Electronic Colloquium on Computational Complexity, Report No. 38 (2002)

Resource Tradeoffs and Derandomization

Rahul Santhanam,
Department of Computer Science,
University of Chicago.
E-mail:rahul@cs.uchicago.edu

ISSN 1433-8092

Abstract

We consider uniform assumptions for derandomization. We provide intuitive evidence that BPP can be simulated non-trivially in deterministic time by showing that (1) $P \not\subseteq i.o.i.POLYLOGSPACE \Rightarrow BPP \subseteq SUBEXP$ (2) $P \not\subseteq i.o.i.SUBPSPACE \Rightarrow BPP = P$. These results extend and complement earlier work of Sipser, Nisan-Wigderson and Lu.

We show similar tradeoffs between simulation of nondeterministic time by nondeterministic space and simulation of randomized algorithms by nondeterministic time. Thse results may be useful in settling the question $BPP \stackrel{?}{=} NEXP$. We state two conjectures under which $BPP \neq NEXP$ - (1) An i.o.i. (infinitely often) analogue of Lautemann's theorem holds (2) Every language in BPP can be reduced to SAT using nondeterministic truth table reductions where the entire set of queries can be computed in nondeterministic polynomial time from the input. Unlike previous approaches, our approach does not seem to require proving circuit lower bounds for $NEXP$.

Finally, we give uniform assumptions under which there is a strict hierarchy for randomized polynomial time and randomized time can be simulated nontrivially by randomized space.

1 Introduction

Much research in complexity theory over the past decade has focussed on derandomization. Techniques are known for eliminating randomness in many important randomized algorithms, but the question of whether it is possible to simulate every polynomial-time randomized algorithm non-trivially in deterministic time, i.e., more efficiently than the standard exponential time simulation, is still open. One approach to this problem is via the notion of pseudo-random generators, which are efficient deterministic procedures expanding short random seeds to strings that "look" random to algorithms from a given class. Given the existence of a pseudo-random generator, it is possible to simulate randomized algorithms efficiently by enumerating the short seeds, running the algorithm on the ouputs of the generator on the seeds, and taking the majority vote.

Following on work by Yao [Yao82] and Blum and Micali [BM84], Nisan and Wigderson showed in a landmark paper [NW94] that the existence of pseudo-random generators is related to the average-case hardness of Boolean functions computable in exponential time. Given a Boolean function such that there is no small circuit computing it on most inputs, we can produce an efficient pseudo-random generator; conversely, a pseudo-random generator

can be used to construct a hard Boolean function. Much work ([BFNW93], [IW97] has gone towards weakening the average-case hardness requirement to a worst-case hardness requirement. The culmination of these efforts is the recent paper by Umans which demonstrates an essentially optimal conversion of hardness to pseudo-randomness.

In this paper, we are concerned with derandomization under uniform assumptions. Very few lower bounds are known for problems in non-uniform models, hence it seems unlikely that we will be able to derandomize probabilistic classes by exhibiting a hard function in the near future. More results are known for uniform classes, so it is worthwhile to find assumptions on uniform classes under which derandomization can be carried out. Of course, our techniques will depend heavily on the powerful machinery developed by Nisan, Wigderson, Impagliazzo et al.

The seminal paper of Sipser [Sip88] first demonstrated a connection between time-space tradeoffs and derandomization. Sipser showed that under a hypothesis about the existence of a certain kind of explicit disperser, later proved by Saks, Srinivasan and Zhou [SSZ98], there is a constant ϵ such that: $DTIME(t) \subseteq i.o.DSPACE(t^\epsilon)$ for all polynomially bounded t, or $P = RP$. Essentially, the disperser hypothesis implies efficient error reduction for RP, and if the time-space tradeoff does not hold, this error reduction can be used to show that an efficient hitting-set generator exists, which can be used to derandomize RP. The results of Andreev, Clementi and Rolim [ACR96] imply that the second clause in Sipser's theorem can be strengthened to $P = BPP$.

Nisan and Wigderson [NW94] demonstrated a relationship between time-space tradeoffs for exponential time classes and derandomization. Unlike Sipser's tradeoffs, their tradeoffs worked even at the low end, i.e., they were able to show $BPP \subseteq SUBEXP$ unless $EXP \subseteq i.o.PSPACE$. In a different direction, Impagliazzo and Wigderson [IW98] showed a remarkable "gap" result for the simulation of BPP by deterministic time - either $BPP = EXP$ or BPP can be simulated in heuristic subexponential time infinitely often. Kabanets [Kab00] considered simulations of randomized algorithms that are successful against *uniform* adversaries. He showed that for every RP algorithm, there is a simulation in zero-error subexponential time (ZPSUBEXP) that is successful infinitely often against adversaries in $ZPSUBEXP$. Lu [Lu00] generalized some of the results of Kabanets and Nisan-Wigerson.

Our main contribution in this paper is to prove results analogous to those of Nisan and Wigderson and Lu for polynomial time classes. These results provide strong intuitive evidence for derandomization.

We cannot claim that our results are strictly stronger than those of Nisan-Wigderson and Lu because we use a different concept of a langauge being contained in a class infinitely often (i.o.). However the notion we use suffices for proving hierarchy theorems and is therefore of interest if the purpose is to separate complexity classes. Our results are stronger than those of Sipser, since we show a much wider range of tradeoffs using the same concept of i.o. as him.

The structure of the paper is as follows: In Section 2, we define the notions and notation we use. In Section 3, we prove the main simulation lemma and show some consequences. In Section 4, we examine the situation for nondeterministic time. In Section 5, we consider uniform assumptions under which we can draw conclusions about the fine structure of randomized classes.

