Structure in Average Case Complexity*

Christoph Karg† *Rainer Schuler*

Universität Ulm
Abteilung für Theoretische Informatik
89069 Ulm (Donau)
GERMANY

chkarg@informatik.uni-ulm.de
schuler@informatik.uni-ulm.de

*An extended abstract of this paper will be presented at the Sixth Annual International Symposium on Algorithms and Computation.

†The research of this author was supported by the Deutsche Forschungsgemeinschaft, grant No. Schö 302/4-1.

Running head:

Structure in Average Case Complexity

Correspondence author:

Rainer Schuler
Universität Ulm
Abteilung Theoretische Informatik
89069 Ulm (Donau)
GERMANY

schuler@informatik.uni-ulm.de

Abstract

In 1990 Schapire gave an equivalent characterization of Levin's notion of functions, that are polynomial on average. This characterization gives a very smooth translation from worst case complexity to average case complexity of the notions for time and space complexity. We prove tight space- and time-hierarchy theorems and discuss the structure of deterministic and nondeterministic average case complexity classes. In particular, we show for polynomial and exponential time-bounded classes that nondeterministic average case is equivalent to deterministic average case if this is true in worst case complexity. We consider tally encodings of randomized decision problems and show that there are tally randomized decision problems in average nondeterministic polynomial time which are not in average deterministic polynomial time if and only if average deterministic exponential time is different from average nondeterministic exponential time.

1 Introduction

Despite having bad worst case behavior, many algorithms are frequently used in practice because they are efficient on the average. A well known example is the Simplex algorithm, a worst case exponential time algorithm for linear programming which performs well in practice, even better than worst case polynomial time algorithms for the same problem. It seems that the instances which cause the bad worst case complexity do not occur in practical applications. A similar example, within P, is the Quicksort algorithm. Even though the worst case is $O(n^2)$ for all (deterministic) implementations, the Quicksort algorithm is often used in practice, since its average case complexity is $O(n \log n)$. Thus, in some cases, the average case complexity of a problem is a better measure than its worst case complexity.

A general theory of average case complexity was introduced by Levin. He defined a robust notion of "functions are polynomial on average" with respect to a probability distribution on all instances [Lev84]. Since then this notion has been considered by many researchers. In particular the notion of completeness and various reductions have been studied [Lev86, Gur91, BDCGL92, WB93, RS93, SY95]. The basic objects of average case complexity are randomized decision problems, pairs consisting of a decision (or search) problem and a probability distribution on the instances of the problem. An open problem, a generalization of the famous "P $\stackrel{?}{=}$ NP" question, is whether all sets in NP can be solved deterministically in polynomial time on average under all "natural" (or easy) probability distributions. Levin considered in his papers polynomial time computable distributions (as natural) [Lev84, Lev86]. This notion seemed to be to restrictive and later the more general notion of polynomial time samplable distributions has been proposed in [BDCGL92].

A first important connection between average case complexity and worst case complexity was given in [BDCGL92]. Ben-David *et al.* show that if all problems in NP can be solved deterministically in polynomial time on average under every polynomial time computable distribution, then deterministic linear-exponential time is equal to nondeterministic linear-exponential time. (This indicates that it is unlikely that all sets in NP are polynomial time solvable on average under polynomial time computable distributions.) However, the question whether the above assumption implies that the polynomial time hierarchy PH (or even P^{NP}) can be solved efficiently on average, is not clear [SW95].

As pointed out in [Gur91], using Levin's definition it is reasonable to define the notion of "polynomial on average" and "linear on average". In this paper we generalize Levin's notion and define "g on average" for arbitrary functions g following a characterization of "polynomial on average" given in [Sch90] (see also [SY92, SY95, Imp95]). The time-bound g is here a two placed function, where the first argument is the length of the input and the second argument is the inverse of a probability weight. Now a function f (i.e. the time bound of a Turing machine) is said to be g on average if for every probability weight ε the probability over all x that $f(x)$ exceeds $g(|x|, 1/\varepsilon)$ is smaller than ε. Note that the notion of "polynomial on average" remains unchanged, but now it is possible to consider functions that are g on average but not $o(g)$ on average. This allows to incorporate results from "classical" average case complexity into the framework given by Levin.

We remark at this placd, that different approaches to define "g on μ-average" have been suggested [RS93, CS95]. However, these approaches define notions which are different to Levin's definition of "polynomial time on average".

The aim of our paper is to give a characterization of average case complexity classes that are

similar in structure to worst case complexity and allow to precisely classify problems according to their average case complexity.

The paper is organized as follows:

In section 2 we recall the necessary notions and definitions of average case complexity and give a definition of g on μ-average.

In section 3 we define deterministic time- and space-bounded average case complexity classes and give time and space hierarchy theorems, which are as tight as those, known from worst case complexity theory (see for example [HU79]). It is possible to show the hierarchy theorems under an "easy" (polynomial-time computable) distribution.

In section 4 the definitions of time- and space-bounded computations are extended to nondeterministic Turing machines. We require that the time-bounds of nondeterministic Turing machines are exactly real time computable. That is, a function f is a time-bound if for all x, $0^{f(x)}$ can be computed by a deterministic transducer in time $O(f)$ (cf. [Gol88]). We feel that this requirement is justified since otherwise nondeterministic polynomial time on average is not contained in any deterministic (on average) time-bounded class.

Section 5 compares the structure of average case complexity classes with the structure in worst case complexity. Wang & Belanger show that if P is a proper subset of NP, then deterministic average polynomial time (AP) is properly included in nondeterministic average polynomial time (ANP) and if there is a randomized decision problem which is hard on positive instances (i.e. no Turing machine is polynomial time bounded on average, even if only instances in the set are considered) then P is a proper subset of NP [WB93]. Under the assumption that the time-bounds (of nondeterministic machines) are exactly real time computable it is possible to show that AP = ANP if and only if P = NP. This equivalence is extended to exponential time classes. Furthermore we show a similar relation for nondeterministic polynomial time and deterministic exponential time, i.e. ANP is included in AE if and only if NP is a subset of E. (AE and E denote average linear-exponential time and linear-exponential time resp.) Note that for the "only if" direction the assumption that the time-bounds are exactly real time computable is necessary.

In section 6 we study upward collapse properties of average case complexity classes. In worst case complexity theory it is known that if P = NP, then EXP = NEXP. We show, that the same relationship also holds in average case complexity. If AP is equal to ANP then deterministic average exponential-time (AEXP) is equal to nondeterministic average exponential time (ANEXP). Book showed, that there are tally sets in NP $-$ P if and only if E $\neq$ NE [Boo74]. We define a tally coding of randomized decision problems which allows us to extend this result to average case complexity.

