Electronic Colloquium on Computational Complexity, Revision 01 of Report No. 038 (2002)

ECCC

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) There is a simulation of P in $POLYLOGSPACE$ that is successful against all polynomial-time adversaries infinitely often, or $BPP \subseteq SUBEXP$ (2) There is a simulation of P in $SUBPSPACE$ that is successful against all polynomial-time adversaries infinitely often, or $BPP = P$. These results complement and extend 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.

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: $DTIME(t)$ has no $DSPACE(t^\epsilon)$-immune languages 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.

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].

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 need to be careful when specifying what it means for a machine of one class to be simulated by a machine in another class. According to the conventional notion, the simulation is successful if it fools all non-uniform adversaries, i.e., the languages accepted by the two machines coincide on all but finitely many inputs (if the simulation is "almost everywhere") or on infinitely many input lengths (if the simulation is "infinitely often"). Thus, 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.

Kabanets [Kab00] defines some notions of what it means for a simulation to be successful against all uniform adversaries (which is weaker than the conventional notion of a simulation). 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 also consider a weaker concept of refutation. A *weak refuter* is a deterministic Turing machine that, on input 1^n, outputs a 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 $quasi_A - quasi_A - C = quasi_A - C$.

We note that there is a still weaker concept of "infinitely often" simulations in the literature (eg. [Sip88]), where for every uniform adversary from a certain bounded complexity class, there is a successful simulation fooling it. Instead, we shall use the notion of immunity for some of our results. Given complexity classes C and D, C is said to have a D-immune language L if L is infinite and no infinite subset of L belongs to D.

All time and space bounds in this paper are assumed to satisfy "reasonable" constructibility conditions, which we shall leave unspecified.

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 polynomial time and $S = \Omega(\log(n))$ is a space-constructible function, there is a DTM M' operating in space

$S\log(S)$ such that for each input x, either (1) $x \in L(M)$ if and only if $x \in L(M^{'})$ or (2) $f_{M,x}$ has no circuits of size S.

Proof Assume that M has k tapes and a binary tape alphabet, and that M operates in time T on an input of length n. Assume wlog that k is a power of 2. We define a DTM $M^{'}$ that simulates M in small space. If the simulation fails on an input x, we can obtain a lower bound on the circuit complexity of $f_{M,x}$.

$M^{'}$ 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 $2\log(T)+O(1)$ (Hence the simulation only makes sense if $S > \log(T)$. 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. □

Note that the simulation can detect its own failure. The simulating machine either outputs the right answer or reports failure.

Theorem 4 ([BFNW93]) If for each k, there is a function $f_k \in E$ such that f_k does not have circuits of size n^k, then $BPP \subseteq i.o.SUBEXP$.

Theorem 5 $P \subseteq quasi_P - POLYLOGSPACE$, 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 there is a polylogarithmic space bound S such that every weak refuter N in P succeeds only finitely often against the simulation of M in space S, we have $P \subseteq quasi_P - POLYLOGSPACE$. Otherwise there is a polynomial-time machine M such that for each k, there is a weak refuter N_k in P that succeeds infinitely often against the simulation in space $\log(n)^{k+1}$. The truth table of a function f_k that does not have circuits of size n^k can be computed in time polynomial in the size of the truth table as follows: Concatenate $C_{M,x}$ for $x \in N_k(1^r), r = 2^{(n-1)/s} \ldots 2^{n/s}$, with the constant s, which depends on the running times of M and N_k, being determined by the condition that the length of the truth table obtained by this process is 2^n. Since N_k succeeds infinitely often, there are infinitely many input lengths on which the function f_k does not have circuits of size n^k. □

As Eric Allender has pointed out, it follows from [NW94] and standard downward translation techniques that either for each adversary in $LOGSPACE$, there is a simulation of P in $POLYLOGSPACE$ that fools the adversary on almost every input length, or $BPP \subseteq i.o.SUBEXP$. The statement we have proved is stronger because the order of the quantifiers is different. Also note that corresponding to a different version of Theorem 4, we can get either a simulation of polynomial-time machines in small space that succeeds infinitely often or a derandomization of BPP that succeeds almost everywhere.

Using a translation argument, we can show the following:

Theorem 6 For each t, $DTIME(t) \subseteq quasi_{DTIME(poly(t))} - DSPACE(polylog(t))$, or $BPP \subseteq i.o.SUBEXP$.

