Towards NEXP versus BPP?

Ryan Williams*

Stanford University

Abstract. We outline two plausible approaches to improving the miserable state of affairs regarding lower bounds against probabilistic polynomial time (namely, the class BPP).

1 Introduction

In recent years, researchers have been gradually developing methods for potentially proving non-uniform circuit lower bounds against "large" complexity classes, such as nondeterministic exponential time (NEXP). These methods grew out of thinking about how to generically *derandomize* probabilistic algorithms: given an algorithm A which makes random decisions and solves a problem with high probability, can we construct a deterministic algorithm B with similar running time and equivalent behavior to A? Many non-trivial and surprising connections have been discovered between this basic problem and that of proving circuit lower bounds (e.g., [10,3,8,1,11,7,9]). Most of these papers show how circuit lower bounds can imply interesting derandomizations. Impagliazzo, Kabanets, and Wigderson proved (among many other results) an implication in the opposite direction: some derandomizations can in fact imply circuit lower bounds for NEXP:

Theorem 1 ([7]). *Suppose every problem in* promiseBPP *can be solved (even nondeterministically) in* $2^{n^{\varepsilon}}$ *time, for every* $\varepsilon > 0$. *Then* $\mathsf{NEXP} \not\subset \mathsf{P}/poly$.

That is, subexponential-time deterministic simulations of probabilistic polynomial time imply that NEXP cannot be simulated with non-uniform polynomial-size circuits. To be more precise, Theorem 1 boils down to the following claim. In the *Circuit Approximation Probability Problem* (a.k.a. CAPP), a Boolean circuit C is given, and one is asked to approximate the quantity $\Pr_x[C(x) = 1]$ within an additive factor of 1/10. IKW show that if CAPP can always be solved in $2^{n^{o(1)}}$ time on circuits of size n, then $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$.

On a circuit with n inputs, the fastest known algorithm for CAPP is simply exhaustive search, taking $\Omega(2^n)$ time. It seems that an improvement from 2^n time to $2^{n^{\varepsilon}}$ time would be major, and far from something one might hope to establish

* Supported in part by a David Morgenthaler II Faculty Fellowship, and the National Science Foundation under Grant Number CCF-1212372. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

in the near future. Fortunately, the hypothesis of Theorem 1 can be significantly weakened. It has been shown that essentially any nontrivial improvement over exhaustive search for CAPP would already yield the desired lower bounds:

Theorem 2 ([12]). *Suppose for every k, CAPP on circuits of size n^k and n inputs can be solved (even nondeterministically) in $O(2^n/n^k)$ time. Then* $\mathsf{NEXP} \not\subset \mathsf{P}/poly$.

Similarly, the paper [12] also shows how slightly faster circuit satisfiability algorithms would also entail $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$. It is possible that Theorem 2 gives a reasonable approach to proving $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$ – to prove the lower bound, it suffices to show that $\mathsf{NEXP} \subset \mathsf{P}/\text{poly}$ (a strong assumption that is algorithmic in nature) yields a nontrivial algorithm for approximating the acceptance probabilities of circuits. This approach was met with skepticism until it was shown how a variant of Theorem 2 can be applied (along with other ideas) to unconditionally prove that NEXP does not have polynomial-size ACC circuits [13]. The approach has also been recently extended to prove ACC circuit lower bounds for $\mathsf{NEXP} \cap \mathsf{coNEXP}$ as well [14].

This is all fine and good, and we are optimistic that circuit lower bounds will continue to be developed by studying the connections between circuit-analysis algorithms and circuit limitations. However, while $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$ is one of the many longstanding open questions in computational complexity theory, it is not the most embarrassing one. Strictly more embarrassingly open questions arise when we begin to discuss the status of lower bounds against probabilistic polynomial time itself. The above results show that circuit lower bounds already follow from tiny improvements on deterministic exhaustive search for problems that are trivial with randomness. However, it is still consistent with current knowledge that randomness is omnipotent! Complexity theory has not yet proved that $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$ (exponential time with an NP oracle is different from probabilistic polynomial time with two-sided error), nor have we established $\mathsf{EXP} \neq \mathsf{ZPP}$ (exponential time is different from zero-error probabilistic polynomial time), yet we believe that $\mathsf{P} = \mathsf{BPP}$ [8].[1]

Troubled by this, we have recently been thinking about how the above theorems and techniques developed for proving circuit lower bounds could potentially apply to lower bounds against BPP and ZPP.[2] This paper suggests two plausible hypotheses, one of which is significantly weaker than solving the CAPP problem in general. We prove that establishing the truth of either of these hypotheses would yield $\mathsf{NEXP} \neq \mathsf{BPP}$.

