Some Results on Derandomization

Harry Buhrman*
CWI and University of Amsterdam

Lance Fortnow†
University of Chicago

A. Pavan‡
Iowa State University

Abstract

We show several results about derandomization including

1. If NP is easy on average then efficient pseudorandom generators exist and P = BPP.
2. If NP is easy on average then given an NP machine M we can easily on average find accepting computations of $M(x)$ when it accepts.
3. For any A in EXP, if NEXP^A is in P^A/poly then $\mathrm{NEXP}^A = \mathrm{EXP}^A$.
4. If A is Σ_k^{p}-complete and NEXP^A is in P^A/poly then $\mathrm{NEXP}^A = \mathrm{EXP} = \mathrm{MA}^A$.

1 Introduction

The past several years have seen several exciting results in the area of derandomization (e.g. [IW97, IW01, KvM02, MV99, IKW02]) giving strong evidence that we can often eliminate randomness from computation. These papers use hardness results to get derandomization and exhibit many exciting applications of these methods.

Can the collapse of complexity classes cause derandomization to occur? If P = NP then one can easily show that efficient pseudorandom generators exist[1]. We weaken this assumption to show that even if NP is just easy on average then pseudorandom number generators exist and P = BPP.

We use this result to study relations between distributional search problems and decision problems. These relations are well understood in the context of worst-case complexity. For example, every NP search problem is reducible to a decision problem in NP, this implies that if P = NP, then accepting computations of NP machines can be computed in polynomial time. We also know that if P = NP, then all optimization problems are solvable in polynomial time, and indeed the entire polynomial-time hierarchy is in P. We do not have analogous results in average-case complexity. Say a class $\mathcal{C}$ is easy on average if for every language L in $\mathcal{C}$ and every polynomial-time computable distribution μ, L can be decided in average-polynomial time with respect to μ. Ideally, one would like to show if NP is easy on average, then the entire polynomial-time hierarchy is easy on average. However, this question is open. The conventional methods used in worst-case complexity that

*Address: CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: buhrman@cwi.nl. Work done while the author visited the NEC Research Institute.

†Address: Department of Computer Science, University of Chicago, Chicago, IL 60637. Email: fortnow@cs.uchicago.edu. http://people.cs.uchicago.edu/~fortnow. Work done while the author was at the NEC Research Institute.

‡Address: Department of Computer Science, 226 Atanasoff Hall, Iowa State University, Ames, IA 50011. Email: pavan@cs.iastate.edu. Work done while the author was at the NEC Research Institute.

[1]Note that efficient pseudorandom generators are different from cryptographic pseudorandom generators. It is known that if P = NP, then cryptographic pseudorandom generators do not exist

exploit the self reducibility of SAT do not seem to carry over to the average-case world. There are some partial results in this direction. Ben-David et. al. showed that every distributional NP search problem is random truth-table reducible to a distributional decision problem [BCGL92]. Schuler and Watanabe [SW95] showed that P^{NP} is easy on average, if every language in NP is solvable in average-polynomial time with respect to every P^{NP}-sampleable distribution. Note that the hypothesis here is much stronger than what we would like, i.e., NP is easy on average.

Here we show that if NP is easy on average, then all search problems in NP are also easy on average. The proof uses the earlier mentioned derandomization result. Thus we obtain a result that is analogous to the result in worst-case complexity. Moreover we extend a result of Ben-David et. al. [BCGL92] and show that if NP is easy on average then one can find in exponential time a witness to any NE predicate.

Impagliazzo, Kabanets and Wigderson [IKW02] use derandomization techniques to show that if all languages in nondeterministic exponential time (NEXP) have polynomial-size circuits then NEXP = MA where MA is the class of languages with Merlin-Arthur protocols. Their result gives the first proof that if NEXP is in P/poly then NEXP = EXP.

Can one extend their result to questions like what happens if $\mathrm{NEXP}^{\mathrm{SAT}}$ is in $\mathrm{P}^{\mathrm{SAT}}$/poly? One cannot apply the Impagliazzo-Kabanets-Wigderson result directly as their proof does not relativize.

For any A in EXP, we get NEXP^A is in P^A/poly implies $\mathrm{NEXP}^A = \mathrm{EXP}^A$.

If A is Σ_k^{p}-complete for some k (for example $A = \mathrm{SAT}$) we get that if NEXP^A is in P^A/poly then $\mathrm{NEXP}^A = \mathrm{MA}^A = \mathrm{EXP}$ where the final EXP is unrelativized. As a corollary we get that $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}}$ is in $\mathrm{P}^{\Sigma_k^{\mathrm{p}}}$/poly for at most one k.

We combine techniques from derandomization, average-case complexity, interactive proof systems, diagonalization and the structure of the polynomial-time and exponential-time hierarchies to prove our results.

2 Preliminaries

We assume a standard background in basic computational complexity.

We let $\Sigma = \{0,1\}$ and A^n represent $A \cap \Sigma^n$. For strings x and y of the same length n we let $x \cdot y$ be the dot product of x and y, viewing x and y as n-dimensional vectors over GF[2].

A tally set is a subset of 0^*.

We define the classes $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$, $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$ and $\mathrm{EE} = \mathrm{DTIME}(2^{2^{O(n)}})$. The classes NE, NEXP and NEE are the nondeterministic versions of E, EXP and EE respectively.

A language A is in $\mathrm{SIZE}(s(n))$ if there is a constant c such that for every n there is a circuit of size $c\,s(n)$ that computes A^n.

A language A is in io$-[\mathcal{C}]$ for some class $\mathcal{C}$ if there is a language B in $\mathcal{C}$ such that for an infinite number of n, $A^n = B^n$.

