Electronic Colloquium on Computational Complexity, Report No. 73 (2008)

ECCC

Structural complexity of **AvgBPP**

Dmitry Itsykson *

August 5, 2008

Abstract

We study class **AvgBPP** that consists of distributional problems that can be solved in average polynomial time (in terms of Levin's average-case complexity) by randomized algorithms with bounded error. We prove that there exists a distributional problem that is complete for **AvgBPP** under polynomial-time samplable distributions. Since we use deterministic reductions, the existence of a deterministic algorithm with average polynomial running time for our problem would imply **AvgP** = **AvgBPP**. Note that, while it is easy to construct a *promise problem* that is complete for **promise-BPP** [Mil01], it is unknown whether **BPP** contains complete *languages*. We also prove a time hierarchy theorem for **AvgBPP** (time hierarchy theorem is also unknown for **BPP**). We compare average-case classes with their classical (worst-case) counterparts and show that the inclusions are proper.

*Steklov Institute of Mathematics at St. Petersburg. E-mail: dmitrits@pdmi.ras.ru. Partially supported by Russian Science Support Foundation, grant RFBR 08-01-00640, and the president of Russia grant "Leading Scientific Schools" NSh-4392.2008.1.

ISSN 1433-8092

Contents

1 Introduction

The existence of complete problems and time (or space) hierarchies are the main structural properties of complexity classes. The complexity classes are usually defined by computational models. For example, the class **P** is defined by polynomial-time deterministic Turing machines, **NP**, by nondeterministic ones, and **BPP**, by randomized two-sided bounded error machines. A time hierarchy theorem states that a given computational model can decide more languages if it is allowed to use more time. A complete language is the hardest language in the class; all other languages can be reduced to it.

Time hierarchies and complete problems are connected, since both of them usually require efficient enumeration of (correct) machines in the respective computational model. It is unknown whether **BPP** has time hierarchy or complete problems under deterministic reductions. The main obstacle is the absence of efficient enumeration of two-sided bounded error randomized Turing machines. Barak showed in [Bar02] that the existence of a **BPP**-complete problem implies a time hierarchy theorem for **BPP**. However, there is a relativized world where $\mathbf{BPP}^{\mathbf{A}}$ has no complete languages [HH86]. Note that if $\mathbf{P} = \mathbf{BPP}$, then **BPP** does have a complete problem since **P** does. The best current result for time hierarchy is superpolynomial: $\mathbf{BPTime}[n^{\log n}] \subsetneq \mathbf{BPTime}[2^{n^\epsilon}]$ [KV87], however, we are not able to prove that $\mathbf{BPTime}[n] \subsetneq \mathbf{BPTime}[n^{100 \log n}]$. Related results for randomized classes include hierarchies for classes with one bit of nonuniform advice: $\mathbf{BPP}/1, \mathbf{ZPP}/1, \mathbf{MA}/1$, etc. [FS07].

In average-case complexity computational problems are supplied with probability distributions on the instances. We say that distributional problem is solvable in average polynomial time if there is *polynomial-time errorless heuristic scheme* that solves this problem [Imp95, BT06]. Polynomial-time errorless heuristic scheme is an algorithm that depends on two parameters: x (input) and δ ("give up" probability), its running time is bounded by a polynomial in $\frac{|x|}{\delta}$; it always outputs the correct answer or "gives up"; the probability of "give up" (according to some distribution on the inputs) is bounded by δ. This definition is given by Imagliazzo in his influential survey paper on average-case complexity [Imp95] and it is equivalent to Levin's definition of average-case tractability [Lev86]. **AvgP** is the class of distributional problems that are solvable in average polynomial time, and **AvgBPP** is the class of distributional problems that are solvable in randomized average polynomial time with bounded error.

Heuristic classes are closely connected with average-case classes. **HeurP** (**HeurBPP**) is defined by *(randomized) polynomial-time heuristic schemes* that in contrast to errorless schemes may give incorrect answer instead of explicit "give up". Heuristic algorithms, in contrast to heuristic schemes may error on some fraction of inputs and the number of errors may not be decreased by increasing running time. $\mathbf{Heur}_{\delta(n)}\mathbf{P}$ ($\mathbf{Heur}_{\delta(n)}\mathbf{BPP}$) is the class of problems that are solvable by (randomized with bounded error) polynomial-time heuristic algorithms on all inputs except fraction $\delta(n)$. A time hierarchy for the class $\mathbf{Heur}_{\frac{1}{n^c}}\mathbf{BPP}$ (with uniform distributions) was proved in [FS04, Per07]. A complete public-key cryptosystem was constructed using a similar idea in [HKN+05] (see also [GHP06]). Such cryptosystem is possible when a decoding algorithm may err with some small probability.

In this paper we consider only polynomial-time samplable distributions and denote the class of all polynomial-time samplable distributions as **PSamp**.

1.1 Results

1. We construct language C and polynomial-time samplable distribution R such that the distributional problem (C, R) is complete for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ under deterministic Turing reductions. Our construction implies that if this problem belongs to $(\mathbf{AvgP}, \mathbf{PSamp})$ (or even to $(\mathbf{Avg}_{\frac{1}{n^c}}, \mathbf{PSamp})$), then $(\mathbf{AvgBPP}, \mathbf{PSamp})$ equals $(\mathbf{AvgP}, \mathbf{PSamp})$. The same result is also correct for $(\mathbf{HeurBPP}, \mathbf{PSamp})$.

   The constructed distribution R is not uniform and is somewhat unnatural samplable. We give two intuitional ideas why it is very hard problem to construct a complete problem with uniform (or uniform-like) distribution:

   - informally, the reduction usually requires enlarging the input size which decreases the probability exponentially, which violates the domination condition in case of uniform distribution.
   - formally, we prove that if there exists a complete problem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ with uniform (or uniform-like) distribution, then there exists partial derandomization of $\mathbf{BPEXP}$, namely for all languages $L \in \mathbf{BPEXP}$ distributional problem (L, U) is solvable by a deterministic algorithm with average exponential running time, where U denotes uniform distribution.

2. We prove a time hierarchy theorem for the class $(\mathbf{AvgBPP}, \mathbf{PSamp})$. Namely, we prove that for every $c \geq 1$ there exists a language L and a polynomial-time samplable distribution D such that $(L, D) \in \mathbf{AvgBPP}$ and $(L, D) \notin \mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c]$ (note that $\mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c] \supseteq \mathbf{Avg}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c] \supseteq \mathbf{AvgBPTime}[n^c]$). Previous results [FS04, Per07] give a hierarchy for randomized heuristic algorithms with bounded error while we extend it to a hierarchy for randomized *errorless heuristic schemes* with bounded error. Weakness of our result is that the distribution D is not uniform. It is an interesting open question to prove the same for uniform D.

3. We compare classes $\mathbf{AvgP}, \mathbf{AvgBPP}, \mathbf{HeurP}, \mathbf{HeurBPP}$ to their worst-case counterparts and show the following inclusions (for polynomial-time samplable distributions):

   - $\mathbf{P} \subsetneq \mathbf{AvgP} \subseteq \mathbf{HeurP} \subsetneq \mathbf{EXP}$;
   - $\mathbf{BPP} \subsetneq \mathbf{AvgBPP} \subseteq \mathbf{HeurBPP} \subsetneq \mathbf{BPEXP}$.

Organization of the paper. In Sect. 2 we define rigorously the notions we use. In Sect. 3 we prove proper inclusions between average-case and worst-case classes, in Sect. 4 we give a construction of a problem complete for $(\mathbf{AvgBPP}, \mathbf{PSamp})$, in Sect. 5 we show that the existence of complete problem with uniform distribution implies derandomization, and in Sect. 6 we prove a time hierarchy theorem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$.

2 Preliminaries

2.1 Notation

We restrict ourselves to the binary alphabet $\{0, 1\}$, we denote the set of all binary words as $\{0, 1\}^*$. A language is any subset of $\{0, 1\}^*$. We identify language and its characteristic function: $x \in L \iff L(x) = 1$.

Recall that the complexity class **P** contains all languages L such that there exists polynomial-time deterministic Turing machine M such that $\forall x L(x) = M(x)$. Complexity class **BPP** contains all languages L such that there exists polynomial time randomized Turing machine M such that $\forall x \Pr\{M(x) = L(x)\} \geq \frac{3}{4}$. If we substitute the polynomial-time restriction by exponential-time restriction (running time is exponential if it is bunded by $2^{p(n)}$, where p is polynomial and n is length of the input), we get the definition of classes **EXP** and **BPEXP**.

Let $f, g : X \to \mathbb{R}_+$ be two functions. We say that $f(x) = poly(g(x))$ if there exists a polynomial $p(x)$ such that for all $x \in X$ $f(x) \leq p(g(x))$.

2.2 Distributions

In the average-case complexity theory computational problems are supplied with distributions on their instances. There are two ways of understanding the notion of distribution on binary strings. The first approach [Lev86, BDCGL92] is to define the probability distribution as a probabilistic measure on the set of all binary strings. The second approach [Imp95, BT06] is to define the distribution as a family of probabilistic measures with a finite support. These two approaches are equivalent in the sense of average-case complexity (see [Imp95, BT06]). We choose the second approach.