In the remainder of the paper, we assume a basic working knowledge of complexity theory, at the level of Arora and Barak's textbook [2]. For example, we

[1] A separation problem like $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$ is strictly more embarrassing than circuit lower bounds, because circuit lower bounds would already imply them, i.e., $\mathsf{EXP}^{\mathsf{NP}} \not\subset \mathsf{P}/\text{poly}$ implies $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$.

[2] It is immediate from the ACC lower bounds work that $\mathsf{NEXP} \neq \mathsf{BPACC}$, where BPACC denotes probabilistic uniform ACC with two-sided error. Moreover, $\mathsf{REXP} \neq \mathsf{BPACC}$ also holds, because $\mathsf{REXP} \subseteq \mathsf{BPP}$ would imply $\mathsf{NP} = \mathsf{RP}$ and hence $\mathsf{NEXP} = \mathsf{REXP}$. We are after bigger fish than these.

expect the reader to understand complexity classes like P/poly, ZPP, BPP, EXP, and NEXP, and possess a high-level familiarity with concepts such as probabilistically checkable proofs and pseudorandomness.

2 Derandomizing CAPP over Simple Distributions of Circuits

In this section, we will show that in order to separate NEXP and BPP, it suffices to give deterministic *heuristics* for the CAPP problem which only barely improve on exhaustive search, and only succeed with very low probability on polynomial-time samplable distributions of circuits.

Given a Boolean circuit C on n inputs, let $tt(C)$ be its *truth table*, the 2^n-bit string whose ith bit equals the value of the circuit on the ith n-bit string. First we show that $\mathsf{NEXP} = \mathsf{BPP}$ implies the existence of efficient probabilistic algorithms that can print small circuits encoding witnesses to NEXP computations. This basically follows from work of Impagliazzo, Kabanets, and Wigderson [7].

Lemma 1. *Suppose* $\mathsf{NEXP} = \mathsf{BPP}$. *Then there is a k such that, for every ℓ and for every* NEXP *verifier V accepting a language L, there is a* BPP *algorithm A_V such that, for all $x \in L$, $A_V(x,r)$ outputs (with probability at least $1 - 1/2^{|x|^\ell}$ over all r) a circuit C_x of size at most $|x|^k$ such that $V(x, tt(C_x))$ accepts.*[3]

Proof. First observe that $\mathsf{NEXP} = \mathsf{BPP}$ implies $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$. By Impagliazzo, Kabanets, and Wigderson, $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$ implies that NEXP has *succinct witness circuits*: for every $L \in \mathsf{NEXP}$, for every verifier algorithm V for L, and every $x \in L$, there is a poly$(|x|)$-size circuit C_x such that $V(x, tt(C_x))$ accepts. We want to show that these C_x can be constructed in BPP, under the assumptions.

For every NEXP verifier V that accepts a language $L \in \mathsf{NEXP}$, there is a k and an exponential-time algorithm $A(x,i)$ which given a string x and index i, enumerates all possible circuits D of size $|x|^k + k$, checking if $V(x, tt(D))$ accepts. If this ever happens, $A(x,i)$ then outputs the ith bit of the encoding of D (otherwise, let $A(x,i)$ output 0).

Under $\mathsf{EXP} = \mathsf{BPP}$, there must exist a BPP algorithm A' equivalent to A: given (x,i), A' outputs (with high probability) the ith bit of such a circuit D. By probability amplification (repeating A' for poly$(|x|)$ times, for each $i = 1, \ldots, |x|^k + k$, and taking the majority answer for each i), there is a probabilistic polynomial-time algorithm A'' which given $x \in L$ prints a circuit D encoding a witness for x, with $1/2^{|x|^\ell}$ probability of error. Let $A_V = A''$. $\square$

Our next ingredient is the *PCPs of Proximity* of Ben-Sasson *et al.*, which imply succinct PCPs for NEXP.

[3] An algorithm V is a *verifier* for $L \in \mathsf{NEXP}$ if there is a k such that for every string x, we have $x \in L$ if and only if there is a y of length $2^{|x|^k}$ such that $V(x,y)$ accepts within $O(2^{|x|^k})$ steps.

