fraction of the $2^{\ell(n)}$ inputs on which C_x outputs 1 as follows. Let $g : \{0,1\}^{r(n)} \rightarrow \{0,1\}^{\ell(n)}$ be a 1/2-pseudorandom generator

for $C_{\ell(n)}^{p(n),p(n)}$. For all $z \in \{0,1\}^{r(n)}$, compute $C_x(g(z))$, and if any of the answers is 1 then conclude that $x \in L$, and otherwise conclude that $x \notin L$. Note that by the properties of g this procedure is guaranteed to correctly classify x with respect to membership in L. The smaller $r(n)$ is in relationship to n the stronger the pseudorandom generator g. For example, if $r(n)$ is $\mathcal{O}(\log(n))$ then the entire procedure runs in deterministic polynomial time, showing that $\mathbf{RP} = \mathbf{P}$.

One way to view the above approach is to find a small set S of $\ell(n)$-bit strings such that the average value of $C_x(y)$ over $y \in S$ is close to the average value of $C_x(y)$ over all $y \in \{0,1\}^{\ell(n)}$. In the above, S can be viewed as the set $\{g(z) : z \in \{0,1\}^{r(n)}\}$. The other important ingredient to the derandomization is that the set S can be efficiently enumerated. This is true in the above approach because g is a $\mathbf{P}$-time function ensemble, and thus S can be enumerated simply by sequencing through all $z \in \{0,1\}^{r(n)}$ and computing $g(z)$.

Suppose one is only interested in the property that S be small and drop the (crucial) requirement that S is easy to enumerate, Then, it is not hard to see that there is such a small set S. To see this, suppose that one chooses randomly a set S of size $n+1$. If $x \notin L$ then, independent of S, x is correctly classified. On the other hand, if $x \in L$, then x is incorrectly classified with probability at most $2^{-(n+1)}$. Since there are at most 2^n such x, the probability that there is an $x \in \{0,1\}^n$ that is correctly classified is at most $1/2$. Thus, there is a set S of size $n+1$ that correctly classifies all $x \in \{0,1\}^n$ with respect to membership in L. This is the approach that [1, Adleman] took to show that $\mathbf{RP} \subseteq \mathbf{P/poly}$, i.e., in a non-uniform sense randomization is no more powerful than deterministic computation. The problem with this approach is that there is no clue of how to efficiently generate the set S in deterministic polynomial time. Thus, the uniform version of the "$\mathbf{RP} = \mathbf{P}$?" question is far from resolved.

3 Derandomizing Particular Algorithms

In some applications, the goal is to completely derandomize the algorithm to produce a deterministic solution. However, in other cases the basic goal is only to drastically reduce the number of random bits used by the randomized algorithms, e.g., reduce from $\mathcal{O}(n)$ bits to $\mathcal{O}(\log(n))$ bits. The philosophical and practical import of this is that it is hard in practice to produce high quality independent random bits at a high rate.

3.1 Pairwise Independence

One of the key ideas in derandomizing randomized algorithms is the observation that sometimes it is the case that the randomized algorithms works just as well if there is not full independence between the random variables. One special but important case is when pairwise independence between the random variables suffices.

Consider a set of random variables indexed by a set U, i.e., $\{Z_i : i \in U\}$, where $Z_i \in T$. For finite $t = |T|$, a uniform distribution assigns $\Pr[Z_x = \alpha] = 1/t$, for all $x \in U$, $\alpha \in T$. If this distribution were furthermore pairwise independent, one would have: for all $x \neq y \in U$,

for all $\alpha, \beta \in T$,

$$\Pr[Z_x = \alpha, Z_y = \beta] = \Pr[Z_x = \alpha] \cdot \Pr[Z_y = \beta] = 1/t^2.$$

This is not the same as complete independence, as evidenced by the following set of three pairwise-independent variables ($U = \{1,2,3\}$, $T = \{0,1\}$, $t = 2$):

s	Z_1	Z_2	Z_3
00	0	0	0
01	0	1	1
10	1	0	1
11	1	1	0

Each row s can be thought of as a function $h_s : U \to T$. Let $\mathcal{S}$ be the index set for these functions, where in this case $\mathcal{S} = \{0,1\}^2$. For all $x \neq y \in U$, for all $\alpha, \beta \in T$,

$$\Pr_{s \in_R \mathcal{S}}[h_s(x) = \alpha \wedge h_s(y) = \beta] = 1/4 = 1/t^2$$

(Notice in particular that $\Pr_{s \in_R \mathcal{S}}[h_s(x) = h_s(y)] = 1/2 = 1/t$.) Any set of functions satisfying this condition is a 2-universal family of hash functions.