Proof It is sufficient to prove that if $DTIME(n) \subseteq quasi_{DTIME(poly(n))} - DSPACE(polylog(n))$, then $DTIME(t) \subseteq quasi_{DTIME(poly(t))} - DSPACE(polylog(t))$. We shall make the reasonable assumption that $t^{-1}(n)$ can be computed in time $poly(n)$ on input 1^n (this is true if t is a convex function and is time-constructible). Now assume, contrary to the hypothesis, that there is a language L in $DTIME(t)$ such that for every language L', there is a weak

refuter $M(L^{'})$ operating in $poly(n)$ time that weakly distinguishes L from $L^{'}$ infinitely often. Let $L_{pad} = \{x1^{t(n)-n} \mid x \in L, x \in \{0,1\}^n\}$. Clearly L_{pad} is in $DTIME(n)$. Now, for any language $R_{pad} \in DSPACE(polylog(n))$, consider $R = \{x \mid x \in \{0,1\}^n, x1^{t(n)-n} \in R_{pad}\}$.$R$ is in $DSPACE(polylog(t))$ and thus $M(R)$ weakly distinguishes L from R infinitely often. We define a weak refuter $M^{'}$ weakly distinguishing L_{pad} from R_{pad}. On input 1^n, $M^{'}$ computes $t^{'}(n) = t^{-1}(n)$, runs $M(R)$ on $1^{t^{'}(n)}$ to produce $poly(n)$ strings $x_1 \ldots x_{poly(n)}$ each of length $t^{'}(n)$, pads each of these strings to length n by adding 1's, and ouptuts the resulting strings. For infinitely many n, one of these strings is in $L_{pad} \triangle R_{pad}$, yielding a contradiction. □
By setting $t(n) = 2^n$, we obtain the theorem of Nisan and Wigderson stating that $EXP = PSPACE$ or $BPP \subseteq i.o.SUBEXP$.

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 7 ([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 $BPP \subseteq i.o.P$.

Theorem 8 For each $\epsilon > 0$ and polynomially bounded t, $DTIME(t) \subseteq quasi_{SUBEXP} - DSPACE(t^{\epsilon})$, or $BPP \subseteq i.o.P$.

Proof Analogous to the proof of Theorem 5, using Lemma 3 with $S = n^{\epsilon}$. □

Ideally, we would like the simulations in Theorem to work almost everywhere, or at least for infinitely many input lengths, against all adversaries, but such a result for nondeterministic machines would imply the separation of BPP from $NEXP$ and so is probably quite hard.

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 9 $P \subseteq quasi_P - NSC$ or $BPP \subseteq i.o.SUBEXP$.

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 an input to the circuit is only of size $O(\log(n))$, 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 10 ([KvM99]) If for each k there is a function $f_k \in NE \cap co-NE$ such that f_k has no SAT-oracle circuits of size n^k, then $AM \subseteq i.o.NSUBEXP$.

Theorem 11 $P \subseteq quasi_P{-}POLYLOGSPACE$ or $AM \subseteq i.o.NSUBEXP$.

Proof We modify the simulation of Lemma 3 and then combine the modified lemma with Theorem 10 to obtain the theorem.

We encode the configurations of a polynomial-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 against some uniform adversary for infinitely many input lengths, we obtain a function in E that does not have polynomial size SAT-oracle circuits and we can apply Theorem 10. □

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 $\oplus P$.

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 that P is not contained in $quasi_P{-}DSPACE(\log(n)\log(\log(n)))$ sets would imply that there is a function in E that does not have linear-size circuits. No such functions are known at present.

4 Probabilistic time vs nondeterministic time

We explore a connection between simulation of probabilistic time by nondeterministic time and tradeoffs between nondeterministic time and space.

The following proposition is analogous to Theorem 4:

Proposition 12 If for each k there is an NP machine M_k that, on infinitely many inputs of the form 1^{2^n}, accepts, and outputs the truth table of a function on n inputs that does not have circuits of size n^k on each accepting path, then $MA \subseteq i.o.NSUBEXP$.

Theorem 13 $NP \subseteq pseudo_{NP}{-}NPOLYLOGSPACE$ or $MA \subseteq i.o.NSUBEXP$

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 succeeds, accept or reject depending on whether the computation accepts or rejects, otherwise reject. Thus the simulating machine $M_k^{'}$ runs in space $\log(n)^k$.