The class Σ_k^{p} represents the kth-level of the polynomial-time hierarchy with $\Sigma_1^{\mathrm{p}} = \mathrm{NP}$. The classes Σ_k^{EXP} represent the kth-level of the exponential-time hierarchy. For $k \geq 1$, $\Sigma_k^{\mathrm{EXP}} = \mathrm{NEXP}^{\Sigma_{k-1}^{\mathrm{p}}}$.

A theorem *relativizes* if for all A, the theorem holds if *all* machines involved have access to A. Most theorems in computational complexity relativize.

Let $\mathcal{C}$ be any complexity class. A language L belongs to $\mathcal{C}$*/poly* if there exists a polynomial-bounded function $f : \mathbb{N} \to \Sigma^*$ and a language L' in $\mathcal{C}$ such that $x \in L \Leftrightarrow \langle x, f(|x|)\rangle \in L'$.

A function $G = \{G_n\}_n$, $G_n : \Sigma^l \to \Sigma^n$ is an *efficient pseudorandom generator* if $l = O(\log n)$, G

is computable in $O(n)$ time, and for any circuit circuit C of size n

$$|\Pr_{x\in\Sigma^n}[C(x)=1]-\Pr_{y\in\Sigma^l}[C(G_n(y))=1]|\leq\frac{1}{n}.$$

2.1 Average-Case Complexity

In this section we give some necessary definitions from average-case complexity. We recommend the survey of Wang [Wan97] for a full background on average-case complexity.

Even if P $\neq$ NP, we still could have that NP is easy for all practical purposes, i.e. given any *reasonable* distribution, NP is easy on average on that distribution.

To formalize this problem, Levin [Lev86] developed a notion of average-polynomial time. A distributional problem is a pair (L,μ) where L is a language and μ is a distribution over Σ^*. A distributional problem (L,μ) is in Average-P, or L is in average polynomial-time with respect to μ, if there exists a Turing machine M computing L and an $\epsilon>0$ such that

$$\sum_{x\in\Sigma^*}\frac{(T_M(x))^\epsilon}{|x|}\,\mu'(x)<\infty$$

Here $\mu(x)$ represents the probability that a chosen string is lexicographically at most x, $\mu'(x)=\mu(x)-\mu(x-1)$ is the probability that the chosen string is x and $T_M(x)$ is the running time of machine M on input x. We denote the set of such (L,μ) by Average-P.

Blass and Gurevich [BG93] extended Levin's definition to randomized average-polynomial time. We use a slightly different and simpler definition for our purposes. We say L is in randomized average-polynomial time with respect to μ, or (L,μ) is in random average-polynomial time if here exists a probabilistic Turing machine M that decides L and an $\epsilon>0$ such that

$$\sum_x\sum_{R_x}\frac{T_M^\epsilon(x)}{|x|}\mu'(x)2^{-|R|}<\infty,$$

where R_x is the set of random strings that M uses on x.

Levin defines the class DistNP which contains pairs of (L,μ) where L is in NP and μ is polynomial-time computable. We say NP is easy on average if DistNP is in Average-P. It remains open, even in relativized worlds, whether this conjecture is truly different than P = NP or if this conjecture implies the collapse of the polynomial-time hierarchy.

2.2 Arthur-Merlin Games

Babai [Bab85, BM88] developed Arthur-Merlin games. In this model an arbitrarily powerful Merlin is trying to convince an untrusting probabilistic polynomial-time Arthur that a string x is in some language L. For x in L Merlin should succeed in causing Arthur to accept with high probability. For x not in L, no variation of Merlin should be able to convince Arthur to accept with more than some small probability.

The class MA consists of languages with protocols where Merlin sends a message followed by Arthur's probabilistic computation. The class AM consists of protocols where Arthur first sends random coins to Merlin who responds with a message that Arthur now deterministically decides whether to accept. Babai shows that MA is contained in AM and this result relativizes.

The classes MAEXP and AMEXP are the same as MA and AM except we allow Arthur to run in time $2^{n^{O(1)}}$.

2.3 If EXP has Small Circuits

Informally a *probabilistically checkable proof system* is a proof for an accepting nondeterministic computation that can be verified with high confidence using only a few probabilistically-chosen queries to the proof. Probabilistically checkable proof systems were developed by Fortnow, Rompel and Sipser [FRS94] and given their name by Arora and Safra [AS98].

Babai, Fortnow and Lund [BFL91] show the following result for EXP.

Theorem 2.1 (BFL) *Every language in* EXP *has an (exponentially-large) probabilistically checkable proof system whose answers are computable in* EXP*. The probabilistic checking algorithm runs in polynomial time with random access to the proof.*

Nisan observed that one can use this formulation to get a consequence of EXP having small circuits.

Theorem 2.2 (BFL-Nisan) *If* EXP *is in* P/poly *then* EXP = MA.

Theorems 2.1 and 2.2 do not relativize though we can get a limited relativization for Theorem 2.2.

Theorem 2.3 *For any A, if* EXP *is in* P^A/poly *then* EXP $\subseteq \mathrm{MA}^A$.

Proof: Let L be in EXP and x be in L. Merlin gives Arthur the circuit C such that C^A computes the probabilistic checkable proof for x in L. Arthur then uses C making queries to A as necessary to verify that x is in L. If $x \notin L$ then every proof will make Arthur reject with high probability and hence also the proof described by the C^A circuit. □

3 Average-Case Complexity and Derandomization

In this section we will show that the assumption that NP is easy on average gives us derandomization. It is known that if NP is easy on average, then BPP = ZPP [Imp95]. Our theorem is an improvement of this result.

Theorem 3.1 *If* NP *is easy on average then pseudorandom generators exist and* P = BPP.

We need several results from the areas of derandomization and average-case complexity. First Impagliazzo and Wigderson [IW97] show that for full derandomization it suffices if E requires large circuits.