2 Preliminaries

In this paper we use the standard notations and definitions of computational complexity theory (see for example [BDG88, Pap94]). For an introduction on average case complexity the reader is refered to [Gur91].

Let $\Sigma = \{0, 1\}$ be fixed and Σ^* denote the set of all finite strings over Σ. For every $x \in \Sigma^*$, let $|x|$ denote the length of x. A set (language) is always a subset of Σ^*. The cardinality of a set X is denoted by $||X||$. We use a standard pairing function $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ that is computable and invertible in polynomial time, defined as $\langle x, y \rangle = x_1 0 x_2 0 \ldots 0 x_{|x|} 1 y$, where $x = x_1 x_2 \ldots x_{|x|}$. This function is extended to n-tuples in the usual way, e.g. $\langle x, y, z \rangle = \langle x, \langle y, z \rangle \rangle$.

A total function μ from Σ^* to $[0,1]$ is called a probability function (or density function), if $\sum_x \mu(x) = 1$. The probability distribution μ^* of μ is given by $\mu^*(x) = \sum_{y \leq x} \mu(y)$, for all x. Intuitively, if the random variable X is defined by the elementary event that a string $x \in \Sigma^*$ is chosen under some distribution μ^*, then $\mu(x)$ is the probability that x is selected, i. e. $\mu(x) = \mathrm{Prob}[X = x]$. Without loss of generality we allow probability functions to converge to a constant $c \neq 1$. For a set $X \subseteq \Sigma^*$ let $\mu(X) = \sum_{x \in X} \mu(x)$.

For a Turing machine M let $M(x)$ denote the output of M on input x. M is said to accept x if $M(x) = 1$ and $L(M)$ denotes the set of strings accepted by M. A Turing machine M is said to accept a set (language) L if $L = L(M)$.

The conditional probability function μ_n of μ is defined as

$$\mu_n(x) = \begin{cases} \mu(x)/\mu(\Sigma^n), & \text{if } |x| = n \text{ and } \mu(\Sigma^n) > 0 \\ 0 \quad\quad\quad\quad\quad\quad , & \text{otherwise} \end{cases}$$

That is $\mu_n(x)$ is equal to the probability, that x occurs under the assumption that the length of the string is n.

A distribution μ^* is called positive, if for all but finitely many x, $\mu(x) > 0$. It is called the standard distribution if

$$\mu(x) = \frac{1}{|x| \cdot (|x|+1) \cdot 2^{|x|}}.$$

A distribution μ^* is computable, if there exists a transducer M, such that for every $x \in \Sigma^*$, $|\mu^*(x) - M(x, 0^k)| < 2^{-k}$ [KF82]. If M is polynomial time-bounded in $|x| + k$, then we say that μ is ptime-computable.

The expected value of a total function $f : \Sigma^* \to \mathbb{R}_0^+$ with respect to a probability function μ is

$$\mathrm{Exp}_\mu[f(x)] = \sum_{x \in \Sigma^+} \mu(x) \cdot f(x).$$

Let A be a boolean predicate, i.e. a function from Σ^* to $\{0,1\}$. The probability of A with respect to μ, denoted by $\mathrm{Prob}_\mu[A(x)]$ is equal to the expected value of A with respect to μ,

$$\mathrm{Prob}_\mu[A(x)] = \mathrm{Exp}_\mu[A(x)].$$

A randomized decision problem is a pair (D, μ) consisting of a recursive language D and a computable probability function μ. An average case complexity class is a set of randomized decision problems. For example the class DistNP [Lev84, BDCGL92, Gur91] is the set of all randomized decision problems (D, μ), where $D \in$ NP and μ^* is ptime-computable.

The complement of a randomized decision problem (D, μ) is $(\overline{D}, \mu)$, where $\overline{D} = \Sigma^* - D$. If $\mathcal{C}$ is a class of randomized decision problems, then co-$\mathcal{C}$ $= \{(\overline{D}, \mu) \mid (D, \mu) \in \mathcal{C}\}$ is called the complement of $\mathcal{C}$.

The foundation of average case complexity is Levin's notion of "functions are polynomial on average".

Definition 2.1 ([Lev84]) *A function f is* polynomial on average with respect to a probability function μ *(or short:* polynomial on μ-average*), if there exists a constant $0 < \varepsilon \leq 1$, such that*

$$\sum_{x \in \Sigma^+} \frac{f(x)^\varepsilon \cdot \mu(x)}{|x|} < \infty.$$

If $\varepsilon = 1$ then f is called linear on μ-average.

The set of average polynomial functions has similar closure properties as the set of polynomials [Lev84, Gur91].

If μ is clear from the context, then functions that are polynomial on μ-average are simply called average polynomial functions.

In this paper we use an equivalent characterization of "polynomial on average" given in [Sch90]. He shows, that a function f is polynomial on μ-average (in the sense of Levin as defined above) if and only if there exists a polynomial p from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{R}_0^+$, such that for all $m > 0$

$$\text{Prob}_\mu[f(x) > p(|x|, m)] < \frac{1}{m}.$$

Using Schapire's characterization it is possible to generalize the notion of "polynomial on average" to "g on average" for arbitrary functions g.

Definition 2.2 (g on μ-average) *Let f be a total function from Σ^* to $\mathbb{R}_0^+$ and μ be a probability function. f is g* on μ-average *for a function g from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{R}_0^+$, if for all $m > 0$*

$$\text{Prob}_\mu[f(x) > g(|x|, m)] < \frac{1}{m}.$$

The definition takes into account that the average case measure depends not only on the given instance x but also on the probability $\mu(x)$ of the occurrence of x. If the instance x does not appear, i. e. $\mu(x) = 0$, it has no affect on the average case analysis. Similar if for some string x, $\mu(x) > 0$ then

$$f(x) \leq g(|x|, \lceil 1/\mu(x) \rceil). \tag{1}$$

The following propositions are examples that the definition of "g on average" is reasonable to analyse the average case complexity of algorithms. First it is shown that if a function has uniform complexity (on strings of the same length), then worst case complexity is equal to average case complexity under any distribution.

Proposition 2.3 *Let f, g be functions from $\mathbb{N}$ to $\mathbb{N}$ and $g'(n, m) = g(n) \cdot g(m)$.*

- *If $f(x) = g(|x|)$ for all $x \in \Sigma^*$, then f is g' on μ-average for any density function μ.*
- *If $f(x) > g'(|x|, 1)$ for all $x \in \Sigma^*$, then f is not g' on μ-average for any density function μ.*

Similar it is possible to show that Quicksort has average case complexity $\Theta((n \log n) \cdot (m \log m))$ under a standard distribution. The intuition of this result is, that for a constant c and every m the probability that on random input x the running time of Quicksort exeeds $c(|x| \cdot \log |x|) \cdot (m \cdot \log m)$ is smaller than $1/m$. On the other hand there exist some constants c and m such that the probability that running time of Quicksort exeeds $(|x| \log |x|) \cdot (m \log m)/c$ is larger than $1/m$.