Either the simulation succeeds against all refuters running in NP, or there is some refuter distinguishing $L(M)$ from $L(M_k^{'})$ infinitely often. If for every NP machine, the first clause holds for some k, then $NP \subseteq pseudo_{NP}{-}NPOLYLOGSPACE$. Otherwise, there is an NP machine M such that for each k, there is a NP refuter R_k distinguishing $L(M)$ from $L(M_k^{'})$ infinitely often.

To apply Proposition 12, we define an NP machine M_k as follows: On input 1^{2^n}, it guesses an input length j between $2^n/s$ and $2^{n-1}/s$, where s

is a constant depending on M. It then runs R_{k+1} on 1^j. If R_{k+1} accepts and produces a string y of length j, M_k runs M on y. If M accepts on y, M_k outputs $C_{M,y}$, where $C_{M,y}$ is the concatenation of configurations on the accepting path. Otherwise, it rejects.

We choose s so that the size of the truth tables output by M_k is 2^n (padding the truth table, if necessary). We still need to show that M_k works correctly. Note that the language accepted by any machine that simulates M in the prescribed manner is a subset of $L(M)$. Hence the refuter, when it is correct, outputs a string that is in $L(M)$ but is not accepted by the simulating machine. Hence all accepting computation paths of M on this string give rise to hard truth tables, which means that every string output by M_k on an accepting computation path is hard.

□

By a simple modification to the proofs of Theorem 5 and Theorem 13, we can obtain the stronger statement that if there is no simulation in space $\log(n)^k$, for some fixed k, then MA can be derandomized.

We are not able to show the stronger simulations against weak refuters for nondeterministic time. But the translation result does hold -

Theorem 14 For each $t, NTIME(t) \subseteq pseudo_{NTIME(poly(t))}-NSPACE(polylog(t))$, or $MA \subseteq i.o.NSUBEXP$.

Using Theorem 7 rather than Theorem 4, we can show -

Theorem 15 $NP \subseteq pseudo_{NSUBEXP} - SUBEXP$ or $MA \subseteq i.o.NP$.

Derandomization of AM can be carried out under the same assumption using similar techniques as in the previous section. Also, all results for NTIME hold with NTIME replaced by RTIME, as can easily be checked.

In the case of NP, it is also interesting to consider simulations against non-uniform adversaries. The proof of the following theorem is analogous to the proof of Theorem 13:

Theorem 16 If NP has a QP-immune set, then $MA \subseteq i.o.NSUBEXP$.

We know, from Lautemann's theorem, that if $NP \subseteq QP$, then $MA \subseteq NP^{NP} \subseteq QP$, and hence $MA \subseteq i.o.NSUBEXP$. On the other hand, if NP has QP-immune sets, by Theorem 16, $MA \subseteq i.o.NSUBEXP$. Thus,

what is of interest is the "gap" between the two statements about the hardness/easiness of NP. We now further refine this gap.

We consider languages in NP that are not too sparse, i.e., languages L such that for any c, $\mid L \cap \{0,1\}^n \mid > 2^{\log(n)^c}$ for all but finitely many n. Let us call the class of such languages NP_d. Also, let $s(n)$ be a function with appropriate constructibility properties that bounds the polylogarithmic functions from above, i.e., for each k, $\log(n)^k = o(s(n))$. We now prove a theorem that can be interpreted as a derandomization based on "hardness" of NP_d -

Theorem 17 If there is a language L in NP_d such that for each $L^{'} \in NSPACE(s(n)), L^{'} \subseteq L$, there is a constant c such that for infinitely many input lengths n, $L^{'}$ contains fewer than $2^{\log(n)^c}$ strings of length n, then $MA \subseteq i.o.NSUBEXP$.

Proof Assume the hypothesis holds for a language L in NP_d accepted by a nondeterministic machine M running in polynomial time. Consider the language $L^{'}$ defined by the simulation of Theorem 13 with space bound $s(n)$. We define a machine N that on infinitely many inputs of the form 1^{2^n} accepts and outputs on each accepting path a truth table of size 2^n that does not have polynomial size circuits. Let $\epsilon > 0$ be a small constant. On input 1^{2^n}, N guesses, for each input length k between $2^{(n-1)^{1/(c+\epsilon)}}$ and $2^{n^{1/(c+\epsilon)}}$, $2^{\log(k)^c}$ strings of length k, simulates M on each of these strings, and accepts only if M accepts on all these strings. On accepting, it outputs the concatenation of the configurations of M on all the strings, padded appropriately to length 2^n.