The *ensemble of distributions* D is the family of functions $\{D_n\}_{n=1}^{\infty}$, where D_n is a mapping $\{0,1\}^* \to [0,1]$ and there exists a finite set $X_n \subset \{0,1\}^*$ such that $\sum_{x \in X_n} D_n(x) = 1$, $D_n(x) > 0$ for all $x \in X_n$, and $D_n(x) = 0$ otherwise. The set X_n is called the support of D_n and denoted as $\operatorname{supp} D_n$; $\operatorname{supp} D = \cup_{n \in \mathbb{N}} \operatorname{supp} D_n$. *A distributional problem* is pair (L, D) of language L and ensemble of distributions D.

Let $\mathfrak{P}$ be the class of *distributional problems* and $\mathfrak{D}$ be the class of distributions. $(\mathfrak{P}, \mathfrak{D}) = \{(L, D) | (L, D) \in \mathfrak{P}, D \in \mathfrak{D}\}$.

In this paper we consider only polynomial-time samplable distributions. The ensemble of distributions D is called *polynomial-time samplable* if there exists a polynomial-time randomized algorithm (sampler) $\mathcal{S}$ such that outputs of $\mathcal{S}(1^n)$ are distributed according to D_n. The set of all polynomial-time samplable distributions is denoted by **PSamp**.

It is easy to see that lengths of strings from the support of D_n are bounded by $poly(n)$ if D is polynomial-time samplable. We think of D_n as a distribution on strings of size approximately n. In Section 2.5 we show that w.l.o.g. we may consider D_n with $\operatorname{supp} D_n \subseteq \{0,1\}^n$.

By uniform distribution U we mean the ensemble of distributions U_n, where U_n is the uniform distribution on $\{0,1\}^n$. In what follows we always mean an ensemble of distributions whenever we use the word distribution.

2.3 Average polynomial time and errorless heuristic schemes

The first notion of average-case tractability was given by Levin in [Lev86]. Function $t : \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ (with distribution D) is called *polynomial on the average*[1] if there exists $\epsilon > 0$ such that $\mathrm{E}_{x \leftarrow D_n} t(x; n) = O(n)$. The distributional problem is solvable in average polynomial time if there exists an algorithm that solves it with average polynomial running time.

The equivalent definition of the average-case tractability was given by Impagliazzo [Imp95]. Distributional problem (L, D) is solvable in *polynomial on the average* time if there exists algorithm

[1] The naive approach is to define that $t(x; n)$ to be polynomial on the average if the expectation of $t(x; n)$ is polynomial. But this naive definition is not closed under some natural operations. For example, it is easy to construct $t(x; n)$ such that the expectation of $t(x; n)$ is polynomial but the expectation of $t^2(x; n)$ is exponential (see [BT06]).

$\mathcal{A}(x; n, \delta)$ (following [BT06] we call such algorithm *errorless heuristic scheme*) that may explicitly "give up" (output $\perp$) such that the following conditions are satisfied:

- (Effectiveness) The running of $\mathcal{A}(x; n, \delta)$ is bounded by $poly(\frac{n}{\delta})$;
- (Correctness) for all x in the support of D, $\mathcal{A}(x; n, \delta) \in \{L(x), \perp\}$;
- (Usefulness) $\Pr_{x \leftarrow D_n}\{\mathcal{A}(x; n, \delta) = \perp\} < \delta$.

A formal proof of equivalence is given in [Imp95, BT06]. The answer $\perp$ in Impagliazzo's definition corresponds to manual interruption of the algorithm from Levin's definition.

The set of all distributional problems that can be solved in average polynomial time is denoted by **AvgP**.

Both these definitions may be extended for bounded error randomized algorithms. We say that randomized algorithm $\mathcal{A}$ solves (L, D) with bounded error if for all x in the support of D_n $\Pr\{\mathcal{A}(x; n) \neq L(x)\} < \frac{1}{4}$. When using Levin's definition we define the running time of the algorithm $\mathcal{A}$ on the input x as $\min\{t | \Pr\{\mathcal{A}(x) \text{ stops in } t \text{ steps}\} \geq \frac{3}{4}\}$.

The Impagliazzo-style definition is as follows:

Definition 2.1 ([BT06, Definition 2.13]). The distributional problem (L, D) is solvable in randomized average polynomial time with bounded error if there exists an algorithm (we call such algorithm as *randomized errorless heuristic scheme*) $A(x; n, \delta)$ such that the following conditions are satisfied:

- (Effectiveness) The running of $\mathcal{A}(x; n, \delta)$ is bounded by $poly(\frac{n}{\delta})$;
- (Correctness) for all x in the support of D, $\Pr\{\mathcal{A}(x; n, \delta) \notin \{L(x), \perp\}\} < \frac{1}{4}$, where the probability is taken over random bits of algorithm $\mathcal{A}$.
- (Usefulness) $\Pr_{x \leftarrow D_n}\{\Pr\{\mathcal{A}(x; n, \delta) = \perp\} \geq \frac{1}{4}\} < \delta$, where the inner probability is taken over the random bits of algorithm $\mathcal{A}$.

The class **AvgBPP** consists of all problems that are solvable in randomized average polynomial time with bounded error.

For every function $g(n)$, we define classes **AvgTime**$[g(n)]$ and **AvgBPTime**$[g(n)]$; the definitions are the same as definitions of **AvgP** and **AvgBPP** but effectiveness conditions are substituted by the following: the running time of $\mathcal{A}(x; n, \delta)$ is bounded by $O(g(\frac{n}{\delta}))$. We also define class $\mathbf{AvgEXP} = \bigcup_{c>0} \mathbf{AvgTime}[2^{n^c}]$.

2.4 Heuristic classes

Similarly to errorless heuristic schemes it is possible to define general heuristic schemes. In the deterministic case we say that distributional problem (L, D) is solvable by *polynomial time heuristic scheme* $A(x; n, \delta)$ if

- (Effectiveness) The running of $\mathcal{A}(x; n, \delta)$ is bounded by $poly(\frac{n}{\delta})$;
- (Usefulness) $\Pr_{x \leftarrow D_n}\{\mathcal{A}(x; n, \delta) \neq L(x)\} < \delta$.

In the randomized case the usefulness condition is formulated as follows:

- (Usefulness) $\Pr_{x \leftarrow D_n}\{\Pr\{\mathcal{A}(x;n,\delta) \neq L(x)\} \geq \frac{1}{4}\} < \delta$, where the inner probability is taken over the random bits of algorithm $\mathcal{A}$.

The set of all distributional problems solvable by polynomial-time (randomized) heuristic schemes is denoted by **HeurP** and **HeurBPP**.

We say that problem (L, D) is solved by *$\delta(n)$-heuristic algorithm* $\mathcal{A}$ if for every n, it holds that $\Pr_{x \leftarrow D_n}\{\mathcal{A}(x;n) \neq L(x)\} < \delta(n)$. We say that (L, D) is solved by *errorless $\delta(n)$-heuristic algorithm* if $\mathcal{A}$ is $\delta(n)$-heuristic and $\mathcal{A}(x;n) \in \{L(x), \perp\}$ for all x and n.

The set of problems solvable by polynomial-time (errorless) δ-heuristic algorithms is denoted by $\mathbf{Heur}_{\delta(n)}\mathbf{P}$ (resp. $\mathbf{Avg}_{\delta(n)}\mathbf{P}$). The classes $\mathbf{Heur}_{\delta(n)}\mathbf{BPP}$ and $\mathbf{Avg}_{\delta(n)}\mathbf{BPP}$ are defined similarly. The classes $\mathbf{Heur}_{\delta(n)}\mathbf{Time}[g(n)]$, $\mathbf{Avg}_{\delta(n)}\mathbf{Time}[g(n)]$, $\mathbf{Heur}_{\delta(n)}\mathbf{BPTime}[g(n)]$, $\mathbf{Avg}_{\delta(n)}\mathbf{BPTime}[g(n)]$, $\mathbf{HeurTime}[g(n)]$, $\mathbf{HeurBPTime}[g(n)]$ are defined in a natural way.

2.5 Distributions on inputs of equal lengths

In this section we show that every distributional problem (L, D) with polynomial-time samplable distribution D is equivalent (in sense of average-case hardness) to some distributional problem (L', D') with polynomial-time samplable D' such that $\operatorname{supp} D'_n \subseteq \{0,1\}^n$ for all $n \in \mathbb{N}$.

Let $\mathcal{S}$ be an n^k-time sampler that generates the distribution D. We define sampler $\mathcal{S}'$ that generates distribution D' as follows:

1. Input: 1^m.
2. If $\exists n : m = n^k + 1$, then
   - $s \leftarrow \mathcal{S}(1^n)$;
   - Return $1^{m-|s|-1}0s$.
3. Otherwise, return $s \leftarrow U(\{0,1\}^m)$.

We define language $L' = \bigcup_{n \in \mathbb{N}}\{1^{n^k-|x|}0x | x \in L \cap \operatorname{supp} D_n\}$.