As input to Quicksort we consider permutations on an initial sequence of the natural numbers. For every $n_0 \in \mathbb{N}$ a standard distribution λ can be defined as follows.

$$\lambda(x) = \begin{cases} \frac{1}{(n-n_0)(n-n_0+1)n!} & , \text{ if } x \text{ is a permutation of } (1, \ldots, n) \text{ and } n > n_0 \\ 0 & , \text{ otherwise.} \end{cases}$$

Proposition 2.4 *Let $g(n, m) = (n \log n) \cdot (m \log m)$. The Quicksort algorithm is $c \cdot g$ on λ-average but not g/c on λ-average, for some constant c and a standard distribution λ.*

Proof. Let $Q(x)$ denote the running time of the Quicksort algorithm on input x. We use the following Lemma. Recall, that $\sum_{x\in\Sigma^n} Q(x)\lambda_n(x) = \Theta(n\log n)$.

Lemma 2.5 *Let λ_n denote the uniform distribution on permutations of $(1,\ldots n)$. There exist constants c_1, c_2 and n_0, such that for all $n > n_0$*

(1) $\forall m > 0 :\ \mathrm{Prob}_{\lambda_n}[Q(x) > c_1\ m\ n\log n] < 1/m$,

(2) $\mathrm{Prob}_{\lambda_n}[Q(x) > (n\log n)/c_2] > 1/c_2$.

Proof. The proof of (1) follows from Markov's inequality.
The proof of (2) is by a standard counting argument. Note that every computation of Quicksort corresponds to exactly one input sequence. There are $n!$ many possible input sequences but only $2^{(n\log n)/c_2}$ computations of Quicksort of length less than $(n\log n)/c_2$. Thus less than $n!/c_2$ many input sequences are computed by Quicksort in time $(n\log n)/c_2$. □

Let c_1, c_2, and n_0 be the constants from Lemma 2.5, let $c = \max\{c_1, (c_2)^2\log c_2\}$ and let λ be a standard distribution for n_0 as defined above.

First we show that Q is $c\cdot g$-bounded on λ-average. For any constant $m > 0$ it holds

$$\begin{aligned}
\mathrm{Prob}_\lambda[Q(x) > c\cdot g(|x|,m)] &\le \mathrm{Prob}_\lambda[Q(x) > c\ m\ n\log n] \\
&= \sum_{n\ge n_0} \mathrm{Prob}_\lambda[|x| = n \wedge Q(x) > c\ m\ n\log n] \\
&= \sum_{n\ge n_0} \frac{1}{(n-n_0)(n-n_0+1)} \mathrm{Prob}_{\lambda_n}[Q(x) > c\ m\ n\log n] \\
&< \frac{1}{m}\cdot\sum_{n\ge n_0}\frac{1}{(n-n_0)(n-n_0+1)} \\
&= \sum_{n\ge 1}\frac{1}{n(n+1)} = \frac{1}{m}.
\end{aligned}$$

To show, that Q is not g/c-bounded on λ-average, let $m = c_2$.

$$\begin{aligned}
\mathrm{Prob}_\lambda\left[Q(x) > \frac{g(|x|,c_2)}{c}\right] &= \sum_{n\ge n_0}\mathrm{Prob}_\lambda\left[|x| = n \wedge Q(x) > \frac{(n\log n)(c_2\log c_2)}{c}\right] \\
&\ge \sum_{n\ge n_0}\frac{1}{(n-n_0)(n-n_0+1)}\mathrm{Prob}_{\lambda_n}\left[Q(x) > \frac{n\log n}{c_2}\right] \\
&> \frac{1}{c_2}.
\end{aligned}$$

□

Now we use the definition "g on average" to define several classes of functions.

Definition 2.6 *Let f by a function from $\mathbb{N}$ to $\mathbb{R}_0^+$. f is called*

- linear on μ-average, *if there exist constants $c, d \in \mathbb{R}_0^+$, such that f is $c\cdot n\cdot m + d$ on μ-average.*
- polynomial on μ-average, *if there exists a polynomial $p : \mathbb{N}\times\mathbb{N} \to \mathbb{R}_0^+$ such that f is p on μ-average.*

- linear-exponential on μ-average, *if there exist constants* $c, d \in \mathbb{R}_0^+$, *such that* f *is* $2^{c \cdot n \cdot m + d}$ *on* μ*-average.*
- exponential on μ-average, *if there exists a polynomial* $p : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_0^+$ *such that* f *is* 2^p *on* μ*-average.*

As mentioned above, the set of polynomial on average functions is closed under addition, multiplication and exponentiation with a constant [Lev84, Gur91]. The same also holds for the set of linear-exponential (exponential, resp.) functions.

Here we give a proof of the closure under exponentiation with a constant. Let f be exponential on μ-average and $c \in \mathbb{R}_0^+$ a constant. Then, by definition, there is a polynomial p, such that for all $m > 0$

$$\mathrm{Prob}_\mu\left[f(x) > 2^{p(|x|,m)}\right] < \frac{1}{m}.$$

Then for all $m > 0$ it holds, that $\mathrm{Prob}_\mu[f(x)^c > p(|x|, m)^c] < 1/m$. Since $(2^{p(|x|,m)})^c$ is an exponential function, $f(x)^c$ is exponential on μ-average. The same argument holds for linear-exponential functions.

3 Deterministic average case complexity

In average case complexity theory we miss one property, which is generally used in worst case complexity theory without explicitly refering to it. This is the constructibility of the time-bounds (space-bounds resp.). It is impossible to enumerate all functions which are polynomial on average, since the question whether an arbitrary function is polynomial on average or not, depends on the choice of the probability function. So the well-known diagonalization technique cannot be applied directly in average case complexity.

Nevertheless many techniques of worst case complexity work in average case complexity too. For example, if we know that the running time of a deterministic machine is polynomial on average under the probability function μ, we can use this machine as a clock to make other machines polynomial on μ-average time-bounded.

3.1 Time-bounded computations

For a deterministic Turing machine M, let $time_M(x)$ denote the length of the computation path (the number of transition steps) of M on input x. If the computation of M terminates on every input then $time_M(x)$ is total. We consider only Turing machines that terminate on every input.