Theorem 3.2 (Impagliazzo-Wigderson) *Suppose there exists a language L in* E *such that for some $\epsilon > 0$ and all but a finite number of n, computing L on strings of length n cannot be done by circuits of size $2^{\epsilon n}$. We then have that efficient pseudorandom generators exist and* P = BPP.

Suppose the assumption needed for Theorem 3.2 fails, i.e. for every $\epsilon > 0$, E has circuits of size $2^{\epsilon n}$. We would like some weak version of Theorem 2.2 but Theorem 2.1 is too weak to apply to this case. We instead use the following result due to Babai, Fortnow, Levin and Szegedy [BFLS91] and Polishchuk and Spielman [PS94].

Theorem 3.3 (BFLS-PS) *For any ϵ, $0 < \epsilon \leq 1$, and given the computational path of a nondeterministic machine using time $t(n)$, there exists a probabilistically checkable proof of this computation computable in time $t^{1+\epsilon}(n)$ and probabilistically verifiable in time* $\log^{O(1/\epsilon)} t(n)$.

From Theorem 3.3 we can get the following lemma.

Lemma 3.4 *Suppose that for all ϵ and A in* E *and for infinitely many lengths n, A^n has circuits of size $2^{\epsilon n}$. For every $\epsilon > 0$ and A in* E*, there exists a Merlin-Arthur protocol where Arthur uses time $2^{\epsilon n}$ and for infinitely many n, if $|x| = n$ then*

- *If x is in A then Arthur will accept with probability at least $2/3$.*
- *If x is not in A, no matter what is Merlin's message, A will accept with probability at most $1/3$.*

Proof: Fix an input x. If x is in A then there is a computation path for this of length $2^{O(n)}$. By Theorem 3.3 there is a probabilistically checkable proof of this computation computable in time $2^{O(n)}$. By assumption, for $\epsilon > 0$ and infinitely many n, this proof can be described by a circuit of size $2^{\epsilon n}$. Merlin just sends over this circuit and Arthur probabilistically verifies this proof in time polynomial in n with random access to the proof which he can simulate by evaluating the circuit in time $2^{\epsilon n}$. □

We also need some results on average-case complexity due to Ben-David, Chor, Goldreich and Luby [BCGL92] and Köbler and Schuler [KS98].

Theorem 3.5 (BCGL) *If* NP *is easy on average then* E = NE

The idea of the proof uses a μ' that puts enough measure on the tally strings such that any algorithm that is polynomial on μ average should take polynomial-time on all the tally strings. Hence the tally sets in NP are in P which implies that E = NE.

Theorem 3.6 (Köbler-Schuler) *If* NP *is easy on average then* MA = NP.

Combining Lemma 3.4 and the techniques of Theorem 3.6 we can show the following.

Lemma 3.7 *Assume* NP *is easy on average. If for all $\epsilon > 0$,* E *infinitely often has circuits of size $2^{\epsilon n}$ then for all A in* E *and $\epsilon > 0$ there is a language L in* NTIME($2^{\epsilon n}$) *such that for infinitely many n, $A^n = L^n$.*

Proof: Consider the following set in CoNP. Let ϵ be a fixed constant between 0 and 1.

$$A = \{x \mid x \text{ is the truth table of a boolean } f : \{0,1\}^{\log n} \to \{0,1\} \text{ of } > \epsilon n \text{ circuit complexity}\}.$$

Most functions f have circuit complexity at least ϵn. Hence, there is a constant c such that at least $1/c$ fraction of strings of length n are in A. Now consider the uniform distribution that is clearly polynomial-time computable, i.e, $\mu'(x) = \frac{1}{n^2}\frac{1}{2^n}$, $|x| = n$. Since NP is easy on average, there exists a machine M that decides A and the running time of M is polynomial on μ-average. From this it follows that there exists a constant k such that for every n, M runs in less than n^k time on at least $(1 - 1/n^2)$ fraction of strings of length n. Thus for every n, there exists at least one string x of length n such that x belongs to A and M runs in n^k time on x.

Let A be any language in E. Since E infinitely often has circuits of size $2^{\epsilon n}$, by Lemma 3.4, there exists a MA protocol $\mathcal{P}$ where Arthur uses $2^{\epsilon n}$ time and the language accepted by the protocol coincides with A for infinitely many lengths n. We show that we can replace this MA protocol with a nondeterministic machine that runs in time $2^{\epsilon n}$. Consider a nondeterministic machine N that on an input x of length n behaves as follows: N guesses a string f of length n and runs M on f for n^k steps. If either M does not halt within n^k steps or M rejects f, then this path terminates without accepting. By the previous arguments there exists at least one guess f such that M accepts f

within n^k steps. Consider this path, f is a boolean function with circuit complexity at least ϵn. By Theorem 3.2, the truth table of a function with circuit complexity ϵn can be used construct an efficient pseudorandom generator that stretches $\log n$ bits to n bits. Given such pseudorandom generator, computation of a MA protocol where Arthur runs in time $t(n)$ can be derandomized to obtain a nondeterministic machine that runs in time $t(n)$. Now N, using f, converts the protocol $\mathcal{P}$ to a nondeterministic computation. Recall that for infinitely many n, the language of the protocol $\mathcal{P}$ coincides with A. Thus for infinitely many n, the language L accepted by N is same as A. □

We are now ready to prove the main theorem of this section.

Proof of Theorem 3.1: Suppose that there exists a language A in E and an ϵ such that for almost every n, A^n has circuit complexity at least $2^{\epsilon n}$, then by Theorem 3.2 we have that pseudorandom generators exist and we are done. We will show that indeed there exists such a hard language in E. By contradiction assume that for every $\epsilon > 0$ and set $A \in$ E there exist infinitely many n such that A^n has circuits of size $2^{\epsilon n}$. For some d to be chosen later, let A be a set in DTIME(2^{dn}) such that for every Turing machine M using time bounded by $2^{(d-1)n}$, and for all but a finite number of n, M fails to compute A correctly on some input of length n. One can construct such an A by simple diagonalization.