Proposition 2.1. $(L, D) \in \mathbf{AvgP}$ (resp. $\mathbf{AvgBPP}, \mathbf{HeurP}, \mathbf{HeurBPP}$) if and only if $(L', D') \in \mathbf{AvgP}$ (resp. $\mathbf{AvgBPP}, \mathbf{HeurP}, \mathbf{HeurBPP}$)

Proof. Suppose that algorithm $\mathcal{A}(x;n,\delta)$ solves (L, D) in $\mathbf{AvgP}$. We define algorithm $\mathcal{A}'(x;\delta)$ that solves (L', D') in $\mathbf{AvgP}$ as follows:

1. If $(\exists n : m = n^k + 1$ and $x = 1^{m-|s|-1}0s)$, then return $\mathcal{A}(s;n,\delta)$;
2. Otherwise, return 0.

Suppose that $\mathcal{A}'(x;\delta)$ solves (L', D') in $\mathbf{AvgP}$. Then the following algorithm $\mathcal{A}(x;n,\delta)$ solves (L, D) in $\mathbf{AvgP}$ as follows:

1. If $|x| > n^k$, then return 0.
2. Otherwise, return $\mathcal{A}'(1^{n^k-|x|}01, \delta)$.

One can easily check that the described transformations satisfy all necessary properties. □

In what follows for technical reasons we restrict ourselves only to polynomial-time samplable distributions with supports of the same lengths. We do not write the parameter n explicitly and use $D(x)$ instead of $D_{|x|}(x)$.

2.6 Reductions

In this section we define *deterministic* Turing reductions between distributional problems. We distinguish errorless and heuristic reductions since average-case classes and heuristic classes use different computational models. Our definition is very similar to [BDCGL92] but here we use Impagliazzo style definition (and computational model) and ensembles of distributions while [BDCGL92] used Levins definition and distributions on the set of all binary strings.

Definition 2.2 (cf. [BDCGL92]). Distributional problem (L, D) is *errorless reducible* to problem (L', D'), if there exists a *deterministic* algorithm with oracle $\mathcal{T}^{L'}(x;\delta)$ with the following properties

1. (Effectiveness) Running time of $\mathcal{T}^{L'}(x;\delta)$ is $poly(\frac{|x|}{\delta})$.
2. (Correctness) $\mathcal{T}^{L'}(x,\delta) \in \{L(x), \perp\}$ for all x in the support of D.
3. (Usefulness) $\Pr_{x\leftarrow D_n}\{\mathcal{T}^{L'}(x,\delta) = \perp\} < \delta$ for all $n \in \mathbb{N}$.
4. (Domination) There exists polynomial $p(n)$ and subset $E_n \subseteq \{0,1\}^n$ of small measure $D_n(E_n) \leq \delta$ such that $\sum_{x\in\{0,1\}^n\setminus E_n} Ask_{\mathcal{T},\delta}(x,y)D_n(x) \leq p(\frac{n}{\delta})D'(y)$, where $Ask_{\mathcal{T},\delta}(x,y) = 1$ if $\mathcal{T}^{L'}(x,\delta)$ asks oracle for string y and $Ask_{T,\delta}(x,y) = 0$ otherwise. Informally speaking, E_n is the small set where $\mathcal{T}$ makes "incorrect" requests to the oracle.

We say that the distributional problem (L, D) is *heuristically reducible* to (L', D'), if we eliminate the correctness condition and substitute the usefulness condition by

- (Usefulness) $\Pr_{x\leftarrow D_n}\{\mathcal{T}^{L'}(x,\delta) \neq L(x)\} < \delta$.

The following lemma shows that the reductions defined above are reasonable.

Lemma 2.1. (1) If (L, D) is errorless reducible to (L', D') and $(L', D') \in$ **AvgP**, then $(L, D) \in$ **AvgP**.

(2) If (L, D) is heuristically reducible to (L', D') and $(L', D') \in$ **HeurP**, then $(L, D) \in$ **HeurP**.

Proof. (1) Let (L', D') be solvable by algorithm $\mathcal{A}'(x,\delta)$ in **AvgP** and $\mathcal{T}^{L'}(x,\delta)$ is errorless reduction of (L, D) to (L', D'). Running time of $\mathcal{A}'(x,\delta)$ is bounded by polynomial $q(\frac{|x|}{\delta})$, the number of lengths of oracle queries of $\mathcal{T}^{L'}(x,\delta)$ is bounded by polynomial $f(\frac{|x|}{\delta})$. Consider the sequence of lengths of queries on the input length n: $k_1, k_2, \ldots, k_{f(\frac{n}{\delta})}$.

We define algorithm $\mathcal{A}(x,\delta)$: it executes $\mathcal{T}^{L'}(x,\frac{\delta}{3})$ and simulates $\mathcal{A}'(y,\epsilon(x,y))$ instead of oracle request y, where $\epsilon(n) = \frac{\delta}{3p(\frac{n}{\delta})f(\frac{n}{\delta})}$. If $\mathcal{A}'$ outputs $\perp$, then $\mathcal{A}(x,\delta)$ returns $\perp$. The probability of

answer $\perp$ of the resulting algorithm is estimated as follows:

$$\Pr_{x\leftarrow D_n}\{\mathcal{A}(x,\delta)=\perp\}\leq$$

$$\Pr_{x\leftarrow D_n}\{\mathcal{T}^{L'}(x,\frac{\delta}{3})=\perp\}+D_n(E_n)+\sum_{x\in\{0,1\}^n\setminus E_n}\sum_{\substack{y\in\{0,1\}^*\\ A'(y,\epsilon(n))=\perp}}Ask_{\mathcal{T},\delta}(x,y)D_n(x)\leq$$

$$\frac{\delta}{3}+\frac{\delta}{3}+\sum_{x\in\{0,1\}^n\setminus E_n}\sum_{i=1}^{f(\frac{n}{\delta})}\sum_{\substack{y\in\{0,1\}^{k_i}\\ A'(y,\epsilon(n))=\perp}}Ask_{T,\delta}(x,y)D_n(x)\overset{\text{(Domination)}}{\leq}$$

$$\frac{2\delta}{3}+\sum_{i=1}^{f(\frac{n}{\delta})}\sum_{\substack{y\in\{0,1\}^{k_i}\\ A'(y,\epsilon(n))=\perp}}p(\frac{n}{\delta})D'(y)=\frac{2\delta}{3}+\sum_{i=1}^{f(\frac{n}{\delta})}p(\frac{n}{\delta})\Pr_{y\leftarrow D'_{k_i}}\{A'(y,\epsilon(n))=\perp\}<$$

$$\frac{2\delta}{3}+\sum_{i=1}^{f(\frac{n}{\delta})}p(\frac{n}{\delta})\epsilon(n)=\frac{2\delta}{3}+f(\frac{n}{\delta})\frac{\delta}{3f(\frac{n}{\delta})}=\delta.$$

(2) The proof is much the same as the proof of (1). □

We can make a tighter statement.

Corollary 2.1 (from the proof of Lemma 2.1). Let (L,D) be (1) errorless; (2) heuristically reducible to (L',D') by means of reduction $\mathcal{T}^{L'}(x,\delta)$, the number of query lengths of $T^{L'}(x,\delta)$ on input of size n is bounded by polynomial $f(\frac{|x|}{\delta})$; $p(\frac{|x|}{\delta})$ is polynomial from the domination condition. All oracle queries y satisfy the inequality $|y|\geq\lceil(\frac{1}{\epsilon(|x|,\delta)})^{\frac{1}{c}}\rceil$, where $\epsilon(n,\delta)=\frac{\delta}{3p(\frac{n}{\delta})f(\frac{n}{\delta})}$. Then (1) If $(L',D')\in\mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}$, then $(L,D)\in\mathbf{AvgP}$. (2) If $(L',D')\in\mathbf{Heur}_{\frac{1}{n^c}}\mathbf{P}$, then $(L,D)\in\mathbf{HeurP}$.

Proof. The proof resembles the proof of Lemma 2.1, the difference is as follows. The problem (L',D') is solvable by algorithm $\mathcal{A}'(y)$. For all queries y made by $\mathcal{T}^{L'}(x,\delta)$, the inequality $\Pr_{y\in D'_{|y|}}\{A'(y)=\perp\}<\frac{1}{|y|^c}\leq\epsilon(|x|,\delta)$ is satisfied. □

Corollary 2.2 (from the proof of Lemma 2.1). Let (L,D) be errorless reducible to (L',D') by means of reduction $\mathcal{T}^{L'}(x,\delta)$. Let the size of all oracle requests made by $\mathcal{T}^{L'}(x,\delta)$ be bounded by $\pi(|x|,\frac{1}{\delta})$ for all $x\in\{0,1\}^n\setminus E_n$, where π is some polynomial-time computable function, and $(L',D')\in\mathbf{AvgTime}[\tau(n)]$. Then there exists errorless heuristic scheme $\mathcal{A}(x,\delta)$ for (L,D) with running time bounded by $poly(\frac{|x|}{\delta})\tau(\frac{\pi(|x|,\frac{1}{\delta})}{\delta})$.