2 Preliminaries

The definitions of deterministic, nondeterministic and probabilistic resource-bounded classes are standard and can be found in [BDG88] and [BDa90].

We use two different concepts of what it means for a language to be contained in a class infinitely often. If L is a language and C is a deterministic (resp. nondeterministic) class defined by a set of resource constraints, we say $L \in i.o.i.C$ if there is a deterministic (resp. nondeterministic) machine deciding L that on infinitely many inputs satisfies the resource constraints defining C. If L is a language and C is a class, we say $L \in i.o.C$ if there is a language $L^{'} \in C$ such that $\{n : L \cap \{0,1\}^n = L^{'} \cap \{0,1\}^n\}$ is infinite.

We say a complexity class C is *easy* if $C \subseteq P$. We say a complexity class C is *somewhat easy* if C is contained in deterministic quasi-polynomial time.

A pseudo-random generator g is a sequence of functions $\{g\}_n, n \geq 1$ such that g_n maps $\{0,1\}^{s(n)}$ to $\{0,1\}^n$ for some $s(n) < n$, and for all circuits C of size less than n,

$| \Pr_{\rho}[C(\rho) = 1] - \Pr_{\sigma}[C(\sigma = 1] | < \frac{1}{n}$

where ρ is uniformly distributed over $\{0,1\}^n$ and σ is uniformly distributed over $\{0,1\}^{s(n)}$. The pseudo-random generators we consider will be computable in linear exponential time. A pseudo-random generator is said to be computable in space S if the ith bit of the output of the generator can be computed in space $S(s(n) + \log(i))$, for each $i, 1 \leq i \leq n$.

We define a concept of *mild non-deterministic truth table reduction.* A language L is in NP^A_{mtt}, where A is a complexity class, if there is a non-deterministic oracle Turing machine M, that given an input x, first computes

in nondeterministic polynomial time, the entire set of queries $Q(x)$ (which is therefore of polynomial size) that it is allowed to make during the computation. Let the string $y_{Q(x)}$ represent the answers to these queries. M runs a non-deterministic computation on input $(x, y_{Q(x)})$ to determine if $x \in L$.

Kabanets [Kab00] defines some notions of what it means for a simulation to be successful against a uniform adversary. A *refuter* is a deterministic Turing machine that, on input 1^n, outputs a string of length n. Given languages L and L', and a deterministic class A of refuters, L and L' are *A-indistinguishable* if there is no machine $M \in A$ such that $M(1^n) \in L \triangle L'$ for infinitely many n. Given a complexity class C, $pseudo_A - C$ is the class of languages L for which there is a language L' in C such that L and L' are A-indistinguishable.

We consider a weaker concept of refutation. A *weak refuter* is a deterministic Turing machine that, on input 1^n, outputs a polynomial-size set of strings, each of which is of length n. Given languages L and L', and a deterministic class A of weak refuters, L and L' are *weakly A-indistinguishable* if there is no machine $M \in A$ such that $M(1^n) \cap (L \triangle L')$ is nonempty for infinitely many n. Given a complexity class C, $quasi_A - C$ is the class of languages L for which there is a language L' in C such that L and L' are weakly A-indistinguishable.

We can also define weak refuters that work almost everywhere rather than infinitely often. $[io - quasi_A] - C$ is the class of languages indistinguishable from some language in a class C by weak refuters of this kind belonging to a class A. We have only discussed deterministic weak refuters but we can also define non-deterministic weak refuters as non-deterministic machines that accept on every input of the form 1^n and output a distinguishing set of strings on every accepting computation path.

Clearly, for each A and C, $C \subseteq quasi_A - C \subseteq pseudo_A - C$. The proofs of the following propositions are straightforward.

Proposition 1 If $C \subseteq D$, then $quasi_A - C \subseteq quasi_A - D$.

Proposition 2 $quasi_A - quasi_A - C = quasi_A - C$.

3 Main simulation

Like Nisan-Wigderson, we essentially give a technique for translating non-uniform upper bounds into good simulations of time by space. By using a method for compressing configurations of Turing machines, we can

get the simulation to work in the polynomial-time domain rather than the exponential-time domain as in Nisan-Wigderson.

Let M be a DTM operating in time T. Assume, without loss of generality, that the tape alphabet of M is binary and that M has k tapes. On input x, let $C^i_{M,x}(t)$ be the contents of the ith tape of M at time t. Let $C_{M,x}(t)$ be the concatenation, over $1 \leq i \leq k$, of $C^i_{M,x}(t)$. The Boolean function $f_{M,x}$ is defined as the function whose truth table is the concatenation $C_{M,x}$, over $1 \leq t \leq T$, of $C_{M,x}(t)$.

Lemma 3 If M is a DTM operating in linear time and $S = \Omega(\log(n))$ is a space-constructible function, $L(M) \in i.o.DSPACE(S\log(S))$ or for almost all inputs x, $f_{M,x}$ has no circuits of size S.

Proof Assume that M has k tapes and a binary tape alphabet, and that M operates in time cn on an input of length n, for some constant c. Assume wlog that both c and k are powers of 2. We define a DTM M' that simulates M in small space. If the simulation fails a.e., we can obtain a lower bound on the circuit complexity of f_M.