A deterministic Turing machine M is polynomial on μ-average time-bounded, if $time_M(x)$ is polynomial on μ-average. This leads to the definition of the class AP, the set of problems that are solvable in polynomial time on average [Lev84, Gur91, BDCGL92, WB93].

Definition 3.1 (AP) *A randomized decision problem* (D, μ) *is in* AP *if* D *is accepted by a deterministic Turing machine, which is polynomial on* μ*-average time-bounded.*

We remark here that there are sets D not in P that are polynomial time solvable on μ-average for every ptime-computable distribution μ, i.e. $(D, \mu) \in$ AP.

In general a deterministic Turing machine M is g on μ-average time-bounded, if $time_M(x)$ is g on μ-average. This notion is used to define deterministic average case time-bounded classes.

Definition 3.2 (ADTIME(g)) *Let g be a function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. The randomized decision problem (D, μ) is in the class* ADTIME(g) *if D is accepted by a deterministic Turing machine, which is g on μ-average time-bounded.*

Note that the function g in the above definition is *not* the (worst case) time-bound of the Turing machine M. If μ is a positive density function and $g(|x|, h(|x|))$ is time-constructible, where

$$h(n) \geq \max_{|x|=n}(\lceil 1/\mu(x) \rceil),$$

then for every D with $(D, \mu) \in$ ADTIME(g) it holds, by equation (1), that

$$D \in \text{DTIME}(g(|x|, h(|x|)).$$

Proposition 3.3 $\text{AP} = \bigcup_{k>0} \text{ADTIME}(n^k m^k + k)$.

Definition 3.4 (AE, AEXP) *A randomized decision problem (D, μ) is in* AE *(*AEXP*, resp.) if D is accepted by a deterministic Turing machine, which is linear-exponential (exponential, resp.) on μ-average time-bounded.*

We get immediately, that

$$\begin{aligned} \text{AE} &= \bigcup_{k>0} \text{ADTIME}(2^{k \cdot n \cdot m + k}) \text{ and} \\ \text{AEXP} &= \bigcup_{k>0} \text{ADTIME}(2^{n^k m^k + k}). \end{aligned}$$

3.2 Space-bounded computations

Let $space_M(x)$ denote the number of tape cells used by the Turing machine M on input x. We say, the Turing machine M is g on μ-average space-bounded, if $space_M(x)$ is g on μ-average. This leads to the definition of ADSPACE(g).

Definition 3.5 (ADSPACE(g)) *Let g be a total function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. The randomized decision problem (D, μ) is in the class* ADSPACE(g) *if D is accepted by a deterministic Turing machine, which is g on μ-average space-bounded.*

We are interested especially in randomized decision problems, which can be computed by polynomial on the average space-bounded Turing machines. We denote the set of these problems by APSPACE, formally defined as

$$\text{APSPACE} = \bigcup_{k>0} \text{ADSPACE}(n^k m^k + k).$$

3.3 Hierarchy theorems

One major result in computational complexity theory was the proof of tight time and space hierarchy theorems. In this section we give similar hierarchy theorems for average case complexity. We define a ptime-computable distribution μ and show that there exists a tight hierarchy of tally sets under the distribution μ. A different approach was followed in [GGH94]. Grape *et al.* show that there exists a hierarchy of languages that are hard to compute on all but polynomial many strings for each length.

Theorem 3.6 (Time Hierarchy Theorem) *Let f and g be total functions from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. If $f(n,m) \geq \log n + \log m$ and*

$$\lim_{n\to\infty} \frac{g(n,n)\log g(n,n)}{f(n,n)} = 0$$

then $\mathrm{ADTIME}(g) \subsetneq \mathrm{ADTIME}(f)$.

Proof. Define the function $h : \mathbb{N} \to \mathbb{N}$ as $h(1) = 4$ and $h(k) = 2^{h(k-1)}$, if $k > 1$. The density function μ over $\{0^{h(k)} \mid k \geq 1\}$ is defined as $\mu(0^{h(1)}) = 7/8$ and

$$\mu(0^{h(k)}) = \sum_{i=h(k-1)}^{h(k)-1} \frac{1}{2^i}.$$

For any $x \notin \{0^{h(k)} \mid k \geq 1\}$, $\mu(x)$ is equal to 0. The functions h and μ provide the following property

$$1 - \sum_{i=1}^{k} \mu(0^{h(i)}) < \frac{1}{h(k)} < \mu(0^{h(k)}). \tag{2}$$

Claim 3.7 *Let $T \subseteq \{0^{h(k)} \mid k \geq 1\}$. If $T \in \mathrm{DTIME}(f(n,n))$ then $(T,\mu) \in \mathrm{ADTIME}(f)$.*

Proof. Let M be a $f(n,n)$ time-bounded deterministic Turing machine with $L(M) = T$. For an arbitrary integer $m > 0$ let

$$k_m = \max\{i \in \mathbb{N} \mid h(i) \leq m\}.$$

Then for all $i \leq k_m$ we get $time_M(0^{h(i)}) \leq f(h(i),h(i)) \leq f(h(i),m)$. We can estimate the probability, that M on input x needs more than $f(|x|,m)$ steps, as:

$$\begin{aligned} \mathrm{Prob}_\mu[time_M(0^n) > f(n,m)] &= 1 - \mathrm{Prob}_\mu[time_M(0^n) \leq f(n,m)] \\ &\leq 1 - \sum_{i=1}^{k} \mu(0^{h(i)}) < \frac{1}{h(k)} \leq \frac{1}{m}. \end{aligned}$$

Thus $(T,\mu) \in \mathrm{ADTIME}(f)$. □

Claim 3.8 *Let $T \subseteq \{0^{h(k)} \mid k \geq 1\}$. If $T \notin \mathrm{DTIME}(g(n,n))$ then $(T,\mu) \notin \mathrm{ADTIME}(g)$.*

Proof. We give a proof by contraposition. Assume, that (T, μ) is a randomized decision problem in ADTIME(g). Then there exists a deterministic Turing machine M with $L(M) = T$, such that for all $m > 0$

$$\text{Prob}_\mu[time_M(0^n) > g(n, m)] < \frac{1}{m}.$$

First note, that $\{0^{h(k)} \mid k \geq 1\}$ in DTIME($\log n$). For every k let $m_k = h(k)$. By inequality (2) it holds $1/\mu(0^{h(k)}) < h(k)$. Thus $time_M(0^{h(k)}) \leq g(h(k), h(k))$. □

Claim 3.9 *There exists a set $T \subseteq \{0^{h(k)} \mid k \geq 1\}$ such that*

$$T \in \text{DTIME}(f(n, n)) - \text{DTIME}(g(n, n)).$$

Proof. We construct the set T by diagonalization. Let M_1, M_2, M_3, ... be an enumeration of all deterministic Turing machines, such that every machine occurs infinitely often. Then T is defined by the following machine.

```
input 0^n;
if (there exists a k such that h(k) = n) then
   do within f(n,n) steps
      simulate M_k on input 0^n;
      if M_k(0^n) halts and accepts then reject
      else accept;
reject;
```

Obviously, it holds $T \in \text{DTIME}(f(n, n)) - \text{DTIME}(g(n, n))$. □

Combining Claim 3.7, Claim 3.8 and Claim 3.9 the theorem follows. □

The fact $f(n, n) = \omega(g(n, n) \log g(n, n))$ is only needed in CLAIM 3.9 to simulate an arbitrary k-tape machine by a fixed 2-tape machine. For details see [HS66].