Proof. The proof is straightforward calculation of running time of algorithm $\mathcal{A}(x,\delta)$ presented in the proof of Lemma 2.1. The only modification: $\mathcal{A}(x,\delta)$ should answer $\perp$ if size of oracle request of $\mathcal{T}^{L'}(x,\delta)$ is more than $\pi(|x|,\frac{1}{\delta})$ (therefore $x\in E_n$).

□

2.7 Decreasing the error in AvgBPP

In this section we show that the constant $\frac{1}{4}$ in Definition 2.1 may be exponentially decreased.

Proposition 2.2 (Chernoff-Hoefding bound). For $X_1,X_2,\ldots,X_N$ identically and independently distributed such that $X_i\in[0,1]$ and $E[X_i]=\mu$, it holds that $\Pr\{|\frac{\sum_{i=1}^N X_i}{N}-\mu|\geq\epsilon\}\leq 2e^{-2\epsilon^2 n}$.

Lemma 2.2 ([BT06]). Distributional problem (L, D) is contained in the class **AvgBPP** if and only if there exists algorithm $\mathcal{B}$ with two parameters (x, δ), such that the following properties hold:

- (Effectiveness) The running of $\mathcal{B}(x, \delta)$ is bounded by $poly(\frac{n}{\delta})$;
- (Correctness) for all x in the support of D, $\Pr\{\mathcal{A}(x; n, \delta) \notin \{L(x), \perp\}\} < 2^{-\Omega(n)}$, where the probability is given over random bits of algorithm $\mathcal{B}$.
- (Usefulness) $\Pr_{x \leftarrow D_n}\{\Pr\{\mathcal{B}(x, \delta) = \perp\} < 2^{-\Omega(n)}\} \geq 1 - \delta$, where the inner probability is taken over random bits of algorithm $\mathcal{A}$.

Proof. Let (L, D) be contained in **AvgBPP** and solvable by $\mathcal{A}(x, \delta)$ according Definition 2.1. We construct the algorithm $\mathcal{B}$ as follows: we repeat $n = |x|$ times the algorithm $\mathcal{A}(x, \delta)$ and return $\perp$ if at least $\frac{n}{3}$ answers equal $\perp$. If less than $\frac{n}{3}$ answers equal $\perp$, then we output the most frequent answer.

We split all binary strings of length n into 3 sets:

- $X_1 = \{x \in \{0,1\}^n | \Pr\{\mathcal{A}(x, \delta) = \perp\} < \frac{1}{4}\}$,
- $X_2 = \{x \in \{0,1\}^n | \frac{1}{4} \leq \Pr\{\mathcal{A}(x, \delta) = \perp\} \leq \frac{1}{3} + \frac{1}{10}\}$,
- $X_3 = \{x \in \{0,1\}^n | \Pr\{\mathcal{A}(x, \delta) = \perp\} > \frac{1}{3} + \frac{1}{10}\}$.

If $x \in X_1$, then Chernoff bound implies that less than $\frac{1}{3}$ fraction of answers of $\mathcal{A}$ equals $\perp$ with probability $1 - 2^{-\Omega(n)}$. Since $\Pr\{\mathcal{A}(x, \delta) = 1 - L(x)\} < \frac{1}{4}$, Chernoff bound implies that $L(x)$ is the most frequent answer given by $\mathcal{A}$ with probability at least $1 - 2^{-\Omega(n)}$. Therefore, $\Pr\{\mathcal{B}(x, \delta) = L(x)\} \geq 1 - 2^{-\Omega(n)}$.

If $x \in X_2$, then since $\Pr\{\mathcal{A}(x, \delta) = 1 - L(x)\} < \frac{1}{4}$, Chernoff bound implies that $1 - L(x)$ is not the most frequent answer given by $\mathcal{A}$ with probability at least $1 - 2^{-\Omega(n)}$. $\Pr\{\mathcal{B}(x, \delta) \neq 1 - L(x)\} \geq 1 - 2^{-\Omega(n)}$.

If $x \in X_3$, then Chernoff bound implies $\Pr\{\mathcal{B}(x, \delta) = \perp\} \geq 1 - 2^{-\Omega(n)}$.

Definition 2.1 implies $D(X_2 \cup X_3) < \delta$, and for $x \in X_1$ it holds that $\Pr\{\mathcal{B}(x, \delta) = L(x)\} \geq 1 - 2^{-\Omega(n)}$. Hence, $\Pr_{x \leftarrow D_n}\{\Pr\{\mathcal{B}(x, \delta) = \perp\} < 2^{-\Omega(n)}\} \geq 1 - \delta$. □

3 Worst-case vs average-case classes

It is easy to see that **AvgP** $\subseteq$ **HeurP**. Indeed it is sufficient to modify **AvgP** algorithm as follows: output 0 instead of answer $\perp$. The similar modification and Lemma 2.2 imply **AvgBPP** $\subseteq$ **HeurBPP**. Whether these inclusions are proper or not is an important open question [Imp95].

Definition 3.1. Let $\mathfrak{C}$ be the class of *languages* and $\mathfrak{D}$ be the class of distributions. $(\mathfrak{C}, \mathfrak{D}) = \{(L, D) | D \in \mathfrak{D}, \exists L' \in \mathfrak{C} : \forall x \in \operatorname{supp} D\ L(x) = L'(x)\}$.

Lemma 3.1. Let f and g be time-constructible functions[2] satisfying $f(n) log^3 f(n) = o(g(n))$. Then there exists unary (i.e., a subset of $\{0\}^*$) language L that separates **DTime**$[f(n)]$ and **DTime**$[g(n)]$.

Proof. We enumerate all deterministic Turing machines with running time bounded by $f(n) \log f(n)$ in such a way that every Turing machine appears in this enumeration infinitely many times: M_i. We consider language $L = \{0^i | M_i(0^i) = 0\}$. $L \in$ **DTime**$[g(n)]$ since it can be solved using a

[2] $f(n)$ is called time-constructible if the value of $f(n)$ can be computed in $O(f(n))$ steps.

simulation. Suppose that L is solved by $O(f(n))$-time Turing machine that has number k in our enumeration. We consider 2 cases: let $0^k \in L$, then $M_k(0^k) = 1$ since M_k solves L, on other hand by definition of L it should be $M_k(0^k) = 0$, contradiction. Let $0^k \notin L$, then $M_k(0^k) = 0$ since M_k solves L, on other hand by definition of L it should be $M_k(0^k) = 1$, contradiction. □

Theorem 3.1. The following inclusions hold:

1. $(\mathbf{P}, U) \subsetneq (\mathbf{AvgP}, U) \subseteq (\mathbf{HeurP}, U)$;
2. $(\mathbf{HeurP}, \mathbf{PSamp}) \subseteq (\mathbf{EXP}, \mathbf{PSamp})$;
3. There exists language $L_{EXP} \in \mathbf{EXP}$ such that for any distribution $D \in \mathbf{PSamp}$, distributional problem (L_{EXP}, D) is not contained in $(\mathbf{HeurP}, \mathbf{PSamp})$.

Proof. 1. Lemma 3.1 implies that there exists the unary language L_p that separates $\mathbf{DTime}[2^{n/2}]$ and $\mathbf{DTime}[2^n]$. $(L_P, U) \notin (\mathbf{P}, U)$ since $L \notin \mathbf{DTime}[2^{n/2}]$.

Now we show that $(L_p, U) \in \mathbf{AvgP}$. Algorithm $\mathcal{A}(x, \delta)$ returns 0 if $x \neq 0^n$. If $x = 0^n$ and $\delta > \frac{1}{2^n}$, then $\mathcal{A}(x, \delta)$ returns $\perp$, and if $\delta \leq \frac{1}{2^n}$ it simulates the machine M that solves L_P in $\mathbf{DTime}[2^n]$ on input 0^n and inverts the result. The running time of $\mathcal{A}(x, \delta)$ is bounded by $O(\frac{x}{\delta^2})$ (if $\delta \leq 2^{-n}$, then $\mathcal{A}(x, \delta)$ may work for 2^{2n} steps).

2. Let distributional problem $(L, D) \in (\mathbf{HeurP}, \mathbf{PSamp})$ be solved by algorithm $\mathcal{A}(x, \delta)$, the distribution D is generated by sampler S with running time bounded by polynomial $q(n)$. Note that every positive value of D on $\{0,1\}^n$ is at least $2^{-q(n)}$. Algorithm $\mathcal{A}(x, \frac{1}{2^{q(|x|)+1}})$ works exponential time and solves $L \cap \operatorname{supp} D$ without errors.

3. We enumerate all deterministic Turing machines with running time bounded by 2^n in such a way that every Turing machine appears in this enumeration infinitely many times: M_i. We consider language $L_{EXP} = \{x | M_{|x|}(x) = 0\}$. Using just simulation we get $L_{EXP} \in \mathbf{EXP}$. Assume that there exists some distribution D such that (L_{EXP}, D) is solved in $\mathbf{HeurP}$ by algorithm $\mathcal{A}(x, \delta)$. We consider a Turing machine that corresponds to the algorithm $\mathcal{A}(x, \frac{1}{10})$; it has number k in our enumeration. Since $\Pr_{x \leftarrow D_k}\{M_k(x) = A(x, \frac{1}{10}) = L_{EXP}(x)\} \geq 0.9$, there exists $x_0 \in \{0,1\}^k$ such that $M_k(x_0) = L_{EXP}(x_0)$, it contradicts the definition of the language L_{EXP}.