M' works as follows: it represents the contents of each tape of M by a circuit accepting the function defined by the tape contents considered as a truth table. Let the encoding of the contents of the ith tape of M after simulating t steps of M be $E^i_{M,x}(t)$. Initially, the tapes are blank, and a small encoding of a circuit accepting none of its inputs can be found easily. We need to specify how M' passes from one encoding to another using only a small amount of space. On separate tapes, M' records the state and tape head positions of M. This costs space $O(\log n)$. To simulate the $t+1$th step of M, M' changes its record of state and tape head positions as specified by the transition table of M. Also, for each tape i of M, it cycles through all circuits A of size S, until it finds one that corresponds to the new contents of that tape. To do this, it checks that the circuit encoded by $E^i_{M,x}(t)$ agrees with the circuit A on every input except the one corresponding to the position of the ith tape head of M at time t. The two circuits should disagree or agree on the input corresponding to this tape position depending on whether the tape symbol at this position is changed or not by M during its $t+1$st time step. Checking agreement or disagreement on an input takes space $O(S\log(S))$, as a circuit of size S can be represented in space $S\log(S)$ and simulated in the same amount of space. As for the input head of M', it moves in the same manner as the input head of M.

If there is a tape i of M and a time t such that M' cannot find a

small encoding of $C^i_{M,x}(t)$, we can show that $f_{M,x}$ has no circuits of size S. Note that $f_{M,x}$ is a function defined on inputs of size $2\log(n) + O(1)$. If $f_{M,x}$ had circuits of size S, we could freeze all the input bits except those corresponding to $C^i_{M,x}(t)$ and obtain a circuit of size S encoding $C^i_{M,x}(t)$, contradicting the assumption that the simulation failed at this stage.

Either the simulation succeeds on infinitely many inputs or it fails on almost every input. If the simulation succeeds infinitely often, we have $L(M) \in i.o.i.DSPACE(S\log(S))$. If the simulation fails on almost every input, the function $f_{M,x}$ has circuit complexity greater than S. Thus at least one of these statements must hold. □

Theorem 4 ([BFNW93]) If there is a function $f \in E$ such that f does not have polynomial-size circuits, then $BPP \subseteq SUBEXP$.

Theorem 5 $P \subseteq i.o.i.POLYLOGSPACE$ or $BPP \subseteq SUBEXP$.

Proof At least one of the following statements must hold - (1) For each DTM M operating in linear time, there is a constant d such that the simulation of Lemma 3 succeeds with $S = \log(n)^d$ and hence $L(M) \in i.o.i.POLYLOGSPACE$ or (2) There exists a DTM M operating in linear time such that, for all constants d, the simulation of Lemma 3 fails with $S = O(\log(n)^d)$ and hence f_M does not have polynomial-size circuits. If statement (1) holds, by a simple translation argument, we have $P \subseteq i.o.i.POLYLOGSPACE$. If statement (2) holds, since the function $g(n) = f_{M,1^n} \in E$ and does not have polynomial-size circuits, by Theorem 4, $BPP \subseteq SUBEXP$. □

Theorem 5 states that, if there is no strong simulation of polynomial time by space, then a low-end simulation of BPP is possible. By varying the parameter S in Lemma 3 and using different hardness-randomness tradeoffs, we can obtain a range of tradeoffs between the strength of the simulation of time by space and the strength of the simulation of randomness by time.

Theorem 6 ([IW97]) If there is a function $f \in E$ such that f does not have circuits of size $2^{\epsilon n}$, for some $\epsilon > 0$, then $BPP = P$.

Theorem 7 For each $\epsilon > 0$ and polynomially bounded t, $DTIME(t) \subseteq i.o.i.DSPACE(t^\epsilon)$, or $BPP = P$.

Proof Analogous to the proof of Theorem 5, using Lemma 3 with $S =$

n^ϵ. □

Theorem 5 and Theorem 7 are derandomizations of BPP under the assumption that deterministic time cannot be simulated infinitely often in small space. We would prefer to have derandomizations under the weaker assumption that there are no small space simulations almost everywhere. Our techniques do not allow us to achieve this, but we can obtain derandomizations under the assumption that there are no small space simulations that fool uniform adversaries almost everywhere.

Theorem 8 $P \subseteq quasi_P - POLYLOGSPACE$, or $BPP \subseteq i.o.SUBEXP$.

Proof Consider the simulation of Lemma 3. If for every polynomial-time machine M accepting an infinite language L, every refuter N in P succeeds only finitely often, we have $P \subseteq quasi_P - POLYLOGSPACE$. Otherwise, there is a polynomial-time machine M and a refuter N in P that succeeds infinitely often. Now we have a strategy for simulating BPP in $i.o.SUBEXP$: for each n, concatenate $C_{M,x}$ for $x \in N(1^n)$ to produce a string f_n and use f_n as hardness source for a pseudo-random generator. Since N succeeds infinitely often, this strategy succeeds infinitely often and we have $BPP \subseteq i.o.SUBEXP$. □

Analogously, we can prove:

Theorem 9 For each $\epsilon > 0$ and polynomially bounded t, $DTIME(t) \subseteq quasi_{QP} - DSPACE(t^\epsilon)$, or $BPP \subseteq i.o.P$.

The simulation in Lemma 3 does not run in polynomial time. If we allow the simulating machines to be nondeterministic, then we can modify the simulation to run in polynomial time.

Let NSC be the class of languages accepted by nondeterministic machines running simultaneously in polynomial time and polylogarithmic space.

Theorem 10 $P \subseteq i.o.i.NSC$ or $BPP \subseteq SUBEXP$.