Corollary 3.10 $\text{AP} \subsetneq \text{AE} \subsetneq \text{AEXP}$.

Corollary 3.11 *For every integer $k > 1$ and any $\varepsilon > 0$ it holds*

$$\text{ADTIME}(n^k m^k) \subsetneq \text{ADTIME}(n^{k+\varepsilon} m^{k+\varepsilon}).$$

A similar theorem can be given for space-bounded computations.

Theorem 3.12 (Space Hierarchy Theorem) *Let f and g be total functions from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. If*

$$\lim_{n \to \infty} \frac{g(n, n)}{f(n, n)} = 0$$

then $\text{ADSPACE}(g) \subsetneq \text{ADSPACE}(f)$.

4 Nondeterministic average case complexity

Before we discuss nondeterministic computations, we fix the complexity measures that will be used. Let $time_N(x)$, the running time of the nondeterministic Turing machine N on input x, be the length of the shortest accepting computation path if N accepts x, and 1 otherwise. Similar, the space needed by the nondeterministic Turing machine N on input x, denoted by $space_N(x)$, is the smallest number of tape cells on any accepting computation path, if N accepts x, and 1 otherwise.

4.1 Computable boundaries

In analogue to [Coo73] we say, a total function $f : \Sigma^* \to \mathbb{N}$ is exactly real time computable if there exists a constant c and deterministic Turing machine which for every input x halts within $c \cdot f(x)$ steps with a string of length $f(x)$ on its output tape.

Note that f depends on x itself and not on the length of x as in Cook's definition of real time computable.

Let g be a total function from $\mathbb{N} \times \mathbb{N} \to \mathbb{R}_0^+$. A nondeterministic Turing machine N is g on the μ-average time-bounded, if there exists an exactly real time computable function f, such that

- $f(x)$ is g on μ-average and
- $time_N(x) \leq f(x)$ for all $x \in L(N)$.

If g is a polynomial, we say N is polynomial on μ-average time-bounded (similar if g is exponential or linear-exponential). A similar definition was proposed in [Gol88].

Forcing the real time computability of the time-bound enables us to simulate nondeterministic computations in a deterministic manner by using f as a clock and checking each possible computation path of N on input x for acceptance. For simplicity we identify $time_N(x)$ with $f(x)$ and assume without loss of generality, that every computation of N on input x halts after exactly $f(x)$ steps.

The term g on the μ-average space-bounded is defined similarly for nondeterministic space-bounded computations.

Definition 4.1 (ANTIME(g), ANSPACE(g)) *Let g be a total function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$.*

- *The randomized decision problem (D, μ) is in* ANTIME(g) *if D is accepted by a nondeterministic Turing machine, which is g on μ-average time-bounded.*
- *The randomized decision problem (D, μ) is in the class* ANSPACE(g) *if D is accepted by a nondeterministic Turing machine, which is g on μ-average space-bounded.*

We define the following nondeterministic average case complexity classes.

$$\begin{aligned}
\text{ANP} &= \bigcup_{k>0} \text{ANTIME}(n^k m^k + k), \\
\text{ANE} &= \bigcup_{k>0} \text{ANTIME}(2^{k \cdot n \cdot m + k}), \\
\text{ANEXP} &= \bigcup_{k>0} \text{ANTIME}(2^{n^k m^k + k}) \text{ and} \\
\text{ANPSPACE} &= \bigcup_{k>0} \text{ANSPACE}(n^k m^k + k).
\end{aligned}$$

We note here that in contrast to worst case complexity the requirement that the time-bounds are exactly real time computable is restrictive. For example there exist problems (D, μ) which cannot be accepted by a nondeterministic polynomial on μ-average time-bounded Turing machine, but are accepted by a nondeterministic Turing machine, whose running time is polynomial on μ-average.

Denote with AverageNP the set of randomized decision problems (D, μ), where D is accepted by a nondeterministic Turing machine N whose running time $time_N(x)$ is polynomial on μ-average (but not necessarily polynomial time-bounded on μ-average as defined above) [WB93]. This class contains problems, which are not computable by any nondeterministic polynomial on average time-bounded machine.

Theorem 4.2 ANP $\subsetneq$ AverageNP.

Proof. Let D by a recursive language, which is accepted by the Turing machine M and is not in DTIME($2^{2^{n^2}}$). Define the probability function μ as

$$\mu(x) = \begin{cases} 1/(|x|^2 \cdot 2^{|x|} \cdot time_M(x)) & , \quad \in D \\ \\ 1/(|x|^3 \cdot 2^{|x|}) & , \quad \text{otherwise} \end{cases}$$

Note that M can be considered as nondeterministic Turing machine.

By Markov's inequality we get for an arbitrary $m > 0$:

$$\begin{aligned} &\mathrm{Prob}_\mu \left[time_M(x) > \frac{\pi^4}{90} \cdot |x| \cdot m \right] < \\ &< \quad \frac{90}{\pi^4 m} \cdot \mathrm{Exp}_\mu \left[\frac{time_M(x)}{|x|} \right] \quad = \quad \frac{90}{\pi^4 m} \cdot \sum_{x \in \Sigma^+} \mu(x) \cdot \frac{time_M(x)}{|x|} \\ &= \quad \frac{90}{\pi^4 m} \cdot \sum_{x \in \Sigma^+} \frac{1}{|x|^{4k} 2^{|x|}} \quad = \quad \frac{1}{m}. \end{aligned}$$

This shows $(D, \mu) \in$ AverageNP.