□

Now we prove a similar theorem for randomized classes:

Theorem 3.2. The following inclusions hold:

1. $(\mathbf{BPP}, U) \subsetneq (\mathbf{AvgBPP}, U) \subseteq (\mathbf{HeurBPP}, U)$;
2. $(\mathbf{HeurBPP}, \mathbf{PSamp}) \subseteq (\mathbf{BPEXP}, \mathbf{PSamp})$;
3. There exists $L \in \mathbf{BPEXP}$ such that for every distribution $D \in \mathbf{PSamp}$, the distributional problem (L, D) is not contained in $(\mathbf{HeurBPP}, \mathbf{PSamp})$.

Proof. 1. In order to prove $(\mathbf{BPP}, U) \subsetneq (\mathbf{AvgBPP}, U)$, it is sufficient to show that there exists a unary language (i.e., a subset of $\{0\}^*$) that separates classes $\mathbf{BPP}$ and $\mathbf{BPTime}[2^n]$. Prefix **u-** to complexity class $\mathfrak{C}$ means the set of unary languages from $\mathfrak{C}$.

The proof uses ideas from the proof of $\mathbf{BPP} \neq \mathbf{BPTime}[2^n]$ from [KV87].

Assume that $\mathbf{u\text{-}BPP} = \mathbf{u\text{-}BPTime}[2^n]$, then $\mathbf{u\text{-}BPP} = \mathbf{u\text{-}BPTime}[n^{\log n}] = \mathbf{u\text{-}BPTime}[2^n]$.

Lemma 3.2 (cf. [KV87, Lemma 3]). Let $f(n), g(n), h(n)$ be time-constructible functions, $f(n), g(n) \geq \log n$, $h(n) \geq n$ is strictly increasing function. Then **u-BPTime**$[f(n)] \subseteq$ **u-BPTime**$[g(n)]$ implies **u-BPTime**$[f(h(n))] \subseteq$ **u-BPTime**$[g(h(n))]$.

Proof. Let language A be solved in $O(f(h(n)))$ steps by algorithm M, we consider padded version of language A: $A^{pad} = \{x0^{h(|x|)-|x|} | x \in A\}$. The language A^{pad} can be solved in time $O(f(n))$ as follows: given the input y, we use binary search to find such x that $y = x0^{h(|x|)-|x|}$ (in time $O(\log |y|)$). After that we execute M on the input x, the resulting time complexity is $O(\log |y| + f(|x|)) = O(f(|y|))$. Hence the language A^{pad} may be solved in $O(g(|y|))$ steps, therefore language A may be solved in $O(g(h(n)))$ steps. □

Suppose that **u-BPTime**$[n^{\log n}] =$ **u-BPTime**$[2^n]$, consider the following sequence of inclusions:

$$\textbf{u-DTime}[2^{n^2 \log n}] \subseteq \textbf{u-BPTime}[2^{n^2 \log n}] \overset{\text{Lemma 3.2}}{\subseteq}$$
$$\textbf{u-BPTime}[(n^{2 \log n})^{\log(n^2 \log n)}] \subseteq \textbf{u-BPTime}[2^n] \subseteq$$
$$\textbf{u-BPTime}[n^{\log n}] \subseteq \textbf{u-DTime}[2^{n^{\log n}}]$$

Lemma 3.1 implies **u-DTime**$[2^{n^{\log n}}] \subsetneq$ **u-DTime**$[2^{n^2 \log n}]$, we get a contradiction. Besides we may conclude that either L_{BPP} or its padded version separates **u-BPTime**$[n^{\log n}]$ and **u-BPTime**$[2^n]$.

2. The inclusion (**HeurBPP**, **PSamp**) $\subseteq$ (**BPEXP**, **PSamp**) has almost the same proof as the second claim of Theorem 3.1. However, in the randomized case we have one additional problem: the resulting **BPEXP** algorithm should works correctly (with bounded error) even on zero probability inputs. It is possible to compute the probability of input in exponential time (since the distribution is polynomial-time samplable), hence **BPEXP** algorithm can reject all such inputs.

3. Let L' be unary language that separates **u-BPTime**$[n^{\log n}]$ and **u-BPTime**$[2^n]$ (the existence of L' was proved in the claim 1 of this Theorem). We define a new language $L = \{x | 0^{|x|} \in L'\}$. $L' \in$ **BPTime**$[2^n]$ implies that $L \in$ **BPTime**$[2^n]$. Assume that $(L, D) \in$ **HeurBPP** for some polynomial-time samplable distribution D. Let (L, D) be solved by algorithm $\mathcal{A}(x, \delta)$ and the distribution D be generated by sampler $\mathcal{S}$. We will show that $L' \in$ **BPP** and it will be a contradiction.

Consider the following randomized algorithm that solved L': If $x \neq 0^n$, then reject. Otherwise generate $y \leftarrow \mathcal{S}(1^n)$ and return $\mathcal{A}(y, \frac{1}{10})$. The error probability of the described algorithm is bounded by $\frac{1}{10}$ (the probability that $\mathcal{A}(y, \frac{1}{10})$ has unbounded error) $+\frac{1}{4}$ (the error probability of $\mathcal{A}(y, \frac{1}{10})$). That is, $L' \in$ **BPP**, which contradicts the construction of L'. □

Note that classes **AvgP**, **HeurP**, **AvgBPP** and **HeurBPP** are not closed under changing the distribution. Indeed in the proofs of Theorem 3.1 and Theorem 3.2 we prove the existence of unary language L_P such that (L_P, U) separates **P** and **AvgP**, and L_{BPP} such that (L_{BPP}, U) separates **BPP** and **AvgBPP**. If we consider distribution D such that $D_n(0^n) = 1$, then $(L_P, D) \notin$ **HeurP** and $(L_{BPP}, D) \notin$ **HeurBPP**.

4 Complete problem

In this section we construct a distributional problem that is complete for (**AvgBPP**, **PSamp**) under errorless reductions and is complete for (**HeurBPP**, **PSamp**) under heuristic reductions.

The way tuples are encoded to be an input of an algorithm is important in average-case complexity. We may use only logarithmic number of extra bits in encoding, because in this case the uniform probability of a string decreases only polynomially. Now we describe the way we encode tuples:

Remark 4.1. Let x and y be two bit strings. One can encode the pair (x, y) as $0^{\lceil \log |x| \rceil}1|x|_2xy$, where $|x|_2$ is the length of the string x written in binary. It is easy to see that $|(x, y)| = |x| + |y| + 2\lceil \log |x| \rceil + 1$. An m-tuple $z = (x_1, x_2, \ldots, x_m)$ can be encoded as $(x_1, (x_2, (x_3, \ldots (x_{m-1}, x_m) \ldots)$. In this case $|z| = \sum_{i=1}^{m} |x_i| + 2\sum_{i=1}^{m-1} \lceil \log |x_i| \rceil + m - 1 < \sum_{i=1}^{m} |x_i| + 2(m-1)\lceil \log(|z| - |x_m|) \rceil + m - 1$.

We will use Chernoff bound (Proposition 2.2) for several times, for this purpose we fix such a number N_0 that $2e^{-\frac{N_0}{1000}} < 0.001$. Each time we apply Chernoff bound the number of random variables should be at least N_0.

We construct a distributional problem (C, R), where C is a language and R is a polynomial-time samplable distribution. The language C will be defined by the algorithm $\mathcal{A}(x, \delta)$, the distribution R will be defined by the sampler $\mathcal{R}$. We will show that the distributional problem (C, R) is in **AvgBPP** (therefore in **HeurBPP**) and that (C, R) is complete for **AvgBPP** under errorless reductions (in a similar way it is possible to prove that the distributional problem (C, R) is complete for **HeurBPP** under heuristic reductions).

We assume that all Turing machines output only an element from the set $\{0, 1, \perp\}$. Technically we may look on the first 2 bits of the first tape and interpret "00" as 0, "11" as 1, "01" and "10" as $\perp$.

Let us consider the auxiliary algorithm $\mathcal{B}$:

Algorithm 4.1. Algorithm $\mathcal{B}(x, \delta)$:

1. Test input x to be a string of the form $(M, y, 1^m, b)$, where $m > |y| + N_0$, $b \in \{0, 1\}$. If no, then reject. (Here M is an encoding of a randomized Turing machine, y is an input of a Turing machine, m is a number of steps that M is allowed to do, b is an answer that would be outputted instead of an answer $\perp$ of the machine M).

2. - If $\delta > \frac{1}{2^m}$, then execute the machine M on y for m steps for $200m^2$ times. If there exists $c \in \{0, 1\}$ that appears at least 80% times, then return c, otherwise return $\perp$.
   - If $\delta \leq \frac{1}{2^m}$, then go through all sequences of random bits of M and execute M on y for m steps. If the fraction of answers that are equal to $\perp$ is at least $\frac{1}{4}$, then return b. Otherwise return the most frequent answer from {0,1}.

3. Return $\perp$.

Note that algorithm $\mathcal{B}(x, \frac{1}{2^{|x|}})$ is deterministic and it recognizes some language B.