Proof We modify the simulation of Lemma 3 so that the simulating machine is nondeterministic and runs in polynomial time. We combine this modified version of Lemma 3 with Theorem 4 to prove the theorem, analogous to the proof of Theorem 5.

The only change in the simulation is that, instead of cycling over all circuits of small size when trying to find an encoding of a new configuration, the simulating nondeterministic machine simply guesses such a circuit. Then it verifies that the circuit is a correct encoding in the same way as in Lemma 3, by cycling over all possible inputs to the circuits. Since the input is only of size $\log(n)$, the verification can be done in polynomial time. If the verification fails, the simulating machine rejects. Clearly, only polynomial time is required for the simulation as a whole, yielding the theorem. □

Actually, we can use our techniques to derandomize the class AM, on the hypothesis that polynomial time cannot be simulated by polylogarithmic space. Klivans and van Melkebeek [KvM99] showed that pseudorandom generators secure against nondeterministic adversaries follow from hardness conditions on SAT-oracle circuits. We can modify Lemma 3 to generate a function that does not have small SAT-oracle circuits, if the simulation of time by space fails. The property of SAT we use here is that it is in linear space.

Theorem 11 ([KvM99]) If there is a function $f \in NE \cap co - NE$ such that f has no SAT-oracle circuits of polynomial size, for some ϵ, then $AM \subseteq NSUBEXP$.

Theorem 12 $P \subseteq i.o.i.POLYLOGSPACE$ or $AM \subseteq NSUBEXP$.

Proof We modify the simulation of Lemma 3 and then combine the modified lemma with Theorem 11 to obtain the theorem.

We encode the configurations of a linear-time deterministic machine with SAT-oracle circuits, rather than ordinary circuits. Each time a query to SAT is made, we run the linear-space algorithm for deciding SAT to answer the query. The size of a query cannot exceed the size of the circuit, hence the space requirements of the simulation remain the same. If the simulation fails almost everywhere, we obtain a function in E that does not have polynomial size SAT-oracle circuits and we can apply Theorem 11. □

We can obtain a conditional low-end simulation (i.e., a complete derandomization of AM) by analogous means. We can also obtain conditional simulations of NP by SPP, or of the polynomial time hierarchy by $\oplus P$. The advantage of these conditional simulations over earlier results is that the conditions are uniform rather than non-uniform.

Note that our technique also suggests that it might be difficult to show

that the Hopcroft-Paul-Valiant simulation of time by space cannot be significantly improved. Any such proof would imply a circuit lower bound for a function in E, and proving such lower bounds is widely believed to be hard. Even showing $P \not\subseteq i.o.i.DSPACE(\log(n)\log(\log(n)))$ would imply that there is a function in E that does not have linear-size circuits. No such functions are known at present.

4 BPP vs NEXP

First, we show that to separate BPP from $NEXP$, it suffices to simulate BPP in non-deterministic subexponential time for infinitely many input lengths. The proof is implicit in [IKW00].

Proposition 13 [IKW00] $BPP \subseteq i.o.NSUBEXP \Rightarrow BPP \neq NEXP$

Proof Assume, on the contrary, that $BPP = NEXP$. Then $EXP = NEXP$. Let L be a language that is complete for NE under linear-time reductions (it is easy to see that such a langauge exists). Then, $L \in DTIME(2^{n^c})$ for some constant c, since $NEXP = EXP$. Thus $NE \subseteq DTIME(2^{n^c})$ and so $i.o.NSUBEXP \subseteq i.o.NE \subseteq i.o.DTIME(2^{n^c})$. This implies $EXP = BPP \subseteq i.o.DTIME(2^{n^c})$, which is a contradiction to the a.e. hierarchy theorem for deterministic time classes [GHS91]. □

Next, we explore a connection between simulation of probabilistic time by nondeterministic time and tradeoffs between nondeterministic time and space.

Theorem 14 $NP \subseteq i.o.i.NPOLYLOGSPACE$ or $BPP \subseteq i.o.NSUBEXP$

Proof We shall give a proof analogous to Lemma 3 for nondeterministic classes. Given an NP machine M such that $L(M)$ is infinite, and a space bound $S = \log(n)^k$ for some k, perform the simulation of Lemma 3 on each computation path of M. If the simulation fails, reject. If the simulation succeeds on at least one accepting computation path of M on x, we say that the simulation succeeds on x. Either there is an infinite set $A \subseteq L(M)$ such that the simulation succeeds on all members of A, in which case $L(M) \in i.o.i.NPOLYLOGSPACE$, or the simulation fails on all but finitely many inputs in $L(M)$. The key point is that by determining the outcome of a computation path of M on an input, we will know whether the computation

on that computation path is a good candidate for producing the truth table of a hard function or not.

Either, for each NP machine, the simulation succeeds for some $S = \log(n)^k$ or there is a machine M such that the simulation fails for all poly-logarithmic space bounds. In the second case, we will be able to use the hypothetical NP machine M for which the simulation fails to produce the truth table of a function that is hard for infinitely many lengths. More precisely,we shall define, for each $\delta > 0$, a nondeterministic machine N_δ that operates in time 2^{m^δ} on input 1^m, outputs the truth table of a hard function on m^δ inputs on every accepting computation, and accepts infinitely many inputs of the form 1^m. Combined with the pseudo-random generator of [BFNW93], N_δ can be used to simulate BPP for infinitely many input lengths in $NTIME(2^{n^\delta})$, and thus $BPP \subseteq i.o.NSUBEXP$.