Definitions and explicit constructions of 2-universal hash functions were first given by [40, Carter and Wegman] . One simple way to construct a general family of hash functions mapping $\{0,1\}^n \to \{0,1\}^n$ is to let $\mathcal{S} = \{0,1\}^n \times \{0,1\}^n$, and then for all $s = (a,b) \in \mathcal{S}$, for all $x \in \{0,1\}^n$ define $h_s(x) = ax + b$, where the arithmetic operations are with respect to the finite field GF[2^n]. Thus, each h_s maps $\{0,1\}^n \to \{0,1\}^n$ and $\mathcal{S}$ is the index set of the hash functions. For each $s = (a,b) \in \mathcal{S}$, one can write:

$$\begin{pmatrix} h_s(x) \\ h_s(y) \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

When $x \neq y$, the matrix is non-singular, so that for any $x, y \in \{0,1\}^n$, the pair $(h_s(x), h_s(y))$ takes on all 2^{2n} possible values (as s varies over all $\mathcal{S}$). Thus if s is chosen uniformly at random from $\mathcal{S}$, then $(h_s(x), h_s(y))$ is also uniformly distributed.

One can view $\mathcal{S}$ as the set of points in a sample space on the set of random variables $\{Z_x : x \in \{0,1\}^n\}$ where $Z_x(s) = h_s(x)$ for all $s \in \mathcal{S}$. With respect to the uniform distribution on $\mathcal{S}$, these random variables are pairwise independent, i.e., for all $x \neq y \in \{0,1\}^n$, for all $\alpha, \beta \in \{0,1\}^n$

$$\Pr_{s \in_R \mathcal{S}}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] = \Pr_{s \in_R \mathcal{S}}[Z_x(s) = \alpha] \cdot \Pr_{s \in_R \mathcal{S}}[Z_y(s) = \beta] = 1/2^{2n}.$$

To obtain a hash function that maps to $k < n$ bits, we can still use $\mathcal{S}$ as the function index family: The value of of the hash function indexed by s on input x is obtained by computing $h_s(x)$ and using the first k bits. The imporant properties of these hash functions are:

- Pairwise independence.
- Succinctness – each function can be described as a $2n$-bit string. Therefore, randomly picking a function index requires only $2n$ random bits.
- The function $h_s(x)$ can easily be computed (in LOGSPACE, for instance) given the function index s and the input x.

In the sequel, unless otherwise specified, references to pairwise independent hash functions refer to this construction, and $\mathcal{S}$ denotes the set of indices for the hash family.

3.2 Applications to Particular Algorithms

The original applications described in [40, Carter and Wegman] were applications to hashing. Subsequently 2-universal hashing has been applied in surprising ways to a rich variety of problems. For a survey of some of these applications, see [31, Luby and Wigderson].

Consider, for example, the MAXCUT problem: given a graph $G = (V, E)$, find a two-coloring of the vertices $\chi : V \to \{0, 1\}$ so as to maximize $c(\chi) = |\{(x, y) \in E : \chi(x) \neq \chi(y)\}|$. The following is a description of a solution to this problem that is guaranteed to produce a cut where at least half the edges cross the cut.