We define an algorithm $\mathcal{A}$ by means of the algorithm $\mathcal{B}$; $\mathcal{A}$ applies $\mathcal{B}$ to some part of the input and ignores the rest of the input.

Algorithm 4.2. Algorithm $\mathcal{A}(x, \delta)$:

1. Test input x to be a string of the form $(M, y, 1^m, b, S, 1^s)$, where $b \in \{0, 1\}$. If no, then reject.

2. Return $\mathcal{B}((M, y, 1^m, b), \delta)$.

Since algorithm $\mathcal{B}(x, \frac{1}{2^{|x|}})$ is deterministic, $\mathcal{A}(x, \frac{1}{2^n})$ is also deterministic and it recognizes some language C; we are going to construct a distribution R such that the resulting distributional problem (C, R) is complete for **AvgBPP**. Informally speaking, if machine M accepts y for m steps, then C contains strings $(M, y, 1^m, b, S, 1^s)$ for all $b \in \{0,1\}, S$ and s; if M outputs $\perp$ on input y for m steps, then exactly one string from $\{(M, y, 1^m, 0, S, 1^s), (M, y, 1^m, 1, S, 1^s)\}$ is contained in C. If M has unbounded error on the input y, then the result may be arbitrary, but the probability of such bad inputs is small because of the choice of the distribution R.

The next Lemma shows conditions on the distribution H under which the problem (C, H) is solvable by algorithm $\mathcal{A}(x, \delta)$ in **AvgBPP**.

Lemma 4.1. Let distribution H satisfy the following property: for every Turing machine M, if the probability of the most frequent answer from $\{0,1\}$ of M on input y for m steps is at most 0.85, then $H(M, y, 1^m, b, S, 1^s) \leq 2e^{-n^2}$, where $n = |(M, y, 1^m, b, S, 1^s)|$. Then $(C, H) \in$ **AvgBPP**.

Proof.

- (Effectiveness) If $\delta > \frac{1}{2^m}$, then running time of $\mathcal{A}$ is bounded by $O(n^4)$. If $\delta \leq \frac{1}{2^m}$, then running time of $\mathcal{A}$ is bounded by $O(\frac{n}{\delta^2})$.
- (Correctness) Let $\delta > \frac{1}{2^m}$ (otherwise the algorithm $\mathcal{A}$ works deterministically and always outputs the correct answer). If the probability of the most frequent answer from $\{0,1\}$ of M on y for m steps is not greater than 0.75, then Chernoff bound implies $\Pr\{\mathcal{A}((M, y, 1^m, b, S, 1^s), \delta) = \perp\} \geq 0.99$. Otherwise, if the probability of the most frequent answer $c \in \{0,1\}$ is greater than 0.75, then $c = C(M, y, 1^m, b, S, 1^s)$. In this case Chernoff bound implies $\Pr\{\mathcal{A}((M, y, 1^m, b, S, 1^s), \delta) = 1 - C(M, y, 1^m, b, S, 1^s)\} < 0.01$.
- (Usefulness) Let $\delta > \frac{1}{2^m}$ (otherwise the algorithm $\mathcal{A}$ works deterministically and does not output $\perp$). If the probability of the most frequent answer from $\{0,1\}$ of M on y for m steps is greater than 0.85, then Chernoff bound implies $\Pr\{\mathcal{A}((M, y, 1^m, b, S, 1^s), \delta) = \perp\} < 0.01$. Otherwise by the statement of the Lemma $H(M, y, 1^m, b, S, 1^s) \leq 2e^{-n^2}$. The total probability of all such inputs Z may be estimated as follows: $H(z) \leq e^{-n^2} 2^{n+1} \leq 2^{-n} < \delta$ (for $n > N_0$). □

We define the distribution R by the sampler $\mathcal{R}$ (this distribution will be used in our complete problem).

Algorithm 4.3. Sampler $\mathcal{R}(1^n)$:

1. Generate string w of length n. If it is not of the form (M, y, r, b, S, σ), where $b \in \{0,1\}$, then return w.
2. Execute sampler S on the input $1^{|y|}$ for $|\sigma|$ steps. Let x denote the result of S.
3. Execute M on x for $|r|$ steps for $200n^2$ times. If each answer from $\{0,1\}$ appears less than 90% times, return 1^n. (Note that by Remark 4.1 the string 1^n does not encode any tuple.)
4. Return $(M, x, 1^{|r|}, b, S, 1^{|\sigma|})$.

Lemma 4.2. Let sampler S correspond to a distribution D. Let $z = (M, x, 1^m, b, S, 1^s)$, $n = |z|$. (1) If machine M on input x for m steps outputs some answer from $\{0,1\}$ with probability not less than 0.95, then $R(z) \geq (1 - 2e^{-n^2}) D(x) 2^{-10 \log(n-s) - 5} \cdot 2^{-|M| - |S|}$. (2) If machine M on input x for m steps outputs some answer form $\{0,1\}$ with probability not greater than 0.85, then $R(z) \leq 2e^{-n^2}$.

Proof. (1) With probability at least $2^{-10\log(n-|\sigma|)-5} \cdot 2^{-|M|-|S|}$ the sampler $\mathcal{R}$ generates string (M, y, r, b, S, σ) on the first step, where $|y| = |x|, |r| = m, |\sigma| = s, b \in \{0,1\}$. (By the Remark 4.1 at most $10\log(n-|\sigma|)-5$ of bits are used to determine the lengths of tuple's items). With probability $D(x)$ the sampler S outputs x on the second step of the sampler $\mathcal{R}$. Chernoff bound implies that the test on the third step of the sampler $\mathcal{R}$ will be passed with probability at least $(1-2e^{-n^2})$.

(2) Chernoff bound implies that the test on the third step of the sampler $\mathcal{R}$ will be passed with probability at most $2e^{-n^2}$. □

The claim (2) of Lemma 4.2 implies that the distribution R satisfies the condition of Lemma 4.1.

Theorem 4.1. $(C, R) \in \mathbf{AvgBPP}$.

Proof. The theorem follows from the claim (2) of Lemma 4.2 and Lemma 4.1. □

Remark 4.2. Assume that the Turing machine M has two inputs: the string x and the rational number $\delta \in (0,1)$. Let M_δ be the Turing machine that simulates M with the value of the second parameter being equal to $\frac{1}{\lceil \frac{1}{\delta} \rceil}$. We may encode M_δ as the pair $(M, \lceil \frac{1}{\delta} \rceil)$, where $\lceil \frac{1}{\delta} \rceil$ is written in binary. By Remark 4.1 $|(M, \lceil \frac{1}{\delta} \rceil)| = |M| + \lceil \log \lceil \frac{1}{\delta} \rceil \rceil + 2\lceil \log |M| \rceil + 1$, and hence $2^{|M_\delta|} \leq 2^{|M|+3} M^2 (\frac{1}{\delta} + 1)$.

Theorem 4.2. (C, R) is a complete problem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ under errorless reductions.

Proof. Let us consider distributional problem (L, D) from **AvgBPP**. It is solvable by a machine M_δ which running time is bounded by polynomial $g(\frac{|x|}{\delta})$. (We assume that the both constants in the Definition 2.1 are decreased to 0.01 by Lemma 2.2). Let distribution D be generated by the sampler S with running time bounded by polynomial $q(n)$.

We describe a reduction in terms of Definition 2.2. The reduction $\mathcal{T}^C(x, \delta)$ makes 2 requests to the oracle: $z_0 = (M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 0, S, 1^{q(|x|)})$ and $z_1 = (M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 1, S, 1^{q(|x|)})$. If the answers of the oracle are different, then return $\perp$. Otherwise return the answer of the oracle.

Let us verify all conditions of a reduction:

1. (Effectiveness) follows from the fact that strings z_0 and z_1 have lengths bounded by $poly(\frac{|x|}{\delta})$.

2. (Correctness) If $C(z_0) \neq C(z_1)$, then $\mathcal{T}^C(x, \delta) = \perp$. If $C(z_0) = C(z_1)$, then $\Pr\{M_\delta(x) = \perp\} < \frac{1}{4}$. By Definition 2.1 (with the decreased constants) $\Pr\{M_\delta(x) \in \{L(x), \perp\}\} \geq 0.99$, hence $\Pr\{M_\delta(x) = L(x)\} \geq 0.74$. By construction $C(z_0)$ is the most frequent answer of M_δ on input x for $g(\frac{x}{\delta})$ steps, therefore $C(z_0) = L(x)$ and $\mathcal{T}^C(x, \delta) = L(x)$.

3. (Usefulness) $\Pr_{x \leftarrow D_n}\{\mathcal{T}^C(x, \delta) = \perp\} = \Pr_{x \leftarrow D_n}\{\Pr\{M_\delta(x) = \perp\} \geq \frac{1}{4}\} < \delta$.