By Lemma 3.7, we have that there is a language B in NTIME(2^n) such that $B^n = A^n$ for infinitely many n. Since NP is easy on average, by Theorem 3.5, E = NE. Impagliazzo, Kabanets, and Wigderson [IKW02] showed that if E = NE, then there exists a fixed constant e such that NTIME(2^n) $\subseteq$ DTIME(2^{en}). Thus B is in DTIME(2^{en}). Choosing $d = e + 1$ gives a contradiction with the construction of A. □

Now we show some interesting applications of the previous theorem.

Theorem 3.8 *Let T be a tally set accepted by an* NP *machine M. If* NP *is easy on average then there exists a polynomial time algorithm such that for every $0^n \in T$ outputs an accepting computation of $M(0^n)$.*

Proof: The idea is to first use the assumption that NP is easy on average to show a probabilistic algorithm that with high probability outputs an accepting computation of $M(0^n)$ for 0^n in T. Theorem 3.1 shows that we can derandomize that algorithm.

Suppose M has computation paths encoded as strings of length n^k. Let R_n be a n^{2k} bit string. We view R_n as concatenation of n^k strings each of length n^k, i.e., $R_n = r_1 r_2 \cdots r_{n^k}$ Consider the NP language L consisting of tuples $\langle 0^n, l, i, R_n \rangle$ such that there exists a computation path p of M such that

1. p is an accepting computation path of $M(0^n)$,
2. $p \cdot r_j = 1$ for all j, $1 \leq j \leq l$ and
3. The ith bit of p is one.

Consider the polynomial-time computable distribution μ with

$$\mu'(\langle 0^n, l, i, R_n \rangle) = \frac{1}{n^{2+2k} 2^{n^{2k}}}$$

for $1 \leq i \leq n^k$ and $0 \leq l \leq n^k$.

Since NP is easy on average, L is easy with distribution μ. Thus there exists a machine M that decides L and a constant $r > 1$ such that

$$\sum_x \frac{T_M(x)^{1/r}}{|x|} \mu'(x) < \infty, \tag{1}$$

where x is of the form $\langle 0^n, l, i, R_n \rangle$.

Claim 3.9 *For all but finitely many n, for every $l \leq n^k$, and for every $i \leq n^k$ there exist at most $2^{n^2 k}/n^{2k+3}$ R_n's such that $T_M(\langle 0^n, l, i, R_n \rangle) > n^{(6k+7)r}$.*

Proof: Suppose not. Then for infinitely many n there exist l and i such that $T_M(\langle 0^n, l, R_n, i \rangle) > n^{(6k+7)r}$ for more than $\frac{2^{n^2}}{n^{2k+3}}$ R_n's. Observe that $\langle 0^n, l, R_n, i \rangle$ can be encoded as a string of length n^{2k+1}.

$$\begin{aligned}
\sum_x \frac{T_M^{1/r}(x)}{|x|}\mu'(x) &= \sum_{n=1}^{\infty} \sum_{|x|=n^{2k+1}} \frac{T_M^{1/r}(x)}{|x|}\mu'(x) \\
&> \sum_{n=1}^{\infty} \frac{n^{6k+7}}{n^{2k+1}} \frac{1}{n^{2k+2}} \frac{1}{2^{n^{2k}}} \frac{2^{n^{2k}}}{n^{2k+3}} \\
&\geq \infty
\end{aligned}$$

This contradicts Equation 1. □

This implies that for all but finitely many n, there exists a set S, of R_n's, with cardinality at least $(1 - \frac{1}{n^3})2^{n^{2k}}$ such that for every $R_n \in S$, for every $i \leq n^k$, for every $l \leq n^k$,

$$T_M(\langle 0^n, l, i, R \rangle) \leq n^{(6k+7)r}.$$

Valiant and Vazirani [VV86] show that if $0^n \in T$, then for a random choice of l and R_n there exists unique accepting path p such that $r_1.p = \cdots = r_l.p = 1$ with probability at least $1/n^2$. Thus for a random choice of R_n and l, with probability at least $1/2n^2$, M runs in $n^{(6k+7)r}$ time on $\langle 0^n, l, i, R_n \rangle$ for every i. Moreover, there exists an unique accepting computation p such that $r_1.p = \cdots = r_l.p = 1$.

Thus the following probabilistic algorithm computes an accepting computation of M on 0^n: Randomly pick R_n and l and construct p by running M on $\langle 0^n, l, i, R_n \rangle$ for $n^{(6k+7)r}$ steps. If either the simulation is not over in the prescribed time or p is not a valid accepting path output $\perp$, else output p.

It is clear that if $0^n \in T$, then the above algorithm outputs an accepting path p with probability with probability at least $1/2n^2$, and if $0^n \notin T$, then it always outputs $\perp$. □

Theorem 3.8 now yields an extension to Theorem 3.5.

Corollary 3.10 *If* NP *is easy on average then for every* NE *predicate one can find a witness in time $2^{O(n)}$.*

This is indeed a strengthening of Theorem 3.5 since Impagliazzo and Tardos [IT89] constructed a relativized world where E = NE but there exists an NE predicate for which no witnesses can be found in time $2^{O(n)}$

We can generalize theorem 3.8 by allowing the inputs to also be chosen according to some distribution giving the following result.

Theorem 3.11 *If* NP *is easy on average then for any* NP *machine N and any polynomial-time computable distribution μ there is a function f computable in average polynomial-time according to μ such that $f(x)$ is an accepting path of N if $N(x)$ accepts.*

Proof: Let N be a NP machine and μ be any polynomial-time computable distribution. If N accepts x, then there is an accepting path p along which N accepts x. Let f be the following partial function.

$$f(x) = \begin{cases} p & \text{if } p \text{ is an accepting path of } N \text{ on } x \\ \perp & \text{else} \end{cases}$$

Note that f is actually a multi-valued function. Our goal is to show that for each x, we can compute *a value* of $f(x)$ in average polynomial time.