Let the NP machine M for which the simulation fails run in time n^k for some constant k. The machine N_δ operates as follows: on input 1^m, it first guesses a string x of length between $2^{(m-1)^\epsilon/k}$ and $2^{m^\epsilon/k}$, where ϵ is to be determined later. N_δ simulates M on x and on a separate tape writes down the concatenation t of configurations of M on x. If M accepts on x, N_δ is guaranteed that t is the truth table of a hard function, so it outputs t and halts in an accepting state. Otherwise, it halts in a rejecting state.

Clearly, N_δ outputs the truth table of a hard function on every accepting computation. Also, since the simulation fails for all but finitely many members of $L(M)$, there are infinitely many m such that N_δ accepts on 1^m. All that remains is to choose ϵ so that N_δ halts in time 2^{n^δ}. Clearly, choosing $\epsilon < \delta$ suffices. □

Using Theorem 4 rather than Theorem 2, we can show -

Theorem 15 $NP \subseteq i.o.i.SUBEXP$ or $BPP \subseteq i.o.NP$.

Corresponding to Theorem 10, we can prove -

Theorem 16 For polynomially bounded T, $NTIME(T) \subseteq i.o.i.NTISP(T^2 polylog(T), polylog(T))$, or $BPP \subseteq i.o.NSUBEXP$

Proof Analogous to proof of Theorem 6.

Theorem 16 is interesting because it demonstrates limitations to showing time-space tradeoffs for nondeterministic classes. For classes defined by mul-

titape TMs, [San01] proved the tradeoff $NTIME(n) \not\subseteq NTISP(n^{2-\epsilon}, \log(n)^k)$, where ϵ and k are any positive constants. Theorem 16 shows that even a slight extension of this result will imply the existence of pseudo-random generators(and hence a superpolynomial circuit size lower bound for a function in $NEXP$) and is therefore likely to be quite hard.

With Theorem 14, we are very close to showing that $BPP \neq NEXP$. What it says is that if NP is not somewhat easy i.o.i., i.e., if it cannot be simulated infinitely often in deterministic quasi-polynomial time,, then BPP can be simulated infinitely often in $NSUBEXP$, and thus $BPP \neq NEXP$ by Proposition 13. On the other hand, we know from Lautemann's theorem that if NP is somewhat easy, then BPP is somewhat easy and hence $BPP \neq NSUBEXP$. The only remaining case is if NP is somewhat easy only infinitely often. We conjecture that Lautemann's theorem can be extended to this case and thus $BPP \neq NEXP$.

Conjecture 1 If, for all constructible t, $NTIME(T) \subseteq i.o.i.DTIME(f(T))$, then $BPTIME(T) \subseteq i.o.i.DTIME(f(f(T)))$

This conjecture is likely to be quite hard to prove because only very sparse subsets of languages in $NTIME(T)$ might be decidable in $DTIME(f(T))$. But we note that a weaker version of the conjecture, where we only ask to be able to show that $BPTIME(T) \subseteq i.o.i.DTIME(f(f(T)))$ when $NTIME(T)$ is in $DTIME(f(T))$ on all "easy" inputs, where an input is "easy" if it represents the truth-table of a function that has polynomial time circuits, is sufficient to separate BPP and $NEXP$.

Our next conjecture concerns the position of BPP in the polynomial time hierarchy.

Conjecture 2 $BPP \subseteq NP^{NP}_{mtt}$.

We show below that this conjecture implies $BPP \neq NEXP$:

Theorem 17 $BPP \subseteq NP^{NP}_{mtt} \Rightarrow BPP \neq NEXP$.

Theorem 17 is a consequence of Lemmas 18, 19 and 20:

Lemma 18 $NP \subseteq quasi_{NP} - QP$, or $BPP \subseteq i.o.NSUBEXP$.

Proof Analogous to the proof of Theorem 9. We again use the simulation of Lemma 3. Either, for each k, the simulation S_k of an NP machine in

$NSPACE(\log(n)^k)$ succeeds almost everywhere against all polynomial time adversaries, or we can simulate BPP efficiently in nondeterministic time. If, for every NP machine M accepting an infinite language , there is a k such that $L(M) \in quasi_{NP} - NSPACE(\log(n)^k$, then the first clause of Lemma 3 holds, since $NPOLYLOGSPACE \subseteq QP$. Otherwise, for each simulation S_k there is a weak refuter N_k that weakly distinguishes $L(M)$ from the language accepted by the simulating machine. For each $\delta > 0$, we define a nondeterministic Turing machine M'_δ that on input 1^n, operates in time 2^{n^δ} and outputs the truth table of a Boolean function on $O(n^\delta)$ inputs with no circuits of size n for infinitely many n. From this, it follows that $BPP \subseteq i.o.NSUBEXP$.

M'_δ operates as follows: Given input 1^n, it runs $N_{1/2\delta}$ on the input to produce strings $x_1, x_2 \ldots x_{p(n)}$, where $p(n)$ is a fixed polynomial and each x_i is of length n. For each x_i in order, M' runs M on x_i and stores the concatenation C_{M,x_i} of configurations of M on x_i corresponding to the current computation path. If M rejects, M' rejects, otherwise it continues the simulation with x_{i+1}. If M accepts on every x_i, M' concatenates C_{M,x_i} for $i = 1 \ldots p(n)$ to produce a string C_n, which is a candidate for the truth table of a hard function.