If the vertices are colored randomly (0 or 1 with probability 1/2) by choosing χ uniformly from the set of all possible $2^{|V|}$ colorings, then:

$$\mathrm{E}[c(\chi)] \quad = \quad \sum_{(x,y)\in E} \Pr[\chi(x) \neq \chi(y)] \quad = \quad \frac{|E|}{2}$$

Thus, there must always be a cut of size at least $\frac{|E|}{2}$. Let $\mathcal{S}$ be the index set for the hash family mapping $V \to \{0, 1\}$. Since the summation above only requires the coloring of vertices to be pairwise-independent, it follows that $\mathrm{E}[c(h_s)] = \frac{|E|}{2}$ when $s \in_R \mathcal{S}$. Since $|\mathcal{S}| = |V|^2$, one can deterministically try h_s for all $s \in \mathcal{S}$ in polynomial time (even in the parallel complexity class NC), and for at least one $s \in \mathcal{S}$, h_s defines a partition of the nodes where at least $\frac{|E|}{2}$ edges cross the partition.

This derandomization approach was developed and discussed in general terms in the series of papers [13, Chor and Goldreich], [27, Luby], [5, Alon, Babai and Itai]. There, the approach was applied to derandomize algorithms such as witness sampling, a fast parallel algorithm for finding a maximal independent set, and other graph algorithms. This work was extended to more efficient parallel approaches in [28, Luby], and generalized in [8, Berger and Rompel] and [32, Motwani, Naor and Naor] to apply to parallel solutions of other combinatorial problems.

Other examples include the use of five-wise independent random variables to choose the pivot elements for quicksort while still maintaining the $\mathcal{O}(n \log(n))$ running time [22, Karloff and Raghavan]. Extending the work of [22, Karloff and Raghavan], [33, Mulmuley]

reanalyzes many classical computational geometry problems, including trapezoidal triangulations, convex hulls, Voronoi diagrams, and hidden surface removal. The new analysis shows that only limited independence between the random variables suffices for the randomized algorithm.

Other ideas that have been used to derandomize particular algorithms include using expander graphs, and combining expander graphs with pairwise independence techniques. An example of this approach is [21, Karger and Motwani].

4 Derandomizing Depth Two Circuits

In this section depth two unbounded fan-in boolean circuits are considered. The circuit subfamily of $\mathcal{C}_n^{2,m(n)+1}$ with a first layer of $m = m(n)$ "and" gates all feeding into a single output "or" gate can be viewed as a formula F in disjunctive normal form (DNF) on n variables with m clauses. Let t be the maximum length of a clause in the formula, and let $\Pr[F]$ denote the probability that a random, independent and uniformly chosen truth assignment to the variables satisfies F. The problem of computing $\Pr[F]$ exactly is known to be #P-complete [39, Valiant]. On the other hand, for many applications a good estimate of $\Pr[F]$ is all that is needed.

[29, Luby and Veličković] introduces several ideas which can be combined with known results to obtain approximation algorithms for the DNF problem. The first idea is that of sunflower reduction in a way similar to the way that [37, Razborov] uses it to prove exponential lower bounds for the size of the smallest monotone circuit for finding the biggest clique in a graph. Given a DNF formula F, one looks for a large collection of clauses which form a sunflower, i.e., the intersection of any two distinct clauses in this collection is the same. Then, all the clauses in this collection are replaced by their common pairwise intersection, thus obtaining another formula which has probability of being satisfied close to that of F.

This procedure is repeated until no large sunflowers can be found, obtaining a new formula F'. A theorem of [14, Erdös and Rado] implies that at the end of this procedure there are not too many clauses in F'. Because F' is so small, it turns out to be easier to approximate $\Pr[F']$, and because of the properties of the reduction this approximation is also a good approximation of $\Pr[F]$. This reduction can be combined with the algorithm from [36, Nisan and Wigderson] (using the improvement based on an observation from [30, Luby, Veličković and Wigderson]) to produce a polynomial time ϵ-approximation for $\Pr[F]$ when $t = \mathcal{O}(\log^{1/8}(nm))$ and ϵ is not too small.

The second approach relies on special constructions of small probability distributions. Given a distribution $\mathcal{D}$ let $\Pr_{\mathcal{D}}[F]$ denote the probability that F is satisfied by a truth assignment chosen according to $\mathcal{D}$. The goal is to find a distribution $\mathcal{D}$ with a small sample space such that $\Pr_{\mathcal{D}}[F]$ is close to $\Pr[F]$. Then, $\Pr_{\mathcal{D}}[F]$ can be calculated by exhaustive consideration of the points in the space. An easy counting argument shows that such a distribution exists. The results of [29] can be viewed as progress towards finding a polynomial time construction of such a distribution.