Theorem 3 ([4]). *Let $T : \mathbb{Z}^+ \to \mathbb{Z}^+$ be a non-decreasing function. Then for every $s > 0$ and every language $L \in \mathsf{NTIME}[T(n)]$ there exists a PCP verifier $V(x, y)$ with soundness s, perfect completeness, randomness complexity $r = \log_2 T(|x|) + O(\log\log T(|x|))$, query complexity $q = poly(\log T(|x|))$, and verification time $t = poly(|x|, \log T)$. More precisely:*

- *V has random access to x and y, uses at most r random bits in any execution, makes q queries to the candidate proof y. and runs in at most t steps.*
- *If $x \in L$ then there is a string y of length $T(|x|)\log^{O(1)} T(|x|)$ such that $\Pr[V(x, y) \text{ accepts}] = 1$.*
- *If $x \notin L$ then for all y, $\Pr[V(x, y) \text{ accepts}] \le s$.*

A standard perspective to take in viewing PCP results is to think of the polynomial-time verifier V as encoding an exponentially long constraint satisfaction problem, where each setting of the random bits in the verifier yields a new constraint. For our purposes, we will want $T(n)$ to be 2^n, and s to be an arbitrarily small constant (e.g., $1/10$). Then Theorem 3 gives a PCP verifier with $n + O(\log n)$ bits of randomness, $\text{poly}(n)$ verification time, and $\text{poly}(n)$ query complexity. By converting this polynomial-time verifier to a polynomial-size circuit that takes randomness as input, we can produce a reduction from every $L \in \mathsf{NTIME}[2^n]$ to the so-called Succinct-CSP problem, with the properties:

- Every instance x of L is reduced to a $\text{poly}(n)$-size circuit C_x.
- The truth table of C_x, $tt(C_x)$, encodes a constraint satisfaction problem with $2^n n^{O(1)}$ constraints and variables.
- Each constraint in $tt(C_x)$ contains $\text{poly}(n)$ variables and can be evaluated in $\text{poly}(n)$ time.
- If $x \in L$ then the CSP $tt(C_x)$ is satisfiable.
- If $x \notin L$ then for every variable assignment to the CSP, at most an s-fraction of the constraints are satisfied.

A *polynomial-time samplable distribution* $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$ on strings has the property that there is an n^k-time algorithm $A(1^n, r)$ such that for all n, the probability of drawing a string $x \in \{0,1\}^{\le n^k}$ from $\mathcal{D}_n$ is exactly

$$\Pr_{r \in \{0,1\}^{n^k}}[A(1^n, r) = x].$$

That is, the polynomial time algorithm A perfectly models the distribution $\mathcal{D}$. (Note that our ensemble of distributions $\{\mathcal{D}_n\}$ is a bit different from standard practice: $\mathcal{D}_n$ does not contain only strings of length n but strings of length up to n^k for some fixed k.) A canonical example of a polynomial-time samplable distribution is the *uniform* distribution on strings. Polynomial-time samplable distributions are central to the study of average-case complexity theory.

We consider *deterministic errorless heuristics* for the CAPP problem, which may print *SAT*, *UNSAT*, or *FAIL*. We pose very weak requirements on the heuristic: if the satisfiable fraction of assignments to the circuit is at least $9/10$, the heuristic must output *SAT* or *FAIL*; when the circuit is unsatisfiable, the algorithm always outputs *UNSAT* or *FAIL*.

Finally, we can state the main result of this section:

Theorem 4. *Suppose for every k, and every polynomial-time samplable distribution $\mathcal{D}$, the CAPP problem on circuits of size n^k and n inputs can be solved in $O(2^n/n^k)$ time by a deterministic heuristic H (possibly dependent on $\mathcal{D}$) such that $\Pr_{E \in \mathcal{D}_n}[H(E) \neq \mathit{FAIL}] > 1/2^n$. Then* $\mathsf{NEXP} \neq \mathsf{BPP}$.

That is, to separate NEXP from BPP, it suffices to design for every *polynomial-time samplable distribution of circuits* a heuristic for the CAPP problem which barely improves over exhaustive search, and only succeeds on a negligible measure of circuits from the given distribution. This is a significantly weaker requirement than designing a worst-case CAPP solver: here we get to see the efficient distribution of circuits that the adversary will construct, and we are allowed to fail on the vast majority of circuits output by the adversary.