It is sufficient to show that, for infinitely many n, every string C_n ouput by M'_δ is the truth table of a function that does not have circuits of size n. By assumption, for infinitely many n, there is a string x output by N_δ on 1^n such that the simulation fails on every accepting path of M on x. Thus $C_{M,x}$ does not have circuits of size n, and hence neither does C_n. □

Lemma 19 If $BPP \subseteq NP^{NP}_{mtt}$, then $NP \subseteq quasi_{NP} - QP \Rightarrow BPP \subseteq quasi_P - QP$

Proof Assume $BPP \subseteq NP^{NP}_{mtt}$ and $NP \subseteq quasi_{NP} - QP$. We show that, under the second assumption, $NP^{NP}_{mtt} \subseteq quasi_{NP} - NQP$. Assume, on the contrary, that there is a language L in $NP^{NP}_{mtt} \setminus quasi_{NP} - NQP$. Without loss of generality, $L = L(M^{SAT})$, where M is a non-deterministic polynomial time oracle machine and all queries made to SAT on an input of length n are of length $p(n)$, where p is a polynomial (this is true because of the paddability of SAT). We show that, if $L \notin quasi_{NP} - NQP$, then $SAT \notin quasi_{NP} - QP$, contradicting the assumption that $NP \subseteq quasi_{NP} - QP$. If $L \notin quasi_{NP} - NQP$, for every language $L' \in NQP$, there is a weak refuter $N_{L'}$ weakly distinguishing L' from L. We construct, for each language L'' in QP, a weak refuter $N'_{L''}$ weakly distinguishing SAT from L''. Let $L' = L(M^{L''})$.

On input 1^n, $N'_{L''}$ first checks if $q(n) = p^{-1}(n)$ exists. If not, it outputs an arbitrary set of strings. Otherwise, it runs $N_{L'}$ on input $1^{q(n)}$ and obtains a polynomial size set of strings $x_1, x_2 \ldots x_{r(n)}$. For each $i, 1 \leq i \leq r(n)$, it guesses the query set Q_i of M on x_i in polynomial time and checks that the set is correct(this can be done since M is a mild truth-table reduction) . It concatenates all sets $Q(i)$ into a set S and outputs S.

We need to show, that for each L'', $N_{L''}$ weakly distinguishes L'' from SAT. The key observation is that $L(M^{L''})$ is in NQP, since M is a non-deterministic polynomial-time machine and L'' is in QP. Thus, if answers to all queries in every set $Q(i)$ agreed with SAT, $N_{L'}$ would not output any strings in $L \triangle L'$ on input 1^n. This cannot happen for all but finitely many n since $N_{L'}$ is a weak refuter; thus $N_{L''}$ infinitely often outputs strings in $SAT \triangle L''$.

Thus we know that $BPP \subseteq quasi_{NP} - NQP$. By translation, since $NP \subseteq quasi_{NP} - QP$, $quasi_{NP} - NQP \subseteq quasi_{NP} - QP$, which yields our result. □

Lemma 20 $EXP \not\subseteq quasi_{NP} - QP$

Proof A straightforward diagonalization.

The concept of a mild truth-table reduction is admittedly somewhat artificial, but Conjecture 2 is interesting because there does not seem to be a proof that it implies $BPP \neq NEXP$ using only standard structural complexity-theoretic techniques such as diagonalization. On the other hand, it is known [Moc96] how to show using such techniques that the class of languages truth-table reducible to NP is a strict subset of $NEXP$(Under the assumption that BPP is in this class, we can actually obtain the stronger result that $BPP \subseteq i.o.NSUBEXP$). It is an interesting question to find weaker assumptions than Conjecture 2 on the position of BPP in the polynomial-time hierarchy under which $BPP \neq NEXP$. There is an oracle relative to which $BPP = P^{NP} = NEXP$ [vM], therefore showing $BPP \subseteq P^{NP}$ implies $BPP \neq NEXP$ would require non-relativizing techniques, while all our techniques relativize.

For the sake of comparison, we state another conjecture the truth of which would imply $BPP \neq NEXP$. This conjecture involves the Minimum Circuit Satisfiability Problem (MCSP). The input to this problem is a pair (t, s) where t is interpreted as the truth table of a function and s as a size bound. The input is accepted if there is a circuit for the function of size less

than or equal to s. MSCP is clearly in NP. Cai and Kabanets [KC00] studied the problem in detail and showed that it is highly unlikely that MCSP is in P or that $MCSP$ is NP-complete under "natural" reductions. The conjecture we make is that MCSP is in $co - NQP$.

Conjecture 3 $MCSP \in co - NQP$.

The idea that this hypothesis is sufficient for separating BPP and $NEXP$ is implicit in [IW97] and in [Rud97], where it is stated in a non-uniform framework.

Proposition 21(folklore) If $MCSP \in co - NQP$, then $BPP \neq NEXP$

Corollary 22 $NP \subseteq co - NQP$ or $BPP \neq NEXP$.

A very similar proof to one in [KC00] shows that there are oracles relative to which the result $MCSP \in co - NP$ does not hold. But it is unclear whether an analogous result holds for the given hypothesis.

Three reasons why we think our conjectures worthy of consideration are: (1) They concern uniform classes, and results for uniform classes are generally easier to show than results for non-uniform classes (2) They are positive inasmuch as they require efficient simulations of a problem or of one class by another as opposed to negative lower bound proofs (3) Their status with respect to the standard classification of approaches to derandomizing BPP [IKW00], discussed in detail below.