Recall that a probability distribution $\mathcal{D}$ on the truth assignments is k-wise independent if any variable is equally likely to be 0 or 1 independently of the value of any other $k-1$ variables. [29, Luby and Veličković] shows that if t is bounded by a constant and $k = c\log(1/\epsilon)$ for some constant c then for any probability distribution $\mathcal{D}$ which is k-wise independent $\Pr_{\mathcal{D}}[F]$ is a good approximation of $\Pr[F]$. The proof works if $\mathcal{D}$ is only k-wise almost independent in a certain technical sense. This combined with the results of [34, Naor and Naor] and [6, Alon, Goldreich, Håstad and Peralta], which produce such distributions with small sample spaces, yields a polynomial time approximation algorithm for $\Pr[F]$ if t is bounded by a constant, a result previously obtained by [3, Ajtai and Wigderson] by different means.

The main contribution of [29, Luby and Veličković] consists of the coloring algorithm. A *proper k-coloring* of F is a coloring of the variables of F using at most k colors such that no clause of F contains two variables of the same color. For all $i \in \{1,\ldots,k\}$, let $\mathcal{D}_i$ be a probability distribution on the variables in color class i such that there is ℓ-wise independence between the variables, and let $\mathcal{D} = \times_{i\in\{1,\ldots,k\}}\mathcal{D}_i$ be the product distribution defining a distribution on all variables of F. It is shown that $|\Pr_{\mathcal{D}}[F] - \Pr[F]| \leq k/2^{\ell}$. This suggests the following approximation algorithm for $\Pr[F]$. Given a proper coloring of F, for all $i \in \{1,\ldots,k\}$, explicitly construct sample space S_i for distribution $\mathcal{D}_i$ and let the sample space for $\mathcal{D}$ be $S = \times_{i\in\{1,\ldots,k\}}S_i$. By exhaustive enumeration of S, compute $\Pr_{\mathcal{D}}[F]$ exactly. This yields an algorithm with running time dominated by $\times_{i\in\{1,\ldots,k\}}|S_i|$ for approximating $\Pr[F]$. The running time of the algorithm depends in an exponential way on the value of k. In general it may not be possible to find a proper k-coloring for F where k is small. Instead, the idea is to find a proper coloring for a subformula G of F such that $\Pr[G]$ is close to $\Pr[F]$. This general approach is used to design deterministic algorithm that on input F and ϵ produces an ϵ-relative approximation of $\Pr[F]$ in polynomial time for fixed ϵ and $t = \log^{1-o(1)}(nm)$. For fixed ϵ and unrestricted clause length the running time the algorithm produces an ϵ-approximation in time polynomial in nm and $2^{\log^{1+o(1)}(nm)}$. For a fixed ϵ this algorithm is faster than the algorithm in [36, Nisan and Wigderson], but, unlike [36, Nisan and Wigderson], the running time grows exponentially with $1/\epsilon$.

5 Derandomizing Constant Depth Circuits

In this section polynomial size boolean circuits of constant depth with unbounded fan-in are considered. This class of circuits is commonly referred to as $\mathbf{AC}^0$. This section describes a pseudorandom generator for this class. The pseudorandom generator runs in time exponential in $\log^c(n)$ time, where the constant c depends on the depth bound on the circuit family. Note that this is much better than a straightforward exhaustive enumeration procedure, which would take time exponential in n. The pseudorandom generator is based on the difficulty of computing parity with a constant depth boolean circuit.

Definition (predicting the parity of its inputs): For $x \in \{0,1\}^n$, $x \odot x = \oplus_{i\in\{1,\ldots,n\}}x_i$ is the parity of the number of ones in x. For any circuit $C \in \mathcal{C}_n^d$, let p_C be the prediction

probability of C for the parity of its input, i.e.,

$$p_C = |\Pr_Z[C(Z) = Z \odot Z] - 1/2|,$$

where $Z \in_R \{0,1\}^n$. Let T_C be the total number of gates in C. The time-success ratio of C for predicting the parity of its inputs is $S_C = T_C/p_C$. ♣

The following beautiful lower bound theorem is a culmination of a number of papers, i.e., [15, Furst, Saxe and Sipser], [2, Ajtai], [42, Yao], [12, Cai], [20, Håstad].