Without loss of generality, We assume that $|p| = |x|$. In the following assume $R = r_1 r_2 \cdots r_n$ where $|r_i| = n$. As before, consider the NP language L consisting of tuples $\langle x, l, R, i\rangle$ such that there exists an accepting path p of N on x, $p \cdot r_1 = p \cdot r_2 = \cdots p \cdot r_l = 1$, and the ith bit of p is one. Let ν be

$$\nu'(\langle x, l, R, i\rangle) = \mu'(x)\frac{1}{n^2}\frac{1}{2^{|R|}}$$

Again, by the result of Valiant and Vazirani, if N accepts x, then for a random choice of R and l there exists a unique accepting path p such that $r_1 \cdot v = \cdots r_l \cdot v = 1$ with probability at least $1/n^2$. Thus we can compute a value of $f(x)$ by randomly choosing a string R and deciding the membership of $\langle x, l, R, i\rangle$, $1 \le i \le n$, in L. This means computing f is random truth-table reducible to (L, ν). By our assumption (L, ν) is in Average-P. This implies f can be computed in random average-polynomial time [BCGL92]. Let M be the machine that computes f in random average polynomial-time. Now our goal is to show that the computation of M can be made deterministic.

We can easily modify M such that on input x, M either outputs $f(x)$ or outputs "?", i.e., M never errs. Thus there exists $c > 0$

$$\sum_{(x,R)} \frac{t^{1/c}(x, R)}{|x|}\mu'(x)2^{-|R|} < \infty,$$

where t is the running time of M. Note that M outputs a value of $f(x)$ with probability at least $1/n^2$.

Blass and Gurevich [BG93] showed how to amplify the success probability of random average-polynomial time machine that never errs. They showed such an algorithm can be converted to an algorithm whose running time is average on polynomial such that the new algorithm always outputs the correct answer, i.e., the success probability of the new machine is one. For our purposes we need the following amplification.

Lemma 3.12 *There exists a randomized machine M' whose running time is polynomial on average with respect to μ. Moreover the success probability of M' is one, the number of random bits used by M' is bounded by a polynomial, and M' always halts within 2^n steps.*

The proof is essentially same as the proof of Blass and Gurevich. For the sake of completeness we give a proof. However we first finish the proof of the theorem assuming above lemma. We defer the proof of Lemma 3.12 until later.

By the above lemma, there exists a constant k such that

$$\sum_{(x,R')} \frac{T^{1/k}_{M'}(x, R')}{|x|}\mu'(x)2^{-|R'|} < \infty.$$

We now show how to derandomize the computation of M'. From now, we denote the random strings of M' with R. Given x, let s_i be the set of R's such that $n^{2^{i-1}} \le T_{M'}(x, R) < n^{2^i}$, and let

$f_i = |s_i|/2^{|R|}$. Since the running time of M' is at most 2^n, this partitions the set of R's into at most $\log n$ sets. Thus there exists i such that $f_i \geq 1/\log n$. Denote such i with m_x. We will first show that the expectation of $\frac{(n^{2^{m_x}})^{1/2k}}{\log n}$ is finite.

$$\begin{aligned}
\infty &> \sum_{(x,R)} \frac{T^{1/k}_{M'}(x,R)}{|x|}\mu'(x)2^{-|R|} \\
&= \sum_x \frac{\mu'(x)}{|x|} \sum_R T^{1/k}_{M'}(x,R)2^{-|R|} \\
&\geq \sum_x \frac{\mu'(x)}{|x|} \sum_i (n^{2^i})^{1/2k} f_i \\
&\geq \sum_x \frac{\mu'(x)}{|x|} \frac{(n^{2^{m_x}})^{1/2k}}{\log n}
\end{aligned}$$

Since DistNP $\subseteq$ Average-P, by Theorem 3.1, there exists a family of PRGs $G_n : n \to 2^n$, and each G_n can be computed in 2^n time. Since G_n is pseudo-random generator, for every circuit C of size 2^n, the probability that C can distinguish the output of G_n from uniform distribution is less than $1/2^n$. Consider the following deterministic simulation of M'.

input x;
for $i = 1$ to $\log n$
    Simulate M' on x using the output of $G_{2^{i+1}\log n}$ as random bits for n^{2^i} steps.
    If any of the simulations halt and produce correct answer then STOP

We claim that for each x, the m_xth iteration of the for loop produces the correct answer. In the following we drop the subscript of m_x. Consider the behavior of M' on x. M' halts in n^{2^m} time on $1/\log n$ fraction of R's when R is chosen randomly. We claim that the mth iteration halts within n^{2^m} steps on at least $1/\log n - 1/n^{2^m} (> 0)$ fraction of the output strings of $G_{2^{m+1}\log n}$. Consider the following circuit C_x in which x and m are hardwired. C_x on input R simulates the mth iteration of M' for n^{2^m} steps and accepts if the computation halts. Since n^{2^m} steps of M' can be simulated by a circuit of size $O(n^{2^m}2^m)$, size of C_x is less than $O(n^{2^m}2^m + |x| + |m|)$ which is at most $n^{2^{m+1}}$. If R is chosen randomly, C_x accepts R with probability at least $1/logn$. Since any circuit of size $n^{2^{m+1}}$ can distinguish the output of $G_{2^{m+1}\log n}$ from uniform distribution with negligible probability, C_x should accept at least $1/\log n - 1/n^{2^m} (> 0)$ fraction of the output strings of $G_{2^{m+1}\log n}$. Thus there exists at least one output of $G_{2^{m+1}\log n}$, r, such that M' halts within n^{2^m} steps when r is used as random seed. Since M' produces correct answer on every random seed, the correctness is clear. The running time of the simulation can be bounded as follows: the ith iteration of the for loop takes at most $n^{2^{i+2}}$ time. Hence the total time taken is bounded by $mn^{2^{m+2}}$ which is less than $nn^{2^{m+2}}$ (since $m \leq \log n$). The following inequalities show that the running time of the simulation is polynomial on μ average.

$$\infty > \sum_x \frac{\mu'(x)}{|x|} \frac{(n^{2^m})^{1/2k}}{\log n}$$

$$\geq \sum_x \frac{\mu'(x)}{|x|}(nn^{2^{m+2}})^{1/9k}$$

Thus we can compute a witness of x in average polynomial time. $\square$

We now prove Lemma 3.12.