4. (Domination) Let $E_n = \{x \in \{0,1\}^n \mid \Pr\{M_\delta(x) = \perp\} \geq 0.01\}$. By definition of M_δ we have $D(E_n) < \delta$. If δ is fixed, then x is uniquely determined by z_0 and z_1. By correctness of M_δ for every $x \in \{0,1\}^n$, $\Pr\{M_\delta(x) = 1 - L(x)\} < 0.01$ and for every $x \in \{0,1\}^n \setminus E_n$, $\Pr\{M_\delta(x) = \perp\} < 0.01$, therefore for every $x \in \{0,1\}^n \setminus E_n$, $\Pr\{M_\delta(x) = L(x)\} > 0.98$. By the claim (1) of Lemma 4.2 for $x \in \{0,1\}^n \setminus E_n$ we have

$$R(z_i) \geq 0.99 D(x) 2^{-5\log(n' - q(|x|)) - 10} \cdot 2^{-|M_\delta| - |S|},$$

where $n' = |z_0| = |z_1|$, $i \in \{0,1\}$. The last inequality proves the domination condition, since $|S|$ is a constant and $2^{|M_\delta|} \leq 2^{|M|+3} M^2 (\frac{1}{\delta} + 1)$ by Remark 4.2.

□

Corollary 4.1. If $(C, R) \in \mathbf{AvgP}$, then $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Proof. Follows from Theorem 4.2 and Lemma 2.1. □

Theorem 4.3. If $(C, R) \in \mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}$, then $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Proof. The proof resembles the proof of Theorem 4.2. We manually increase the size of the oracle requests in order to use Corollary 2.1.

In terms of the proof of Theorem 4.2 we denote $k = |(M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 0, S, 1)| - 1$ and $p(k, \delta) = \frac{1}{0.99}D(x)2^{5\log(k)+|M_\delta|+|S|+9}$, $p(k, \delta)$ is bounded by $poly(\frac{|x|}{\delta})$. Note that $p(k, \delta)$ is the polynomial from the proof of the domination condition in Theorem 4.2.

We increase the size of the oracle request by adding $\lceil(\frac{1}{\epsilon(n)})^{\frac{1}{c}}\rceil$ ones to the running time of sampler S, where $\epsilon(n) = \frac{\delta}{3p(k,\delta)}$. (If S is a correct sampler, then it stops when it should stop). Here are the new oracle requests: $z_0 = (M_\delta, x, 1^{g(\frac{x}{\delta})+N_0}, 0, S, 1^{q(|x|)+\lceil(\frac{1}{\epsilon(n)})^{\frac{1}{c}}\rceil})$, $z_1 = (M_\delta, x, 1^{g(\frac{x}{\delta})+N_0}, 1, S, 1^{q(|x|)+\lceil(\frac{1}{\epsilon(n)})^{\frac{1}{c}}\rceil})$.

In order to use Corollary 2.1 it is left for us to note that for given δ and $|x|$ the size of the oracle requests is uniquely determined. The polynomial $p(k, \delta)$ in the domination condition does not depend on the last element of z_i (and therefore it is the same as in the proof of Theorem 4.2). □

Corollary 4.2. If $(\mathbf{AvgBPP}, \mathbf{PSamp}) \subseteq (\mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}, \mathbf{PSamp})$, then $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Theorem 4.4.

1. Distributional problem (C, R) is complete for $(\mathbf{HeurBPP}, \mathbf{PSamp})$ under heuristic reductions.
2. If $(C, R) \in \mathbf{HeurP}$, then $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$
3. If $(C, R) \in \mathbf{Heur}_{\frac{1}{n^c}}\mathbf{P}$, then $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$.
4. If $(\mathbf{HeurBPP}, \mathbf{PSamp}) \subseteq (\mathbf{Heur}_{\frac{1}{n^c}}\mathbf{P}, \mathbf{PSamp})$, then $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$.

Proof. The proof is analogous to the proofs of Theorem 4.2, Corollary 4.1, Theorem 4.3 and Corollary 4.2. □

Note that classes $(\mathbf{AvgBPP}, \mathbf{PSamp})$ and $(\mathbf{HeurBPP}, \mathbf{PSamp})$ have the same complete problem (although under the different reductions). In particular, it means that if $(\mathbf{AvgBPP}, \mathbf{PSamp}) \subseteq (\mathbf{HeurP}, \mathbf{PSamp})$, then $(C, R) \in (\mathbf{HeurP}, \mathbf{PSamp})$ and $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$. If $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$, then since $\mathbf{AvgP} \subseteq \mathbf{HeurP}$, we get $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$.

5 Complete problem with uniform distribution implies derandomization

In this section we give some intuition why the resulting complete problem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ is not hard with respect to the uniform distribution, but is hard with respect to somewhat unnatural

samplable distribution. We use ideas from [Gur91], where Gurevich shows that the existence of a complete problem in the distributional **NP** with uniform distribution under deterministic reductions implies **EXP** = **NEXP**. Gurevich used this argument as a motivation for the usage of randomized reductions (but a complete problem for **AvgBPP** under randomized reduction is trivial and useless).

Definition 5.1 ([Gur91]). Distribution D is called *flat* if there exists $\epsilon > 0$ such that for every $x \in \{0,1\}^*$, $D(x) \leq 2^{-|x|^\epsilon}$.

Theorem 5.1. If there exists a problem (L, D) with flat distribution D that is complete for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ under errorless reductions, then $(\mathbf{BPEXP}, U) \subset \mathbf{AvgEXP}$.

Proof. Consider a problem $(X, U) \in (\mathbf{BPEXP}, U)$ that is solvable in $\mathbf{BPTime}[2^{n^c}]$. We define the padded version of language X: $X^{pad} = \{(x, 1^{2^{|x|^c}}) | x \in X)\}$. It is easy to see that $X^{pad} \in \mathbf{BPP}$. We also define the distribution U^{pad} such that $U^{pad}(x, 1^{2^{|x|^c}}) = 2^{-|x|}$; it is obviously polynomial-time samplable. Since $(X^{pad}, U^{pad}) \in (\mathbf{BPP}, U) \subseteq (\mathbf{AvgBPP}, U)$ there is a reduction $\mathcal{T}$ of (X^{pad}, U^{pad}) to (L, D). Let $p(\frac{n}{\delta})$ be the polynomial from the domination condition. We fix some $\delta \in (0, 1)$. Let $\mathcal{T}_\delta$ make up to $f(\frac{n}{\delta})$ requests to the oracle, where n is the length of the input and f is a polynomial. We consider $y = (x, 1^{2^{|x|^c}})$ and the set of requests of $\mathcal{T}(y, \delta)$: $z_1, z_2, \ldots, z_k$, where $k \leq f(\frac{n}{\delta})$. For every $i, 1 \leq i \leq k$, for every $y \in \{0,1\}^n \setminus E_n$ the domination condition implies $D(z_i) \geq \frac{2^{-|x|}}{p(\frac{|y|}{\delta})}$. Since z_i is flat, $2^{-|z_i|^\epsilon} \geq \frac{2^{-|x|}}{p(\frac{|y|}{\delta})}$, hence $|z_i| \leq (|x| \log p(\frac{n}{\delta}))^{\frac{1}{\epsilon}} \leq poly(|x| \log \frac{1}{\delta})$.

Let (L, D) be solvable in **AvgBPP** by an algorithm $\mathcal{A}(x, \delta)$ with running time bounded by a polynomial $g(\frac{n}{\delta})$. We may derandomize $\mathcal{A}(x, \delta)$ in time $2^{poly(\frac{n}{\delta})}$. By Corollary 2.2, (X^{pad}, U^{pad}) is solvable by deterministic errorless heuristic scheme $\mathcal{B}(y, \delta)$ with running time bounded by $poly(\frac{|y|}{\delta}) 2^{(poly(\frac{poly(|x| \log \frac{1}{\delta})}{\delta}))} = poly(\frac{|y|}{\delta}) 2^{poly(\frac{|x|}{\delta})} = 2^{poly(\frac{|x|}{\delta})}$. Finally $\mathcal{B}'(x, \delta) = \mathcal{B}((x, 1^{2^{|x|^c}}), \delta)$ solves (X, U) in **AvgEXP**. □

6 Time hierarchy theorem

In this section we modify techniques from [Per07] to prove a time hierarchy for $(\mathbf{AvgBPP}, \mathbf{PSamp})$.

We consider a sequence n_i such that $n_1 = 1$, $n_{i+1} = 2^{2^{n_i}}$. We split all natural numbers into the segments from n_i to $n_i^* = n_{i+1} - 1$.

For every randomized Turing machine M we denote by $\widehat{M}$ such a machine that on input x executes $M(x)$ for $|x|^2$ times and outputs the most frequent answer. (Here we assume that all Turing machines return only one bit).

We enumerate all Turing machines with the time bound n^{c+1} in such a way that every Turing machine appears in this enumeration infinitely many times: M_i.

We describe a language L that will be used in the proof of a time hierarchy. On lengths form the i-th segment L depends on the Turing machine M_i. If $n_i \leq |x| < n_i^*$, then we identify x and a real number between 0 and 1. We define $\theta_x = \frac{1}{2} + (x - \frac{1}{2})\frac{1}{n^a}$. Let $\pi_n = \Pr_{y \in U(\{0,1\}^{n+1})}\{\widehat{M_i}(y) = 1\}$, where probability is taken over y and the random bits of M_i; $x \in L \iff \theta_x \geq \pi_n$. If $|x| = n_i^*$, then $x \in L \iff \Pr_{y \in U(\{0,1\}^{n_i})}\{\widehat{M_i}(y) = 1\} < \frac{1}{2}$, where probability is taken over y and the random bits of M_i.