Parity Theorem : There is a constant $\kappa > 0$ such that for any $C \in \mathcal{C}_n^{d,n}$, the time-success ratio S_C of C satisfies $S_C \geq 2^{n^{\kappa/d}}$.

Based on this lower bound, [35, Nisan], [36, Nisan and Wigderson], describe the following simple but elegant generator g. Set

$$r = \log(n)^{c(d+1)},$$

where $c > 0$ is a fixed constant. Let $t_1, \ldots, t_n \subset \{1, \ldots, r\}$ be sets that satisfy the following two properties:

(1) For all $i \in \{1, \ldots, n\}$, $|t_i| = \sqrt{r}$.

(2) For all $i, j \in \{1, \ldots, n\}$, $i \neq j$, $|t_i \cap t_j| \leq \log(n)$.

It is an exercise to find an efficient construction of the sets $t_1, \ldots, t_n$ with these two properties. For all $s \in \{0,1\}^r$, for each $i \in \{1, \ldots, n\}$, define function $b_i(s) = \oplus_{j \in t_i} s_j$, i.e., $b_i(s)$ is the parity of the number of ones in the bits of s indexed by t_i. Finally, let

$$g(s) = \langle b_1(s), \ldots, b_n(s) \rangle.$$

The following theorem is due to [35, Nisan], [36, Nisan and Wigderson].

Theorem : Let $q(n) = 2^{\log(n)^{c\kappa/2}}/n^3$. For all $C \in \mathcal{C}_n^{d,n}$, the time-success ratio S_C of C for distinguishing g satisfies $S_C \geq q(n)$.

This theorem shows how to derandomize any constant depth boolean circuit. Consider an alternate derandomization question where the circuit can be over any type of gates (i.e., not restricted to "and" and "or" gates). The techniques of [35, Nisan], [36, Nisan and Wigderson] do not apply to this problem even in the case when the depth is restricted to two, as one of the gates can be a "parity" gate, and the entire construction is based on lower bounds of boolean circuits for computing parity.

[30, Luby, Velickovic and Wigderson] provide the first subexponential simulation for circuits with parity gates. This paper shows how to ϵ-approximate the accepting probability of any *GF[2]* polynomial of size n in deterministic time double exponential in $\sqrt{\log(n/\epsilon)}$. The technique is strongly based on the pseudorandom generator of [35, Nisan] used for $\mathbf{AC}^0$ circuits described previously. [30, Luby, Velickovic and Wigderson] are able to apply this idea, and use the lower bounds on multiparty communication complexity of [7, Babai, Nisan and Szegedy].

6 Derandomizing General Circuits

In this section pseudorandom generators for polynomial size boolean circuits of unrestricted depth are considered. If, for each fixed polynomial n^c, there is a ϵ-pseudorandom generator $g : \{0,1\}^{r(n)} \to \{0,1\}^{\ell(n)}$ for $\mathcal{C}_n^{n^c,n^c}$, then for any $C \in \mathcal{C}_n^{n^c,n^c}$ one can ϵ-approximate the probability that C outputs 1 on a random input in deterministic time polynomial in n times $2^{r(n)}$. Thus, if $r(n) = \mathcal{O}(\log(n))$ then this is polynomial time overall. There is no proof in sight of this result at this point in time, and in fact it is easy to see that such a result is impossible if $\mathbf{NP} = \mathbf{P}$. On the other hand, as described below, it turns out that there is pseudorandom generator for this class if there are strong enough one-way functions.

Intuitively, a one-way function is easy to compute but hard to invert on average.

Definition (one-way function): Let $f : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ be a **P**-time function ensemble and let $\mathcal{C}$ be a circuit family. The success probability (inverting probability) of $C \in \mathcal{C}$ (with $\ell(n)$ input bits and n output bits) for f is

$$\delta = \Pr_X[f(C(f(X))) = f(X)],$$

where $X \in_R \{0,1\}^n$. Then, f is a $\mathbf{S}(n)$-secure one-way function for $\mathcal{C}$ if the time-success ratio of $\mathcal{C}$ is at least $\mathbf{S}(n)$. ♣

The definition of a one-way permutation is exactly the same as the definition of a one-way function, except that $f : \{0,1\}^n \to \{0,1\}^n$ as a function of $x \in \{0,1\}^n$ is a permutation.