It is useful to put Theorem 4 in perspective with another result on average-case complexity. Buhrman, Fortnow, and Pavan [5] have shown that if every problem in NP can be solved in polynomial time for every polynomial-time samplable distribution, then $\mathsf{P} = \mathsf{BPP}$. That is, if all NP problems can be efficiently solved in this way, we can separate EXP from BPP. Theorem 4 shows that a significantly weaker assumption suffices to separate NEXP from BPP.

Proof of Theorem 4. Let $L \subseteq \{1^n \mid n \geq 0\}$ be chosen so that $L \in \mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n]$. (Such languages exist, due to the nondeterministic time hierarchy of Žák [15].) By Theorem 3, there is a polynomial-time reduction from L to Succinct-CSP, which outputs a circuit C_{1^n} on $n + O(\log n)$ inputs. If $\mathsf{NEXP} = \mathsf{BPP}$ then Lemma 1 says that for all NEXP languages L and verifiers V for L, there is a probabilistic polynomial-time algorithm A that prints valid witness circuits for V, with probability of error at most $1/3^n$ on inputs of size n. Let V be the verifier which, on input 1^n, tries to check that its certificate is a satisfying assignment for $tt(C_{1^n})$. Then A is then a probabilistic polynomial-time algorithm that (with high probability) on input 1^n prints a circuit D_{1^n} such that $tt(D_{1^n})$ is a satisfying assignment for $tt(C_{1^n})$, for $1^n \in L$. We can think of A as a deterministic algorithm A' which takes 1^n as one input, and a poly($|x|$)-bit random string r as a secondary input, where the overall output of A' is determined by the randomness r.

We design a polynomial-time samplable distribution $\mathcal{D}$ of circuits, as follows. Given 1^n, our polynomial-time algorithm B first runs the reduction of Theorem 3 to produce a circuit C_{1^n} such that $1^n \in L$ if and only if $C_{1^n} \in$ Succinct-CSP. Then B picks a random seed r and runs $A'(1^n, r)$ which (with probability at least $1 - 1/3^n$) prints a circuit D_{1^n} encoding a satisfying assignment for the Succinct-CSP instance C_{1^n}. Using multiple copies of the two circuits C_{1^n} and D_{1^n}, one can construct a polynomially larger circuit E with $n + O(\log n)$ inputs and the following properties:

- If D_{1^n} encodes a satisfying assignment to $tt(C_n)$ then E is unsatisfiable.
- If D_{1^n} does not encode a satisfying assignment to $tt(C_n)$ then E is satisfiable on at least $9/10$ of its possible inputs.

(For a proof of this construction, see [12].) Algorithm B then outputs E.

Suppose there is a heuristic H for the CAPP problem which runs in deterministic $O(2^n/n^k)$ time for all desired k, outputs *SAT* or *FAIL* when the fraction of assignments to the circuit which are satisfying is at least 9/10, *UNSAT* or *FAIL* when the fraction is 0, and on circuits randomly drawn from $\mathcal{D}$, H outputs *FAIL* with probability at most $1 - 1/2^n$. We wish to give a nondeterministic algorithm N which recognizes the language L in nondeterministic time $O(2^n/n)$, contradicting the choice of L.

On an input 1^n, N nondeterministically guesses a random seed r for the algorithm A', and runs $B(1^n)$ with this choice of seed r for A'. B outputs a circuit E, which is then fed to H. If H prints *UNSAT* then N accepts, otherwise N rejects.

Note that N can be made to run in time $O(2^n/n)$: although the circuit E has $n+O(\log n)$ inputs, we can choose k to be larger than the constant c in the big-O term, so that the heuristic H runs in time $O(2^{n+c\log n}/(n+c\log n)^k) \leq O(2^n/n)$.

We now argue that N is correct. If $1^n \in L$, then on at least $(1 - 1/3^n)$ of the seeds r, $A'(1^n, r)$ outputs a circuit D_{1^n} encoding a satisfying assignment for $tt(C_{1^n})$. When such an r is guessed, the circuit E drawn from $\mathcal{D}_n$ is unsatisfiable; hence in this case, at least a $(1 - 1/3^n)$ measure of the circuits in $\mathcal{D}_n$ are unsatisfiable. The satisfiability algorithm H outputs *FAIL* on less than $1/2^n$ of the circuits drawn from $\mathcal{D}_n$. Hence there is a random seed $r^\star$ such that the circuit $E^\star$ output by $\mathcal{D}_n$ is unsatisfiable, and $E^\star$ is declared *UNSAT* by H. Note N accepts provided that such an $r^\star$ is guessed by N.