Now suppose, $(D, \mu) \in$ ANP, i.e. there exists a Turing machine N with $L(N) = D$ and a constant k, such that $time_N(x)$ is $n^k m^k + k$ on μ-average time-bounded. Since μ is positive, it holds for all $x \in \overline{D}$, that $time_N(x) \leq |x|^k \lceil 1/\mu(x) \rceil^k + k = |x|^4 2^{k|x|} + k$. Consider the following algorithm M' for deciding membership in D:

```
input x;
if (time_N(x) ≤ |x|^4 2^{k|x|} + k) then
    simulate N on input x in a deterministic manner;
    if N(x) accepts then accept
        else reject;
else accept;
```

The running time of M' consists of the deterministic simulation of N, so $time_{M'}(x)$ is bounded by $2^{2^{|x|^2}}$. This implies $D \in$ DTIME($2^{2^{n^2}}$). This is a contradiction. □

The same argument shows that AverageNP is not contained in ADTIME(g) for any fixed (recursive) function g.

4.2 Relations between average case measures

Next we state some relations between average case time and space complexity measures. It is not surprising, that these relations are the same as in worst case complexity.

Lemma 4.3 *Let g be a total function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$.*

(1) $\mathrm{ADTIME}(g) \subseteq \mathrm{ADSPACE}(g)$.

(2) $\mathrm{ANTIME}(g) \subseteq \mathrm{ANSPACE}(g)$.

(3) $\mathrm{ADTIME}(g) = co\text{-}\mathrm{ADTIME}(g)$.

(4) $\mathrm{ADSPACE}(g) = co\text{-}\mathrm{ADSPACE}(g)$.

(5) There exists a constant $c > 0$, such that $\mathrm{ANTIME}(g) \subseteq \mathrm{ADTIME}(c^g)$.

(6) There exists a constant $c > 0$, such that $\mathrm{ADSPACE}(g) \subseteq \mathrm{ADTIME}(c^g)$.

(7) $\mathrm{ANSPACE}(g) \subseteq \mathrm{ADSPACE}(g^2)$.

(8) If $g(n,m) \geq \log n + \log m$ then $\mathrm{ANSPACE}(g) = co\text{-}\mathrm{ANSPACE}(g)$.

Proof. The statements (1)–(4) follow directly from the above definitions of the average case complexity classes.

To proof (5), let (D,μ) be a randomized decision problem in $\mathrm{ANTIME}(g)$ and let N be a nondeterministic Turing machine such that $L(N) = D$ and

$$\mathrm{Prob}_\mu[time_N(x) > g(|x|,m)] < \frac{1}{m}$$

for every integer $m > 0$.

Consider a deterministic Turing machine M that determines if N accepts an input x by constructing a list of all configurations of N on input x that are accessible from the initial configuration. The number of these configurations is bounded by $d^{time_N(x)}$ for a constant d, which is independent from x. Recall that we assume that $time_N(x)$ is exactly real time computable. So this list can be computed within $c^{time_N(x)}$ steps where c is an appropriate constant. Therefore:

$$\mathrm{Prob}_\mu\left[time_M(x) > c^{g(|x|,m)}\right] \leq \mathrm{Prob}_\mu\left[c^{time_N(x)} > c^{g(|x|,m)}\right] < \frac{1}{m}.$$

This implies $(D,\mu) \in \mathrm{ADTIME}(c^g)$.

The proof of (6) is similar to (5). (7) can be obtained by porting Savitch's theorem to average case complexity theory. (8) can be obtained by porting the result of Immerman-Szelepcsényi, that $\mathrm{NSPACE}(g) = \mathrm{co\text{-}NSPACE}(g)$. □

Theorem 4.4

(1) $\mathrm{ANP} \subseteq \mathrm{AEXP}$.

(2) $\mathrm{APSPACE} = \mathrm{ANPSPACE} = co\text{-}\mathrm{ANPSPACE}$.

(3) $\mathrm{APSPACE} \subseteq \mathrm{AEXP}$.

Proof. By extensive usage of Lemma 4.3. □

5 Relations to worst case complexity

Now we give some (fundamental) relations between worst case and average case complexity classes. We compare deterministic and nondeterministic average case complexity classes with their worst case counterparts. We show that a collapse of a deterministic and a nondeterministic class in the average case is equivalent to a collapse in the worst case.

A relation between worst case complexity and average case complexity, that follows immediately from the definitions is the following proposition.

Proposition 5.1 *Let* $\mathcal{C} \in \{\mathrm{P}, \mathrm{E}, \mathrm{EXP}, \mathrm{NP}, \mathrm{NE}, \mathrm{NEXP}, \mathrm{PSPACE}\}$. *Then for all* $D \in \mathcal{C}$ *and for any computable* μ *it holds that* $(D, \mu) \in \mathrm{A}\mathcal{C}$.

It is shown in [WB93], that if P $\neq$ NP then AP $\neq$ ANP. Wang & Belanger use in their proof (polynomial) complexity cores, which exist for NP-sets, if P $\neq$ NP. We show the converse direction by using explicitly the exactly real time computability of nondeterministic time-bounds. Furthermore we extend these results to the exponential time average case classes.

In the proof we will use generalized complexity cores, which are studied in [BD87]. Let $\mathcal{C}$ be a class of languages and for any language A let $\mathcal{C}_A$ be the set $\{C \in \mathcal{C} \mid C \subseteq A\}$. A set H is a complexity core (or hard core) for A with respect to $\mathcal{C}$ if for every $C \in \mathcal{C}_A$ the intersection $C \cap H$ is a finite set. If $H \subseteq A$, then H is called a proper complexity core for A with respect to $\mathcal{C}$.

Theorem 5.2 (Theorem 2.10 in [BD87]) *Let* $\mathcal{C}$ *be a recursively enumerable class of recursive sets, that is closed under finite union and finite variation. Any infinite recursive set not in* $\mathcal{C}$ *has an infinite recursive proper complexity core with respect to* $\mathcal{C}$.

Since P fulfils the conditions of the Theorem, there exists —under the assumption P $\neq$ NP— a proper complexity core for SAT (or any NP-complete language) with respect to P. Assigning high probability to the elements of the core we can construct a randomized decision problem in ANP which is not in AP. The same argument holds for exponential and linear-exponential time.

Theorem 5.3

(1) P = NP *if and only if* AP = ANP.

(2) E = NE *if and only if* AE = ANE.

(3) EXP = NEXP *if and only if* AEXP = ANEXP.