There are three distinct approaches to derandomizing probabilistic classes. One is to construct pseudo-random generators, which is known to be as hard as proving circuit lower bounds. Another is to construct efficient approximators of circuit acceptance probabilities, which is equivalent to showing Promise-BPP is easy. In [IKW00], it is shown that simulating Promise-BPP in $NSUBEXP$ would imply that there is a function in $NEXP$ that does not have polynomial size circuits. This, again, is thought to be hard. The third and most general approach is to show that there exists, for every probabilistic algorithm, a nontrivial deterministic (or non-deterministic) algorithm simulating it, without necessarily knowing how to construct such an algorithm.

Note that our approach to derandomizing BPP does not require pseudo-random generators to exist. We know that either time-bounded classes can be simulated efficiently by space-bounded classes or pseudo-random generators exist, but we do not know which of the two is the case. A proof of Con-

jecture 1 or Conjecture 2 would imply a successful attack on $BPP \neq NEXP$ satisfying the third criterion of [IKW00], and cannot be used in any obvious way to show circuit lower bounds. Hence these seem to us to be promising directions.

Finally, we note that by performing the simulation of Theorem 15 infinitely often against uniform non-deterministic adversaries, we obtain:

Theorem 23 $NP \subseteq [io - quasi_{NP}] - SUBPSPACE$ or $BPP \subseteq NP$

This is stronger than Theorem 24 in the paper by Kabanets [Kab00] which introduced the notion of uniform adversaries.

5 The fine structure of randomized classes

An interesting question is to find hardness conditions that imply results about the fine structure of randomized classes, i.e., about the relationships between randomized classes with specific resource bounds. In this section, we give conditions for the existence of a hierarchy for randomized polynomial time and for the nontrivial simulation of randomized time by deterministic space.

Unlike in the case of deterministic time and nondeterministic time, no strong hierarchy theorems are known for probabilistic time classes. Cai, Nerurkar and Sivakumar [CNS99] showed that if a version of the permanent is not in probabilistic sub-exponential time, then probabilistic quasi-polynomial time has a tight hierarchy. We consider a uniform assumption under which probabilistic polynomial time has a tight hierarchy. Our result is essentially an elaboration of Theorem 7.

Theorem 24 If there is a language $L \in DTIME(n) \setminus i.o.i.DSPACE(n^\epsilon)$ for some $\epsilon > 0$, then for each polynomially bounded t and $\delta > 0$, $BPTIME(t) \subset BPTIME(t^{1+\delta})$.

Proof If the hypothesis holds, we can construct, as in the proof of Theorem 7, the truth table of a hard function in time polynomial in the size of the truth table. It is implicit in [IW97] that this implies there is a constant c such that $BPTIME(t) \subseteq DTIME(t^c)$ for each polynomially bounded t. Now suppose, for the sake of contradiction, that there exist t and $\delta > 0$ such that $BPTIME(t) = BPTIME(t^{1+\delta})$. By translation, we obtain, $BPTIME(t) = BPTIME(t^{2c})$. Thus $DTIME(t^{2c}) \subseteq BPTIME(t^{2c}) =$

$BPTIME(t) \subseteq DTIME(t^c)$, which is a contradiction to the hierarchy theorem for deterministic time [HS66]. □

Note that the corresponding result for randomized space classes is known unconditionally, as a corollary to the simulation of randomized space by deterministic space using the recursive matrix powering technique.

Our second result gives a hardness hypothesis under which randomized time can be simulated nontrivially by randomized space (and, in fact, by deterministic space). In the seminal paper of Hopcroft, Paul and Valiant [HPV77], it is shown that $DTIME(t) \subseteq DSPACE(t/\log(t))$ for constructible time bounds t. But their techniques do not extend to showing a corresponding result for randomized classes. We are able to show the result under a hardness assumption. Showing the result unconditionally might be difficult because it would imply a nontrivial simulation of randomized time by deterministic time.

Theorem 25 If there is a language $L \in DSPACE(S)$ for some polynomially bounded S such that L does not have polynomial size circuits, then for each polynomially bounded T, $BPTIME(T) \subseteq DSPACE(T/\log(T))$.

Theorem 25 will follow from Lemmas 26, 27 and 28:

Lemma 26 For any T and $\epsilon > 0$, there is a $(\log(T), T^\epsilon)$ design of size T that can be generated in space $O(T^\epsilon)$.

Proof The proof is the same as in [KvM99], except that a different version of Chernoff's Lemma is used in the analysis.

Lemma 27 If there is a language $L \in DSPACE(S)$ for some polynomially bounded S such that L does not have polynomial size circuits, then for each T, there exists an $\epsilon > 0$ such that there is a pseudo-random generator $G : \{0,1\}^{T^\epsilon} \to \{0,1\}^{T^2}$ computable in space $O(T/\log(T))$.

Proof Essentially, we use the Nisan-Wigderson generator [NW94] - Lemma 26 guarantees the space efficiency of the generator. The Nisan-Wigderson generator needs a source that is hard on average rather than worst case hard, so we first convert the Boolean function f corresponding to L to a function g that cannot be approximated by polynomial-size circuits. The Sudan-Trevisan-Vadhan [STV01] reduction from average-case hardness to worst-case hardness is space-efficient, thus by applying this reduction, we obtain

from f a function g that can be computed in linear space. By Lemma 26, a Nisan-Wigderson design can be constructed in space $O(T/\log(T))$, hence only space $O(T^\epsilon + T/\log(T)) = O(T/\log(T))$ is required to compute any bit of the output of the Nisan-Wigderson generator corresponding to this design. □

Lemma 28 If there is a pseudo-random generator $G : \{0,1\}^{T^\epsilon} \to \{0,1\}^{T^2}$ computable in space $O(T/\log(T))$, then $BPTIME(T) \subseteq DSPACE(T/\log(T))$

Proof The idea is simple: we use the Hopcroft-Paul-Valiant simulation of time by space. Doing the simulation with pseudo-random strings rather than random strings allows us to recompute information as and when needed efficiently, without having to use more storage space than that required for the random seed.