Proof of Lemma 3.12:

Since NP is easy on average, there exists a machine N' that runs in average polynomial time with respect to μ, and N' on input x decides whether N accepts x. Given x, M' first checks whether N accepts x, by running N'. This step does not need any randomness. If N accepts x, then M' randomly picks n^3 strings $R_1, R_2, \cdots R_{n^3}$ and simulates $M(x, R_i)$, for each i, in parallel for 2^n steps. M' halts when one of the computations $M(x, R_i)$ halts and produces correct answer. If none of the computations produce correct answer, then M' deterministically computes an accepting path in 2^n time. It is clear that for every x, M' correctly computes $f(x)$ on every random choice, and M' always halts in 2^n steps. The running time of M' can be estimated as follows.

Call a string R is *good* if M produces correct answer on (x, R). Assume that M uses n^l random bits, i.e., $|R| = n^l$. There are at least $2^{n^l}/n^2$ good Rs. Let $g_1, g_2, \cdots g_k$ be the good strings. Observe that M' uses n^{l+3} random bits. Let S' denote the set of strings of length n^{l+3}. Divide S' into $k+1$ sets $S'_0, S'_1, \cdots S'_k$ as follows. Given a string R' of length n^{l+3}, we view R' as concatenation of n^3 substrings each of length n^l. A string R' belongs to S'_0 if none of the g_i's is a substring of R'. R' belongs to $S'_i (1 \leq i \leq k)$ if R' has g_i as substring.

If we randomly chose a string of length n^l, it is good with probability at least $1/n^2$. Thus if we randomly choose n^3 strings the probability that none of them is good is at most $(1-1/n^2)^{n^3} \leq 2^{-n}$. This implies that $|S'_0|$ is at most $2^{n^{l+3}}/2^n$. The cardinality of S'_i $(1 \leq i \leq k)$ is at most $n2^{n^{l+3}-n^l}$. Note that $S' = \bigcup_i S'_i$. Finally, observe that for each R' in S'_i $(i > 0)$, the running time of M' is bounded by $n^3 t(x, g_i)$. Consider the following inequalities.

$$\begin{aligned}
\sum_{(x,R')} \frac{T_{M'}(x,R')}{|x|}\mu'(x)2^{-|R'|} &= \sum_x \frac{\mu'(x)}{|x|}\sum_{R'} T_{M'}(x,R')2^{-|R'|} \\
&\leq \sum_x \frac{\mu'(x)}{|x|}[\sum_{i\geq 0}\sum_{R'\in S'_i} T_{M'}(x,R')2^{-|R'|}] \\
&\leq \sum_x \frac{\mu'(x)}{|x|}[\sum_{R'\in S'_0} T_{M'}(x,R')2^{-|R'|} + \sum_{i>0}\sum_{R'\in S'_i} T_{M'}(x,R')2^{-|R'|}] \\
&\leq \sum_x \frac{\mu'(x)}{|x|}[2^n \times \frac{2^{n^{l+3}}}{2^n}2^{-|R'|} + \sum_{i>0}\sum_{R'\in S'_i} T_{M'}(x,R')2^{-|R'|}] \\
&\leq \sum_x \frac{\mu'(x)}{|x|}[1 + \sum_i n^3 t(x,g_i)|S'_i|2^{-|R'|}] \\
&\leq \sum_x \frac{\mu'(x)}{|x|}[1 + \sum_i n^4 t(x,g_i)2^{-n^l}] \\
&\leq constant + \sum_x\sum_i \frac{\mu'(x)}{|x|}n^4 t(x,g_i)2^{-n^l} \\
&\leq constant + \sum_{(x,R),R\ is\ good} \frac{\mu'(x)}{|x|}\frac{n^4 t(x,R)}{2^{|R|}}
\end{aligned}$$

Recall that

$$\sum_{(x,R)} \frac{\mu'(x)}{|x|} \frac{t^{1/c}(x,R)}{2^{|R|}} < \infty.$$

Since

$$\sum_{(x,R),R\ good} \frac{\mu'(x)}{|x|} \frac{n^4 t(x,R)}{2^{|R|}} \leq \sum_{(x,R)} \frac{\mu'(x)}{|x|} \frac{t^5(x,R)}{2^{|R|}},$$

it follows that there exists a constant $k \geq 5c$ such that

$$\sum_{(x,R')} \frac{T^{1/k}_{M'}(x,R')}{|x|} \mu'(x) 2^{-|R'|} < \infty.$$

□

Impagliazzo and Levin [IL90] showed that for every NP search problem (R, μ) with a p-sampleable distribution, there exists a search problem (R', ν), where ν is the uniform distribution, such that (R, μ) is randomly reducible to (R', ν). Building on this, Schuler and Watanabe [SW95] showed that if NP is easy on average, then for every language L in NP and for every p-sampleable distribution μ, the distributional problem (L, μ) can be solved in randomized average-polynomial time. Moreover, the machine that solves (L, μ) never errs. We obtain the following improvement.