[9, Blum and Micali] introduce the concept of a pseudorandom generator that is useful for cryptographic (and other) applications, and gave it the significance it has today by providing the first provable construction of a pseudorandom generator based on the conjectured difficulty of a well-known and well-studied computational problem, the discrete log problem.

Discrete log problem : Let $p \in \{0,1\}^n$ be a prime number and let g be a generator of $\mathcal{Z}_p^*$, i.e., for all $y \in \mathcal{Z}_p^*$, there is a unique $x \in \mathcal{Z}_{p-1}$ such that $g^x = y \bmod p$. Given p, g and $x \in \mathcal{Z}_{p-1}$, define $f(p,g,x) = \langle p, g, g^x \bmod p\rangle$. It is possible to compute $g^x \bmod p$ given p, g and x in $n^{\mathcal{O}(1)}$ time. The discrete log function is a permutation as a function of x, i.e., the unique inverse of $f(p,g,x)$ is $\langle p, g, x\rangle$. The values of p and g are not necessarily chosen randomly. The prime p is selected to have special properties which seem in practice to make the discrete log function hard to invert. An example of such a property is that p is selected so that that $p-1$ has some fairly large prime divisors. For a large class of primes p and generators g there is no known **P**-time function ensemble that on input p, g and $g^x \bmod p$ can produce x on average for $x \in_R \mathcal{Z}_{p-1}$.

The original construction of a pseudorandom generator based on the discrete log problem is quite complicated. The following elegant theorem, due to [17, Goldreich and Levin], substantially simplifies all previously known constructions of pseudorandom generators based on one-way functions. The simplest known proof of this theorem is due to C. Rackoff, R. Venkatesan and L. Levin, inspired by [4, Alexi, Chor, Goldreich and Schnorr].

Definition (inner product bit is hidden): Let $f : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ be a **P**-time

function ensemble. Let $x \in \{0,1\}^n$ and $z \in \{0,1\}^n$. Then, the *inner product bit* of $f(x)$ is $x \odot z$. The success probability (prediction probability) of circuit $C \in \mathcal{C}$ (with $\ell(n)+n$ input bits and one output bit) for the inner product bit of f is

$$\delta = \Pr_{X,Z}[C(f(X),Z) = X \odot Z] - \Pr_{X,Z}[C(f(X),Z) \neq X \odot Z].$$

where $X, Z \in_R \{0,1\}^n$. Then, the inner product bit of f is a $\mathbf{S}(n)$-secure for $\mathcal{C}$ if the time-success ratio is at least $\mathbf{S}(n)$. ♣

Hidden Bit Theorem : If f is a one-way function then the inner product bit of f is hidden. In particular, if there is a circuit family $\mathcal{C}$ with time-success ratio $\mathbf{S}(n)$ for predicting the inner product bit then there is a circuit family $\mathcal{C}'$ with time-success ratio $\mathbf{S}(n)^c$ for inverting f for some constant $c > 0$.

¿From the Hidden Bit Theorem, it is not hard to show that if $f(x)$ is a one-way permutation for $\mathcal{C}$ then $g(x,z) = \langle f(x), z, x \odot z \rangle$ is a pseudorandom generator for $\mathcal{C}$ that stretches by 1 bit. This simple construction of a pseudorandom generator was one of the motivating forces behind the work of [17, Goldreich and Levin].

One can construct a pseudorandom generator that stretches by an arbitrary polynomial amount based on any one-way permutation.

Theorem : Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way permutation. Define $\mathbf{P}$-time function ensemble $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{\ell(n)}$, as

$$g(x,z) = \langle z, x \odot z, f(x) \odot z, f^{(2)}(x) \odot z, \ldots, f^{(\ell(n)-n-1)}(x) \odot z \rangle,$$

where $f^{(i)}$ is the function f composed with itself i times. Then g is a pseudorandom generator.