If $1^n \notin L$ then for all seeds r, the circuit D printed by $A'(1^n, r)$ cannot encode a satisfying assignment for $tt(C_{1^n})$, so the resulting circuit E is always satisfied by at least 9/10 of its possible input assignments. The heuristic H on circuit E will always print *SAT* or *FAIL* in this case, and N rejects in either of the two outcomes. □ □

So, NEXP $\neq$ BPP would follow from CAPP heuristics that barely beat exhaustive search and output *FAIL* on all but a small fraction of inputs. Should we expect the existence of such heuristics to be easier to establish than worst-case CAPP algorithms? The answer is not clear. However it does seem plausible that one may be able to show NEXP = BPP itself implies heuristic algorithms for CAPP, which would be enough to prove the desired separation result.

3 Pseudorandomness for Deterministic Observers

Our second direction for randomized time lower bounds is a simple reflection on Goldreich and Wigderson's work regarding pseudorandomness with efficient deterministic observers [6]. Informally, when one defines pseudorandomness, we have a *pseudorandom generator* (a function that maps short strings to long strings) along with a class of *observers* (efficient algorithms), and the generator is said to be pseudorandom to that class if every observer exhibits extremely similar behavior on the uniform distribution and the distributions of outputs of the generator.

An alternative way to define pseudorandomness is via *unpredictability*: a generator is unpredictable if no observer, given i bits of a random output from the generator, can predict the $(i+1)$st bit with probability significantly better than $1/2$, for all i. A central theorem in the theory of pseudorandomness is that the unpredictability criterion and the pseudorandomness criterion are essentially equivalent, when the class of observers is the set of probabilistic polynomial-time algorithms. That is, a generator which is unpredictable is also pseudorandom, and vice-versa.

What if the class of observers consists of *deterministic* polynomial-time algorithms? Then the connections between pseudorandomness and unpredictability are actually open problems. For every polynomial $p(n)$, Goldreich and Wigderson give an explicit distribution computable in time poly$(p(n))$ which is unpredictable for all deterministic $p(n)$-time observers, by applying pairwise independent distributions in a clever way. They pose as an open problem whether their distribution is *pseudorandom* for all deterministic $p(n)$-time observers. We show that exhibiting an exponential-time computable distribution that is pseudorandom to linear-time observers implies $\mathsf{EXP} \neq \mathsf{BPP}$. In fact, it suffices to construct an exponential-time generator G that is given the code of a particular linear-time observer A, and G only has to fool A.

Theorem 5. *Suppose for every deterministic linear-time algorithm A, and all $\varepsilon > 0$, there is a $\delta > 0$ and algorithm G that runs in $O(2^m)$ time on inputs of length m, produces outputs of length $m^{1/\varepsilon}$, and*

$$\left|\Pr_{x\in\{0,1\}^n}[A(x)=1] - \Pr_{y\in\{0,1\}^{n^\varepsilon}}[A(G(y))=1]\right| < 1/2-\delta.$$

Then $\mathsf{EXP} \neq \mathsf{BPP}$.

Proof. Assume $\mathsf{EXP} = \mathsf{BPP}$. Choose a language $L \subseteq \{1^n \mid n \geq 0\}$ such that $L \in \mathsf{TIME}[2^n] \setminus \mathsf{TIME}[2^{n/2}]$ (which can be easily derived by direct diagonalization). By assumption, and by amplification, there is a *deterministic* n^k-time algorithm $B(\cdot,\cdot)$ such that

– If $1^n \in L$, then $\Pr_{x\in\{0,1\}^{n^k}}[B(1^n,x)=1] > 1-1/2^n$.
– If $1^n \notin L$, then $\Pr_{x\in\{0,1\}^{n^k}}[B(1^n,x)=1] < 1/2^n$.