Theorem 3.13 *If* NP *is easy on average, then every distributional* NP *problem with a p-sampleable distribution can be solved in average-polynomial time.*

Proof: The proof of this theorem is exactly the same as the previous theorem. Assume NP is easy on average. Let L be any language in NP and μ be a p-sampleable distribution. By the result of Schuler and Watanabe [SW95] there exists a machine M that decided L randomized average-polynomial time with respect to μ. The last step of the previous theorem shows that the computation of a machine that runs in average polynomial-time can be derandomized. Thus L can be solved in average-polynomial time. □.

4 NEXP and Polynomial-Size Circuits

Impagliazzo, Kabanets and Wigderson [IKW02] use derandomization techniques to prove the following consequence of NEXP having small circuits.

Theorem 4.1 (IKW) *If* NEXP *is in* P/poly *then* NEXP = EXP = MA.

Van Melkebeek (see [IKW02]) shows that the converse also holds.

Theorem 4.1 does not relativize: Buhrman, Fortnow and Thierauf [BFT98] exhibit an oracle relative to which MAEXP is in P/poly. If Theorem 4.1 applied in this relativized world then we would have by padding that NEE is in MAEXP but by Kannan [Kan82] even EE does not have polynomial-size circuits in any relativized world.

We show in this section that one can get some weak relativizations of Theorem 4.1.

Theorem 4.2 *For any A in* EXP*, if* NEXP^A *is in* P^A/poly *then* $\mathrm{NEXP}^A = \mathrm{EXP}^A$.

We can do better if A is complete for some level of the polynomial-time hierarchy.

Theorem 4.3 *Let A be complete for Σ_k^{p} for any $k \geq 0$. If NEXP^A is in $\mathrm{P}^A/\mathrm{poly}$ then $\mathrm{NEXP}^A = \mathrm{MA}^A = \mathrm{EXP}$.*

Theorem 4.3 has the following interesting corollary.

Corollary 4.4 *There is at most one k such that $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}}$ is in $\mathrm{P}^{\Sigma_k^{\mathrm{p}}}/\mathrm{poly}$.*

Proof: Suppose that we had a $j < k$ such that the statement in Corollary 4.4 holds for both j and k. By Theorem 4.3 we have that $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}} = \mathrm{EXP} = \mathrm{NEXP}^{\Sigma_j^{\mathrm{p}}}$ in $\mathrm{P}^{\Sigma_j^{\mathrm{p}}}/\mathrm{poly}$. Kannan [Kan82] gives a relativizable proof that Σ_2^{EXP} is not in P/poly. Relativizing this to Σ_j^{p} shows that $\Sigma_{j+2}^{\mathrm{EXP}}$ is not in $\mathrm{P}^{\Sigma_j^{\mathrm{p}}}/\mathrm{poly}$. But $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}} = \Sigma_{k+1}^{\mathrm{EXP}}$ contains $\Sigma_{j+2}^{\mathrm{EXP}}$ contradicting the fact that $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}}$ is in $\mathrm{P}^{\Sigma_j^{\mathrm{p}}}/\mathrm{poly}$. □

To prove Theorem 4.3 we first observe that the techniques of Impagliazzo, Kabanets and Wigderson do relativize to show

Lemma 4.5 *For all A, if NEXP^A is in $\mathrm{P}^A/\mathrm{poly}$ and EXP^A is in AM^A then $\mathrm{NEXP}^A = \mathrm{EXP}^A$.*

Theorem 4.1 follows immediately from Lemma 4.5 and Theorem 2.2.

Buhrman and Homer [BH92] give a relativizing proof of the following result.

Theorem 4.6 (Buhrman-Homer) *If $\mathrm{EXP}^{\mathrm{NP}}$ is in $\mathrm{EXP}/\mathrm{poly}$ then $\mathrm{EXP}^{\mathrm{NP}} = \mathrm{EXP}$.*

We show the following generalization.

Theorem 4.7 *For any $k \geq 0$, if $\mathrm{EXP}^{\Sigma_k^{\mathrm{p}}}$ is in $\mathrm{EXP}/\mathrm{poly}$ then $\mathrm{EXP}^{\Sigma_k^{\mathrm{p}}} = \mathrm{EXP}$.*

Proof: By induction in k. The case $k = 0$ is trivial. For $k > 0$, we have $\Sigma_k^{\mathrm{p}} = \mathrm{NP}^{\Sigma_{k-1}^{\mathrm{p}}}$.

Relativizing Theorem 4.6 to $\Sigma_{k-1}^{\mathrm{p}}$ gives

$$\mathrm{EXP}^{\Sigma_k^{\mathrm{p}}} \subseteq \mathrm{EXP}^{\Sigma_{k-1}^{\mathrm{p}}}/\mathrm{poly} \Rightarrow \mathrm{EXP}^{\Sigma_k^{\mathrm{p}}} = \mathrm{EXP}^{\Sigma_{k-1}^{\mathrm{p}}}. \tag{2}$$

If $\mathrm{EXP}^{\Sigma_k^{\mathrm{p}}}$ is in $\mathrm{EXP}/\mathrm{poly}$ then $\mathrm{EXP}^{\Sigma_{k-1}^{\mathrm{p}}}$ is in $\mathrm{EXP}/\mathrm{poly}$ so by induction $\mathrm{EXP}^{\Sigma_{k-1}^{\mathrm{p}}} = \mathrm{EXP}$. Plugging this into Equation (2) gives us Theorem 4.7. □