Proof. We give a proof for exponential time (4). Suppose EXP $\neq$ NEXP and consider the Exponential Bounded Halting problem EBH, which is defined as

$$EBH = \{\langle N, x, t\rangle \mid \text{The Turing machine } N \text{ accepts } x \text{ in} \leq t \text{ steps}\}.$$

Note that N is a nondeterministic Turing machine and k is encoded in binary. Since EBH is NEXP-complete, it holds $EBH \in \mathrm{NEXP} - \mathrm{EXP}$. By Theorem 5.2 there exists a complexity core $H \subseteq EBH$ with respect to EXP. Consider the probability function

$$\mu(x) = \begin{cases} 1/(|x|^2||H^{|x|}||) & , \quad \text{if } x \in H \\ 1/(|x|^2 2^{|x|}) & , \quad \text{otherwise} \end{cases}$$

Since $EBH \in$ NEXP it holds $(EBH, \mu) \in$ ANEXP. Now suppose, that $(EBH, \mu) \in$ AEXP. Then there exists a deterministic Turing machine M with $L(M) = EBH$ and a constant $k > 0$, such that $time_M(x)$ is $2^{n^k m^k}$ on μ-average time-bounded. H is a complexity core for EBH with respect to EXP. So there exists a constant n_0, such that for all $x \in H$, $|x| \geq n_0$, it holds $time_M(x) > 2^{|x|^k |x|^{2k}}$. For $m = n_0^2$ we can estimate

$$\begin{aligned}
\mathrm{Prob}_\mu\left[time_M(x) > 2^{|x|^k m^k}\right] &\geq \mathrm{Prob}_\mu\left[x \in H \wedge time_M(x) > 2^{|x|^k m^k}\right] \\
&> \sum_{x \in \Sigma^*} \mu(x) \left[x \in H^{n_0} \wedge time_M(x) > 2^{|x|^k n_0^{2k}}\right] \\
&= \frac{||H^{n_0}||}{n_0^2 \cdot ||H^{n_0}||} = \frac{1}{m}.
\end{aligned}$$

This is a contradiction. Thus $(EBH, \mu) \notin$ AEXP.

For the other direction assume EXP = NEXP. Then $EBH \in$ EXP, i.e. there exists a constant k such that $EBH \in$ DTIME(2^{n^k}). We define the set

$$EBH' = \left\{ \langle N, x, t \rangle \mid \text{The Turing machine } N \text{ accepts } x \text{ in } \leq \sqrt[k]{t} \text{ steps} \right\},$$

which is in DTIME$(2^{c \cdot n})$ for a constant $c > 0$.

Let (D, μ) be a randomized decision problem in ANEXP. Let N be a nondeterministic Turing machine with $L(N) = D$ and $p_D : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ a polynomial such that for all $m > 0$

$$\mathrm{Prob}_\mu\left[time_N(x) > 2^{p_D(|x|,m)}\right] < \frac{1}{m}.$$

Consider the deterministic Turing machine M:

input x;
$t := time_N(x)^k$;
if $\langle N, x, t \rangle \in EBH'$ then accept
else reject;

Since $|\langle N, x, time_N(x)^k \rangle|$ is bounded by $4 \cdot (|N| + |x| + \log(time_N(x)^k))$ the running time of M can be computed as

$$\begin{aligned}
time_M(x) &= time_N(x)^k + time_{EBH'}(\langle N, x, time_N(x)^k \rangle) \\
&\leq time_N(x)^k + 2^{c \cdot |\langle N, x, time_N(x)^k \rangle|} \\
&\leq time_N(x)^k + 2^{c \cdot 4 \cdot (|N| + |x|)} \cdot time_N(x)^{4 \cdot k} \\
&\leq 2^{d \cdot |x|} \cdot time_N(x)^{4 \cdot k} \qquad \text{for } d \geq 5 \cdot c \cdot |N|.
\end{aligned}$$

Let $p(n, m) = d \cdot |x| + 4 \cdot k \cdot p_D(n, m)$. Then it holds for any $m > 0$:

$$\begin{aligned}
\mathrm{Prob}_\mu\left[time_M(x) > 2^{p(|x|,m)}\right] &\leq \mathrm{Prob}_\mu\left[2^{d \cdot |x|} \cdot time_N(x)^k > 2^{d \cdot |x| + k \cdot p_D(|x|,m)}\right] \\
&= \mathrm{Prob}_\mu\left[time_N(x)^{4 \cdot k} > \left(2^{p_D(|x|,m)}\right)^{4 \cdot k}\right] \\
&= \mathrm{Prob}_\mu\left[time_N(x) > 2^{p_D(|x|,m)}\right] < \frac{1}{m}.
\end{aligned}$$

Thus $(D, \mu) \in$ AEXP. □

The next result links the assumption that ANP is included in AE to worst case complexity.

Theorem 5.4 NP $\subseteq$ E *if and only if* ANP $\subseteq$ AE.

Proof. Similar to Theorem 5.3 □

6 Upward collapse properties and tally sets

A well known structural result in worst case complexity is the upward collapse property, that is, a collapse of some smaller deterministic and nondeterministic time-bounded classes implies a collapse of the respectively (exponentially) larger time-bounded classes. In this section we show that the upward collapse properties also hold in average case complexity. In particular the implication "If P = NP then E = NE (EXP = NEXP)" similarily holds in average case complexity.

Theorem 6.1

(1) If AP = ANP *then* AE = ANE.

(2) If AP = ANP *then* AEXP = ANEXP.

Proof. (1) We use Theorem 5.3: AP = ANP $\Rightarrow$ P = NP $\Rightarrow$ E = NE $\Rightarrow$ AE = ANE.
(2) analogue to (1). □

Another famous result is from Book. In [Boo74] he shows that E $\neq$ NE if and only if there exists a tally set in NP $-$ P. Here we give an average case analogue of this theorem. Interestingly this leads to weaker exponential time on average classes called APE in the deterministic and ANPE in the nondeterministic case. Recall that the time-bound is a function in the length of the input and the probability weight of the strings which exeed the time-bound. For example in the definition of AP, to achieve a probability weight smaller than $1/m$ the time-bound is a function polynomial in m and in the length n of the input. Similar, for AE the function is allowed to be exponential in m and n. In the case of APE (and ANPE) the time-bound is exponential in n but polynomial in m.

Definition 6.2 (APE, ANPE)

$$\begin{aligned} \text{APE} &= \bigcup_{k>0} \text{ADTIME}(2^{k\cdot n}m^k + k) \\ \text{ANPE} &= \bigcup_{k>0} \text{ANTIME}(2^{k\cdot n}m^k + k) \end{aligned}$$

Using the hierarchy theorem, we get immediately APE $\subsetneq$ AE.

Lemma 6.3 E = NE *if and only if* APE = ANPE.

Proof. Similar to Theorem 5.3. □

Corollary 6.4 AE = ANE *if and only if* APE = ANPE.