Note that since the discrete log is a permutation as a function of x for a fixed value of p and g, the construction described in this theorem can be immediately applied to yield an extremely simple pseudorandom generator based on the discrete log. This theorem is a combination of a theorem due to [16, Goldreich, Goldwasser and Micali] and the Hidden Bit Theorem of [17, Goldreich,Levin].

There is a generalization of this work that shows a more complicated reduction from an arbitrary one-way function (e.g., a function that is far from being a permutation) to a pseudorandom generator, due to [19, Håstad, Impagliazzo, Levin and Luby]. For a more complete history and description, see [26, Luby].

References

[1] L. Adleman, "Two Theorems on Random Polynomial Time", **FOCS**, 1978, pp. 75–83.

[2] M. Ajtai, "$\sum_1^1$-Formulae on Finite Structures", *Annals of Pure and Applied Logic*, Vol. 24, 1983, pp. 1–48.

[3] M. Ajtai and A. Wigderson, "Deterministic Simulation of Probabilistic constant depth circuits", **FOCS**, 1985, pp. 11-19.

[4] W. Alexi, B. Chor, O. Goldreich, C. Schnorr, "RSA/Rabin Functions: Certain Parts are as Hard as the Whole", *SIAM J. on Computing*, Vol. 17, No. 2, April 1988, pp. 194–209.

[5] N. Alon, L. Babai, A. Itai, "A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem", *Journal of Algorithms*, Vol. 7, 1986, pp. 567–583.

[6] N. Alon, O. Goldreich, J. Håstad, R. Peralta, "Simple constructions of almost k-wise independent random variables", *Random Structures and Algorithms*, 1992, Vol. 3, No. 3, pp. 289–304. (see also addendum in *Random Structures and Algorithms*, 1993, Vol. 4, No. 1, pp. 119–120.)

[7] L. Babai, N. Nisan, and M. Szegedy, "Multiparty protocols and logspace-hard pseudo-random sequences", preliminary version in **STOC**, 1989, pp. 1–11, journal version in *JCSS*, Vol. 45, No. 2, October 1992, pp. 204–232.

[8] B. Berger, J. Rompel, "Simulating $(\log^c n)$-wise Independence in *NC*", Proceedings of **FOCS**, 1989, pp. 1–7.

[9] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. on Computing*, Vol. 13, 1984, pp. 850–864. A preliminary version appears in **FOCS**, 1982, pp. 112–117.

[10] R. Boppana and R. Hirschfeld, "Pseudo-random generators and complexity classes", **Advances in Computer Research**, Vol. 5, 1989, editor S. Micali, JAI Press, pp. 1–26.,

[11] J. Boyar, "Inferring Sequences Produced by Pseudo-Random Number Generators", *J. of the ACM*, Vol. 36, No. 1, 1989, pp.129–141.

[12] J. Cai, "With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy", *J. of Computer and System Sci.*, Vol. 38, 1989, pp. 68–85. A preliminary version appears in **STOC**, 1986, pp. 21–29.

[13] B. Chor, O. Goldreich. "On the power of two-point sampling," *Journal of Complexity*, Vol. 5, 1989, pp. 96–106.

[14] P. Erdös and R. Rado, "Intersection theorems for systems of sets", Journal of the London Math. Society, Vol. 35, pp. 85–90.

[15] M. Furst, J. Saxe, M. Sipser, "Parity, Circuits and the Polynomial Time Hierarchy", **FOCS**, 1981, pp. 260–270.

[16] O. Goldreich, S. Goldwasser, S. Micali, "How to Construct Random Functions", *J. of ACM*, Vol. 33, No. 4, 1986, pp. 792–807. A preliminary version appears in **FOCS**, 1984.

[17] O. Goldreich, L. Levin, "A Hard-Core Predicate for any One-way Function", **STOC**, 1989, pp. 25–32.

[18] S. Goldwasser and S. Micali, "Probabilistic Encryption", *J. of Computer and System Sci.*, Vol. 28, 1984, pp. 270–299. A preliminary version appears in **STOC**, 1982, pp. 365–377.