Define the algorithm $A(x) = B(1^{|x|^{1/k}}, x)$, which runs in linear time. Let $\varepsilon = 1/(2k)$, and suppose there were a function G satisfying the hypotheses of the theorem for algorithm A and ε. Then we could simulate L in $\mathsf{TIME}[2^{O(n^{1/2})}]$ (a contradiction), as follows: given 1^n, compute the $O(2^{n^{1/2}})$-size set of strings $S = \{G(y) \mid y \in \{0,1\}^{n^{1/2}}\} \subseteq \{0,1\}^{n^k}$ in $O(2^{2n^{1/2}})$ time, then output the majority value of $A(x)$ over all $x \in S$, in $2^{n^{1/2}}\text{poly}(n)$ time. This $O(2^{2n^{1/2}})$ time algorithm decides L, because if $1^n \in L$ then $\Pr_{y\in\{0,1\}^{n^{1/2}}}[A(G(y))=1] > 1/2+\delta/2$, and if $1^n \notin L$ then $\Pr_{y\in\{0,1\}^{n^{1/2}}}[A(G(y))=1] < 1/2-\delta/2$. □

The above simple result shows that the ability to (slightly) fool deterministic linear-time algorithms with exponential-time generators is already enough to separate EXP and BPP. The basic idea can be easily extended in several different ways. For one, we could make the generator G very powerful, and still derive a breakthrough lower bound: if G were also allowed to have free oracle access to the SAT problem (asking exponentially long NP queries about the behavior of A) in the above hypothesis, we could separate $\mathsf{EXP}^{\mathsf{NP}}$ from BPP. For another:

Theorem 6. *Suppose for every deterministic linear-time algorithm A, and all $\varepsilon > 0$, there is an algorithm G that runs in $O(2^m)$ time on inputs of length m, produces outputs of length $m^{1/\varepsilon}$, and for every n, if $\Pr_{x\in\{0,1\}^n}[A(x) = 1] > 1 - 1/n$ then there is a $y \in \{0,1\}^{n^\varepsilon}$ such that $A(G(y)) = 1$. Then $\mathsf{EXP} \neq \mathsf{ZPP}$.*

That is, we only require that, when A accepts the vast majority of n-bit strings, the algorithm G prints *at least one* n-bit string (out of 2^{n^ε}) that is accepted by A. The proof is analogous.

4 Conclusion

In this short paper, we outlined two potential directions for attacking the basic separation problems between exponential time and probabilistic polynomial time. In general we believe that proving separations like $\mathsf{NEXP} \neq \mathsf{BPP}$ are not impossible tasks, but a couple of new ideas will probably be needed to yield the separation. The reader should keep in mind that such separation results will require non-relativizing techniques (there are oracles relative to which $\mathsf{NEXP} = \mathsf{BPP}$ and $\mathsf{NEXP} \neq \mathsf{BPP}$), so no simple black-box arguments are expected to yield new lower bounds against BPP. However, in this day and age, non-relativizing tools are not so hard to come by.

References

1. Andreev, A.E., Clementi, A.E.F., Rolim, J.D.P.: Worst-case hardness suffices for derandomization: A new method for hardness-randomness tradeoffs. TCS: Theoretical Computer Science 221(1-2), 3–18 (1999)
2. Arora, S., Barak, B.: Computational Complexity - A Modern Approach. Cambridge University Press (2009)
3. Babai, L., Fortnow, L., Nisan, N., Wigderson, A.: BPP has subexponential time simulations unless EXPTIME has publishable proofs. Computational Complexity 3(4), 307–318 (1993)
4. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Short PCPs verifiable in polylogarithmic time. In: IEEE Conference on Computational Complexity, pp. 120–134 (2005)
5. Buhrman, H., Fortnow, L., Pavan, A.: Some results on derandomization. Theory Comput. Syst. 38(2), 211–227 (2005)
6. Goldreich, O., Wigderson, A.: On pseudorandomness with respect to deterministic observers. In: ICALP Satellite Workshops 2000, pp. 77–84 (2000)
7. Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: Exponential time vs. probabilistic polynomial time. JCSS 65(4), 672–694 (2002)

8. Impagliazzo, R., Wigderson, A.: P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In: Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, pp. 220–229 (1997)
9. Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Computational Complexity 13(1-2), 1–46 (2004)
10. Nisan, N., Wigderson, A.: Hardness vs randomness. JCSS 49(2), 149–167 (1994)
11. Sudan, M., Trevisan, L., Vadhan, S.: Pseudorandom generators without the xor lemma. Journal of Computer System Sciences 62(2), 236–266 (2001)
12. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. In: ACM Symposium on Theory of Computing, pp. 231–240 (2010)
13. Williams, R.: Non-uniform ACC circuit lower bounds. In: IEEE Conference on Computational Complexity, pp. 115–125 (2011)
14. Williams, R.: Natural proofs versus derandomization. In: ACM Symposium on Theory of Computing (to appear, 2013)
15. Žák, S.: A Turing machine time hierarchy. Theoretical Computer Science 26(3), 327–333 (1983)