More precisely, given an input x and a randomized Turing machine M operating in time T, we construct a deterministic Turing machine M' that simulates M. M' enumerates all strings y of length T^ϵ, and performs the Hopcroft-Paul-Valiant simulation for each y with $G(y)$ substituting for the random string. It then outputs the majority vote of the answers. During a simulation with a string $G(y)$, if any information needs to be recomputed, M' can do this efficiently, since it can obtain any bit of $G(y)$ from y using only space $O(T/\log(T))$. Clearly, after the computation for a particular y has been completed, space can be re-used for the next y, hence the entire computation takes only $O(T/\log(T))$ space. Since G is a pseudo-random generator with respect to circuits of size T^2, and since the computation of M on x can be simulated by a circuit of that size, the answer output by M' is correct. □

Corollary 29 If SAT does not have polynomial-size circuits, then for each polynomial t, $BPTIME(t) \subseteq DSPACE(t/\log(t))$.

Corollary 30 Unless the polynomial-time hierarchy collapses to the second level, for each polynomially bounded t, $BPTIME(t) \subseteq DSPACE(t/\log(t))$.

References

[ACR96] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. Hitting sets derandomize BPP. In Friedhelm Meyer auf der Heide and Burkhard Monien, editors, *Automata, Lan-*

guages and Programming, 23rd International Colloquium, volume 1099 of *Lecture Notes in Computer Science*, pages 357–368, Paderborn, Germany, 8–12 July 1996. Springer-Verlag.

[BDa90] José Luis Balcázar, Josep Díaz, and Joaquim Gab arró. *Structural Complexity 2*. Springer-Verlag, New York, NY, 1990.

[BDG88] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*. Springer-Verlag, New York, NY, 1988.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequence of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984. This paper introduces the notion of cryptographically secure pseudo-random number generator.

[CNS99] Jin-Yi Cai, Ajay Nerurkar, and D. Sivakumar. Hardness and hierarchy theorems for probabilistic quasi-polynomial time. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 726–735, New York, May 1999. Association for Computing Machinery.

[GHS91] John G. Geske, Dung T. Huynh, and Joel I. Seiferas. A note on almost-everywhere-complex sets and separating deterministic-time-complexity classes. *Information and Computation*, 92(1):97–104, May 1991.

[HPV77] J. Hopcroft, W. Paul, and L. Valiant. On time versus space. *Jrnl. A.C.M.*, 24(2):332–337, April 1977.

[HS66] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

[IKW00] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. In Frances M. Titsworth, editor, *Proceedings of the Sithteenth Annual Conference on Computational Complexity (CCC-01)*, pages 2–12, Los Alamitos, CA, June 18–21 2000. IEEE Computer Society.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC '97)*, pages 220–229, New York, May 1997. Association for Computing Machinery.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs. time: derandomization under a uniform assumption. In IEEE, editor, *39th Annual Symposium on Foundations of Computer Science: proceedings: November 8–11, 1998, Palo Alto, California*, pages 734–743, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1998. IEEE Computer Society Press.

[Kab00] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. In *Proceedings of the 15th Annual IEEE Conference on Computational Colmplexity (COCO-00)*, pages 150–157, Los Alamitos, CA, July 4–7 2000. IEEE Press.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In ACM, editor, *Proceedings of the thirty second annual ACM Symposium on Theory of Computing: Portland, Oregon, May 21–23, [2000]*, pages 73–79, New York, NY, USA, 2000. ACM Press.

[KvM99] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In ACM, editor, *Proceedings of the thirty-first annual ACM Symposium on Theory of Computing: Atlanta, Georgia, May 1–4, 1999*, pages 659–667, New York, NY, USA, 1999. ACM Press.

[Lu00] Chi-Jen Lu. Derandomizing arthur-merlin games under uniform assumptions. In *ISAAC 2000*, pages 302–312, 2000.

[Moc96] Sarah E. Mocas. Separating classes in the exponential-time hierarchy from classes in PH. *Theoretical Computer Science*, 158(1–2):221–231, May 1996.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[Rud97] Rudich. Super-bits, demi-bits, and NP /qpoly-natural proofs. In *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*. LNCS, 1997.

[San01] Santhanam. Lower bounds on the complexity of recognizing SAT by turing machines. *IPL: Information Processing Letters*, 79, 2001.

[Sip88] M. Sipser. Expanders, randomness, or time versus space. *Journal of Computer and System Sciences*, 36, 1988. Contains a discussion on efficiently reducing the probability of error in randomized algorithms. It also describes a relationship between pseudorandomness, time and space used by certain algorithms if certain types of expander graphs can be explicitly constructed.

[SSZ98] Michael Saks, Aravind Srinivasan, and Shiyu Zhou. Explicit OR-dispersers with polylogarithmic degree. *Journal of the ACM*, 45(1):123–154, January 1998.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer System Sciences*, 62(2):236–266, 2001.

[vM] Dieter van Melkebeek. Personal communication.

[Yao82] A. C. Yao. Theory and application of trapdoor functions. In IEEE, editor, *23rd annual Symposium on Foundations of Computer Science, November 3–5, 1982, Chicago, IL*, pages 80–91, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1982. IEEE Computer Society Press.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'