[19] R. Impagliazzo, L. Levin, J. Håstad, M. Luby, "A Pseudorandom generator from any one-way function," 20^{th} **STOC**, 1989, and submitted to *SIAM J. on Computing.*

[20] J. Håstad, "Computational limitations for small depth circuits", Ph.D. thesis, MIT press, 1986.

[21] D. Karger and R. Motwani, "Derandomization through Approximation: An *NC* Algorithm for Minimum Cuts", **STOC**, 1994, pp. 497–506.

[22] H. Karloff, P. Raghavan, "Randomized Algorithms and Pseudorandom Numbers", *Journal of the ACM*, 1993, Vol. 40, pp. 421–453.

[23] D. Knuth, **Semi-Numerical Algorithm, The Art of Computer Programming**, Addison-Wesley, Second Edition, Vol. 2, 1981.

[24] A. N. Kolmogorov, "Three Approaches to the Concept of the Amount of Information", *Probl. Inf. Transm.*, Vol. 1, No. 1, 1965.

[25] H. Krawczyk, "How to Predict Congruential Generators", *J. of Algorithms*, Vol. 13, 1992. pp. 527–545.

[26] M. Luby, **Pseudorandomness and Cryptographic Applications**, *Princeton Computer Science Notes*, David R. Hanson and Robert E. Tarjan, Editors, Princeton University Press, January 1996.

[27] M. Luby, "A Simple Parallel Algorithm for the Maximal Independent Set Problem," *SIAM J. on Computing*, Vol. 15, No. 4, November 1986, pp. 1036–1053.

[28] M. Luby, "Removing Randomness in Parallel Computation Without a Processor Penalty," *J. of Computer and System Sciences*, Vol. 47, No. 2, 1993, pp. 250-286.

[29] M. Luby, B. Veličković, "On Deterministic Approximation of DNF", to appear in a special issue of *Algorithmica* devoted to randomized algorithms.

[30] M. Luby, B. Veličković, and A. Wigderson, "Deterministic Approximate Counting of Depth-2 Circuits", appears in the *Proceedings of the Second Israeli Symposium on Theory of Computing and Systems*, 1993.

[31] M. Luby and A. Wigderson, "Pairwise Independence and Derandomization", *ICSI Tech Report No. TR-95-035*, July, 1995.

[32] R. Motwani, J. Naor, M. Naor, "The Probabilistic Method Yields Parallel Deterministic Algorithms", *J. of Computer and System Sciences*, 1994, Vol. 49, pp. 478–516.

[33] K. Mulmuley, "Pseudo-random Generators in Geometric Algorithms", to appear in a special issue of *Algorithmica* devoted to randomized algorithms.

[34] M. Naor and S. Naor, "Small Bias Probability Spaces: Efficient Constructions and Applications", **STOC**, 1990, pp. 213-223.

[35] N. Nisan, "Pseudorandom bits for constant depth circuits", *Combinatorica*, Vol. 1, 1991, pp. 63–70.

[36] N. Nisan and A. Wigderson. "Hardness vs. Randomness", *JCSS*, Vol. 49, No. 2, 1994, pp. 149–167.

[37] A. Razborov, "Lower bounds on the monotone complexity of some Boolean functions," *Doklady Akademii Nauk SSSR* 281:4, 1985, pp. 798–801, (in Russian). English translation in *Soviet Mathematics Doklady* 31, 354–357.

[38] A. Shamir, "On the generation of cryptographically strong pseudo-random sequences", *ACM Transactions on Computer Systems*, Vol. 1, No. 1, 1983, pp. 38–44. A preliminary version appears in the 8^{th} *ICALP* and appears in **Lecture Notes on Computer Science**, 1981, Springer Verlag, pp. 544–550.

[39] L. Valiant, "The complexity of computing the permanent", *Theoretical Computer Science*, 1979, No. 8, pp 189-201.

[40] M. Wegman, J. Carter. "New hash functions and their use in authentication and set equality," *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265–279.

[41] A. Yao, "Theory and Applications of Trapdoor Functions", **FOCS**, 1982, pp. 80–91.

[42] A. Yao, "Separating the Polynomial-Time Hierarchy by Oracles", **FOCS**, 1985, pp. 1–10.