To see that N works correctly, note that by the assumption on density of L, N always accepts. Furthermore, by the hypothesis, for infinitely many k, there are at most $2^{\log(k)^c}$ strings on which the simulation succeeds and hence the sequence of configurations corresponding to any accepting path of any other input in L of length k is hard. A polynomial fraction of this hardness is preserved after concatenating all the truth tables, proving the theorem. □

Next we prove a theorem that can be interpreted as a derandomization based on the "easiness" of NP_d.

Theorem 18 If for each language L in NP_d, there is a language $L^{'}$ such that $L \subseteq L^{'}$,$L^{'} \in DTIME(2^{\log(n)^c})$ for some constant c, and for each n, if L does not contain all strings of length n, then there is at least one string x

of length n such that $x \notin L$ and $x \notin L'$, then $MA \subseteq NSUBEXP$.

Proof Consider $L = \{x \mid x$ is the truth table of a function accepted by a circuit of size $< s(\log(n))\}$. Clearly, $L \in NP_d$. Thus there is $L' \in DTIME(2^{\log(n)^c})$ satisfying the conditions of the hypothesis. Let M be a deterministic Turing machine running in time $2^{\log(n)^c}$ and accepting the complement of L. Choose a small constant $\epsilon > 0$. We construct a nondeterministic Turing machine N that outputs hard strings as follows: N, on input 1^{2^n}, guesses a string x of length $2^{n^{1/(c+\epsilon)}}$, and runs M on it, accepting and outputting x padded to length 2^n only if M accepts. It is easy to see that N accepts on every input of length 2^n and only outputs strings that are hard. □

Though the theorems above show that it is very likely that $MA \subseteq i.o.NSUBEXP$, this result is probably hard to prove, since by the results of [IKW00], this would imply $NEXP \not\subseteq P/poly$. In an earlier version of this paper, we suggested an approach to showing $BPP \subseteq i.o.NSUBEXP$ by situating BPP lower and lower in the polynomial-time hierarchy. As recent results of Impagliazzo and Kabanets indicate that proving $BPP \subseteq i.o.NSUBEXP$ would imply circuit lower bounds in either the Boolean or algebraic model, we now believe this approach will not be fruitful, and hence do not pursue it.

Corresponding to Theorem 9, we can prove -

Theorem 19 For polynomially bounded T, $NTIME(T)$ has no $NTISP(T^2 polylog(T), polylog(T))$-immune sets , or $MA \subseteq i.o.NSUBEXP$

Proof Analogous to proof of Theorem 6.

Theorem 19 is interesting because it demonstrates limitations to showing time-space tradeoffs for nondeterministic classes. For classes defined by multitape 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.

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 8.

Theorem 20 If there is a language $L \in DTIME(n) \setminus quasi_{SUBEXP} - DSPACE(n^{\epsilon})$ 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 8, 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 $BPTIME(t) \subseteq i.o.DTIME(t^c)$ 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 $DTIME(t^{2c}) \subseteq BPTIME(t^{2c}) = BPTIME(t) \subseteq i.o.DTIME(t^c)$, which is a contradiction to the a.e.hierarchy theorem for deterministic time [GHS91]. □

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

Theorem 21 will follow from Lemmas 22, 23 and 24:

Lemma 22 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 23 If there is a language $L \in DSPACE(S)$ for some polynomially bounded S such that L does not have polynomial size circuits infinitely often, 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 22 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 22, 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 24 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 25 If QBF does not have polynomial-size circuits infinitely often, then for each polynomial t, $BPTIME(t) \subseteq DSPACE(t/\log(t))$.

Corollary 26 Unless PSPACE collapses to the second level of the polynomial-time hierarchy, for each polynomially bounded t, $BPTIME(t) \subseteq i.o.DSPACE(t/\log(t))$.

6 Acknowledgments

Thanks to Dieter van Melkebeek and Eric Allender for pointing out flaws in an earlier version of this paper.

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, Languages 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.

[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.

[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.

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

[San01] Rahul 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.

[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'