We introduce the probability distribution D, that will help us to solve the language L in **AvgBPP**. The most hardest instance of L is x with $\theta_x \approx \pi_n$. We define distribution D in such a

way that such x will have very small probability, therefore **AvgBPP** algorithm will have enough time to solve this instance.

We define the distribution D by the sampler $\mathcal{D}$.

Algorithm 6.1. Sampler $\mathcal{D}(1^n)$:

1. If $n = n_i^*$, then return $x \leftarrow U(\{0,1\}^n)$;
2. Execute $\widehat{M}_i$ on the random input of length $n+1$ for $10^6 n^{2a+c+10}$ times and calculate frequency $\widetilde{\tau}_n$ of answer 1.
3. We call a string x *bad* if $|\theta_x - \widetilde{\tau}_n| < \varepsilon = \frac{1}{100n^a}$ and we call it *good* otherwise.
4. Repeat n^{c+4} times:
   - Generate $x \leftarrow U(\{0,1\}^n)$;
   - If x is good, return x;
5. Return $x \leftarrow U(\{0,1\}^n)$.

Now we verify that distribution D has two desired properties:

Lemma 6.1. Let $n_i \le n < n_i^*$. (1) For all $x \in \{0,1\}^n$ $D(x) \le 2^{-n}\frac{1}{1-\alpha}$, where $\alpha = \frac{1}{n^a+2}$. (2) If $|\theta_x - \pi_n| \le \frac{1}{2^n}$, then $D(x) \le 2^{-n^{c+2}}$.

Proof. (1) The probability that uniformly generated random string is bad is less than $2\varepsilon = \frac{1}{50n^a} < \alpha = \frac{1}{n^a+2}$. The probability that a string x is generated in the first iteration of step 4 of sampler $\mathcal{D}$ is 2^{-n}; the probability that x is generated on the second iteration is less then $\alpha 2^{-n}\alpha$, and so on. Totally $D(x) \le 2^{-n}(1 + \alpha + \alpha^2 + \dots) \le 2^{-n}\frac{1}{1-\alpha}$.

(2) The probability that $|\pi_n - \widetilde{\tau}_n| \le \frac{\varepsilon}{2}$ is at least $1 - 2e^{-2n^{c+10}} \ge 1 - 2^{-n^{c+3}}$ by Chernoff bound. Therefore if $|\theta_x - \pi_n| \le \frac{1}{2^n}$, then with probability at least $1 - 2e^{-2n^{c+10}} \ge 1 - 2^{-n^{c+3}}$ the string x is bad and $D(x) \le 2^{-n^{c+3}} + 2^{-n^{c+3}} \le 2^{-n^{c+2}}$.

□

Now we prove that $(L, D) \in$ **AvgBPP** and $(L, D) \notin$ **Heur**$_{\frac{1}{2}-\frac{1}{n^a}}$**BPTime**$[n^c]$.

Theorem 6.1. $(L, D) \in$ **AvgBPP**

Proof. We show that distributional problem (L, D) is solvable by $\mathcal{L}(x, \delta)$ in **AvgBPP**:

1. If $|x| = n_i^*$, then execute $\widehat{M}_i$ on all inputs of length n_i and with all sequences of random bits and return the most infrequent answer.

   Now $n_i \le |x| < n_i^*$.

2. If $\delta \ge 2^{-n^{c+2}}$
   (a) If $n_i \le |x| < n_i^*$, then execute $\widehat{M}_i$ on $K = 1024\frac{n^a}{\delta^2}$ random inputs of length $n+1$ and compute $\widetilde{\pi}_n$ that is the frequency of answer 1.
   (b) If $\theta_x \ge \widetilde{\pi}_n + \frac{\delta}{16n^a}$, return 1.
   (c) If $\theta_x \le \widetilde{\pi}_n - \frac{\delta}{16n^a}$, return 0.
   (d) Return $\perp$.

3. If $\delta < 2^{-n^{c+2}}$, then compute π_n deterministically, that is execute $\widehat{M}_n$ on all inputs of length $n+1$ with all sequences of random bits. If $\theta_x \geq \pi_n$, return 1, otherwise return 0.

The running time of $\mathcal{L}(x,\delta)$ is $poly(\frac{|x|}{|\delta|})$. If $\theta_x \geq \pi_n + \frac{\delta}{8n^a}$, then Chernoff bound implies that with probability at least 0.99 (for large enough n) $\mathcal{L}(x,\delta)$ outputs $1 = L(x)$. If $\theta_x \leq \pi_n - \frac{\delta}{8n^a}$, then Chernoff bound implies that with probability at least 0.99 (for large enough n) $\mathcal{L}(x,\delta)$ outputs $0 = L(x)$. If $\theta_x \geq \pi_n$, then by Chernoff bound the probability of the answer 0 is at most 0.01 and if $\theta_x \leq \pi_n$, then the probability of the answer 1 is at most 0.01.

If $\delta > \frac{1}{2^{n-1}}$, then the probability of the answer $\perp$ may be estimated as: $\Pr_{x \leftarrow D_n}\{\mathcal{L}(x,\delta) = \perp\} \leq \frac{1+2^n\frac{\delta}{4}}{2^n} < \delta$. If $\frac{1}{2^{n^{c+2}}} < \delta \leq \frac{1}{2^{n-1}}$, then $\frac{\delta}{8n^a}$-neighbourhood of π_n contains at most one number θ_x and by the claim (2) of Lemma 6.1 $D(x) < 2^{-n^{c+2}} < \delta$. If $\delta \leq \frac{1}{2^{n^{c+2}}}$, then $\mathcal{L}(x,\delta) \neq \perp$. □

Theorem 6.2. $(L,D) \notin \mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c]$.

Proof. Proof by contradiction. Suppose that problem a (L,D) is solvable by a Turing machine M_k in $\mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c]$.

The claim (1) of Lemma 6.1 implies that for every subset $S \subseteq \{0,1\}^n$, $D(S) \leq \frac{U(S)}{1-\alpha} = \frac{(n^a+2)U(S)}{n^a+1}$. Hence the machine M_k correctly solves L on a set of inputs with uniform measure at least $(\frac{1}{2} + \frac{1}{n^a})\frac{n^a+1}{n^a+2} = (\frac{1}{2} + \frac{1}{2n^a})$.

For every $x \in \{0,1\}^{n_k^*}$, $L(x) = a \in \{0,1\}$. Since M_k solves L on $\frac{1}{2} + \frac{1}{2n^a}$ fraction of inputs, and machine $\widehat{M}_k$ has probability of error at most e^{-n^2} on such inputs, we may conclude that for every $x \in \{0,1\}^{n_k^*-1}$, $L(x) = a$. If we continue this reasoning we get that for every $x \in \{0,1\}^{n_k}$, $L(x) = a$. Hence a is the most frequent answer of $\widehat{M}_k$ on $x \in \{0,1\}^{n_k}$; it contradicts with the choice of a. □

Note, that $\mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c] \supseteq \mathbf{Avg}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c] \supseteq \mathbf{AvgBPTime}[n^c]$. It completes the prove of the time hierarchy theorem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$.

Acknowledgments. The author thanks Dima Grigoriev for bringing his attention to the problem and Edward A. Hirsch for fruitful discussions. The author also thanks Alexei Pastor, Ilya Posov and Kirill Shmakov for useful comments.

References

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 194–208, London, UK, 2002. Springer-Verlag.

[BDCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundation and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *FOCS*, pages 316–324, 2004.

[FS07] Lance Fortnow and Rahul Santhanam. Time hierarchies: A survey. Technical Report 07-004, Electronic Colloquium on Computational Complexity, 2007.

[GHP06] Dima Grigoriev, Edward A. Hirsch, and K. Pervyshev. A complete public-key cryptosystem. Technical Report 06-046, Electronic Colloquium on Computational Complexity, 2006.

[Gur91] Yuri Gurevich. Average case complexity. In *ICALP*, pages 615–628, 1991.

[HH86] Juris Hartmanis and Lane A. Hemachandra. Complexity classes without machines: On complete languages for up. In *ICALP '86: Proceedings of the 13th International Colloquium on Automata, Languages and Programming*, pages 123–135, London, UK, 1986. Springer-Verlag.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT*, pages 96–113, 2005.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *SCT '95: Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, page 134, Washington, DC, USA, 1995. IEEE Computer Society.

[KV87] Marek Karpinski and Rutger Verbeek. *Randomness, provability, and the separation of Monte Carlo time and space*, pages 189–207. Springer-Verlag, London, UK, 1987.

[Lev86] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.

[Mil01] Peter Bro Miltersen. *Handbook on Randomization*, volume II, chapter 19. Derandomizing Complexity Classes. Kluwer Academic Publishers, July 2001.

[Per07] Konstantin Pervyshev. On heuristic time hierarchies. In *IEEE Conference on Computational Complexity*, pages 347–358, 2007.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/