Proof of Theorem 4.3: Suppose $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}}$ is in $\mathrm{P}^{\Sigma_k^{\mathrm{p}}}/\mathrm{poly}$. Since $\mathrm{P}^{\Sigma_k^{\mathrm{p}}}$ is in EXP and $\mathrm{EXP}^{\Sigma_k^{\mathrm{p}}} \subseteq \mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}}$, by Theorem 4.7 we have $\mathrm{EXP}^{\Sigma_k^{\mathrm{p}}} = \mathrm{EXP}$. We also have EXP in $\mathrm{P}^{\Sigma_k^{\mathrm{p}}}/\mathrm{poly}$ so by Theorem 2.3 we have EXP in $\mathrm{MA}^{\Sigma_k^{\mathrm{p}}}$ and thus $\mathrm{EXP}^{\Sigma_k^{\mathrm{p}}}$ in $\mathrm{MA}^{\Sigma_k^{\mathrm{p}}}$. By Lemma 4.5 with A a complete language for Σ_k^{p}, we have $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}} = \mathrm{EXP}^{\Sigma_k^{\mathrm{p}}}$ and thus $\mathrm{NEXP}^{\Sigma_k^{\mathrm{p}}} = \mathrm{EXP} = \mathrm{MA}^{\Sigma_k^{\mathrm{p}}}$. □

To prove Theorem 4.2 we need the following relativizable results due to Impagliazzo, Kabanets and Wigderson [IKW02].

Theorem 4.8 (IKW, Theorem 6) *If $\mathrm{NEXP} \neq \mathrm{EXP}$ then for every $\epsilon > 0$,*

$$\mathrm{AM} \subseteq \mathrm{io}\text{–}[\mathrm{NTIME}(2^{n^\epsilon})/n^\epsilon].$$

Theorem 4.9 (IKW, Claim 11) *If NEXP is in P/poly then there is a universal constant d_0 such that*

$$\mathrm{NTIME}(2^n)/n \subseteq \mathrm{SIZE}(n^{d_0})$$

for all sufficiently large n.

Proof of Theorem 4.2: Suppose we have an A in EXP such that NEXP^A is in $\mathrm{P}^A/\mathrm{poly}$ and $\mathrm{NEXP}^A \neq \mathrm{EXP}^A$. Since Theorem 4.8 relativizes we have that

$$\mathrm{AM}^A \subseteq \mathrm{io}\text{–}[\mathrm{NTIME}^A(2^{n^\epsilon})/n^\epsilon].$$

We also have EXP in $\mathrm{P}^A/\mathrm{poly}$ so by Theorem 2.3 we have EXP in MA^A which is contained in AM^A and thus $\mathrm{io}\text{–}[\mathrm{NTIME}^A(2^{n^\epsilon})/n^\epsilon]$.

Since Theorem 4.9 relativizes we have

$$\mathrm{NTIME}^A(2^n)/n \subseteq \mathrm{SIZE}^A(n^{d_0})$$

for some fixed d_0 and sufficiently large n.

Combining we get

$$\mathrm{EXP} \subseteq \mathrm{io}\text{–}[\mathrm{SIZE}^A(n^{d_0})]. \tag{3}$$

Since A is a fixed language in EXP, a straightforward diagonalization argument shows that Equation (3) is false. □

5 Further Research

Perhaps one can use Theorem 3.1 to get other consequences from NP being easy on average with the ultimate goal to show that this assumption is equivalent to P = NP.

It would be nice to unify the results in Section 4 to show that for any A in EXP if NEXP^A is in $\mathrm{P}^A/\mathrm{poly}$ then $\mathrm{NEXP}^A = \mathrm{MA}^A = \mathrm{EXP}$.

Also one should look for other ways to bring in various theoretical techniques to prove other new and interesting results on derandomization.

Acknowledgments

Thanks to Peter Bro Miltersen, Russell Impagliazzo, Dieter van Melkebeek, Valentine Kabanets and Osamu Watanabe for helpful discussions and the anonymous referees for comments on the presentation.

References

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, New York, 1985.

[BCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44:193–219, 1992.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on the Theory of Computing*, pages 21–31. ACM, New York, 1991.

[BFT98] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proceedings of the 13th IEEE Conference on Computational Complexity*, pages 8–12. IEEE, New York, 1998.

[BG93] A. Blass and Y. Gurevich. Randomizing reductions of search problems. *SIAM Journal of Computing*, 22:949–975, 1993.

[BH92] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Proceedings of the 12th Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 652 of *Lecture Notes in Computer Science*, pages 116–127. Springer, Berlin, Germany, 1992.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[FRS94] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science A*, 134:545–557, 1994.

[IKW02] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential versus probabilistic time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[IL90] R. Impagliazzo and L. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 812–821. IEEE Computer Society Press, 1990.

[Imp95] R. Impagliazzo. A personal view of average-case complexity theory. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 134–147. IEEE Computer Society Press, 1995.

[IT89] R. Impagliazzo and G. Tardos. Decision versus search problems in super-polynomial time. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 222–227. IEEE, New York, 1989.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 220–229. ACM, New York, 1997.

[IW01] R. Impagliazzo and A. Wigderson. Randomness vs. time: Derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, December 2001.

[Kan82] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–56, 1982.

[KS98] J. Köbler and R. Schuler. Average-case intractability vs. worst-case intractability. In *The 23rd International Symposium on Mathematical Foundations of Computer Science*, volume 1450 of *Lecture Notes in Computer Science*, pages 493–502. Springer, 1998.

[KvM02] A. Klivans and D. van Melkebeek. Graph nonisomorhism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[Lev86] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15:285–286, 1986.

[MV99] P. Miltersen and V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 71–80. IEEE, New York, 1999.

[PS94] A. Polishchuk and D. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 194–203. ACM, New York, 1994.

[SW95] R. Schuler and O. Watanabe. Towards average-case complexity analysis of NP optimization problems. In *Proceedings of the tenth Annual Conference in complexity theory*, pages 148—161. IEEE Computer Society, 1995.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Wan97] J. Wang. Average-case computational complexity theory. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 295–328. Springer, 1997.