The tally encoding of the randomized decision problem (D, μ) is a randomized decision problem (D', μ') where $D' = \{0^x \mid x \in D\}$ and

$$\mu'(y) = \begin{cases} \mu(x) & , \text{ if } y = 0^x \text{ for some } x \in \Sigma^*, \\ 0 & , \text{ otherwise} \end{cases}$$

This definition is based on the assumption that the tally encoding 0^x of an instance x occurs with the same probability as x itself. So we can use transformations between tally and binary encoding without changing the underlying probability distribution. We denote the tally encoding of (D, μ) with $tally(D, \mu)$.

We say, (D, μ) is a randomized tally decision problem, if $D \subseteq \{0\}^*$ and $\mu(x) = 0$ if $x \notin \{0\}^*$.

Lemma 6.5 *Let (D, μ) be any randomized decision problem. Then:*

(1) $(D, \mu) \in$ APE if and only if $tally(D, \mu) \in$ AP.

(2) $(D, \mu) \in$ ANPE if and only if $tally(D, \mu) \in$ ANP.

Proof. We give a proof for the deterministic case. The proof for nondeterministic computations is similar. Let $(D, \mu) \in$ APE. Then there exists a Turing machine M with $L(M) = D$ and a constant k such that for all $m > 0$

$$\text{Prob}_\mu\Big[time_M(x) > 2^{k \cdot |x|} m^k + k\Big] < \frac{1}{m}.$$

To show that $(D', \mu') = tally(D, \mu) \in$ AP, consider the following machine M', that accepts D':

input 0^x;
simulate M on input x;
if $M(x)$ *accepts* then accept;
else reject;

The time of M' is used for the simulation of M, so $time_{M'}(0^x) = time_M(x)$. For every integer $m > 0$ it follows

$$\begin{aligned} \text{Prob}_{\mu'}\Big[x = 0^y \wedge time_{M'}(x) > |x|^k m^k + k\Big] &= \text{Prob}_{\mu'}\Big[x = 0^y \wedge time_M(y) > |x|^k m^k + k\Big] \\ &= \text{Prob}_{\mu'}\Big[x = 0^y \wedge time_M(y) > 2^{k \cdot |y|} m^k + k\Big] \\ &\leq \text{Prob}_\mu\Big[time_M(x) > 2^{k \cdot |x|} m^k + k\Big] < \frac{1}{m}. \end{aligned}$$

For the converse let (D, μ) be a randomized decision problem such that $(D', \mu') = tally(D, \mu)$ is in AP. Then by definition there exists a Turing machine M' with $L(M') = D'$ and a constant $k > 0$ such that $n^k m^k + k$ guarantees the polynomial on μ-average time-bound of M', i.e. for all $m > 0$

$$\text{Prob}_\mu\Big[time_{M'}(x) > |x|^k m^k + k\Big] < \frac{1}{m}.$$

Consider the following machine M:

input x;
$y := \mathtt{0}^x$;
simulate M' *on input* y;
if $M'(y)$ *accepts* then accept;
else reject;

The machine M accepts D by simulating M' and the running time of M is $time_M(x) = O(|\mathtt{1}^x|) + time_{M'}(\mathtt{1}^x)$.

$$\begin{aligned}
&\mathrm{Prob}_\mu\Big[time_M(x) > 2^{k\cdot|x|}m^k + k\Big] \\
&= \mathrm{Prob}_\mu\Big[y = \mathtt{0}^x \wedge time_M(x) > |y|^k m^k + k\Big] \\
&= \mathrm{Prob}_\mu\Big[y = \mathtt{0}^x \wedge time_{M'}(y) > |y|^k m^k + k\Big] \\
&= \mathrm{Prob}_{\mu'}\Big[time_{M'}(x) > |x|^k m^k + k\Big] < \frac{1}{m}.
\end{aligned}$$

This implies $(D, \mu) \in \mathrm{APE}$. □

Theorem 6.6 AE $\neq$ ANE *if and only if there exists a randomized tally decision problem in* ANP $-$ AP.

Proof. The Theorem follows from Corollary 6.4 and Lemma 6.5. □

7 Conclusions and Open Problems

We have proposed a definition of "g on average" based on Schapire's characterization of Levin's notion of "polynomial on average". This definition —while preserving the notion of "polynomial on average"— allows to precisely measure the average case complexity of functions (or resource bounds of Turing machines resp.).

The notion of exactly real time computable functions has been used to define nondeterministic average case time- and space-bounded complexity classes. We have shown that the requirement, that the time-/space-bound of a nondeterministic Turing machine is exactly real time computable, is sufficient to get a structure in average case complexity similar to worst case complexity. Without any requirement on the computability of the time-bounds, nondeterministic computations that are time-bounded on average are not contained in any fixed (worst case) time-bounded class.

A crucial question in averge case complexity is the complexity of the underlying distribution. It is known that there exist certain (non-computable, universal) distributions such that for every algorithm its worst case complexity is equal to its average case complexity. In particular there are exponential time computable distributions such that any problem which is in AP under this distribution is already in P. Thus it seems reasonable to consider only distributions that are easy to compute (i.e. ptime-computable), or easy to generate (i.e. polynomial-samplable).

Some of the results (like the hierarchy theorems) hold for easy distributions. However for the collapse results (Theorem 5.3), the distributions "need enough power" to compute the instances of the complexity core. Thus it is not known whether "AP = ANP if and only if P = NP" still holds if we restrict ourselves on less powerful (i.e. polynomial time computable) distributions.

A further goal would be to close the gap between the average case complexity (as considered here) and the traditional (lengthwise) average case complexity. More precisely let $\mu_1, \mu_2, \ldots$ be a sequence of probability distributions such that $\mu_n(\Sigma^n) = 1$ and $\mu_n = 0$ if $|x| \neq n$. Now assume that for a function f and all n

$$\sum_{x \in \Sigma^n} f(x) \cdot \mu_n(x) = g(x).$$

This should be equivalent to the statement that f is $g(n, m) = g(n) \cdot g(m)$ on μ-average for some appropriate distribution μ such as $\mu(x) = n^{-2} \cdot \mu_n(x)$, where $n = |x|$.

References

[BD87] R.V. Book and D. Du. The existence and density of generalized complexity cores. *Journal of the Association for Computing Machinery*, 34(3):718–730, 1987.

[BDCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992.

[BDG88] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer–Verlag, 1988.

[Boo74] R.V. Book. Tally languages and complexity classes. *Information and Control*, 26:281–287, 1974.

[Coo73] S.A. Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7:343–353, 1973.

[CS95] J. Cai and A. Selman. Average time complexity classes. Technical Report TR95-019, ECCC Trier, 1995.

[GGH94] M. Goldmann, P. Grape, and J. Håstad. On average time hierarchies. *Information Processing Letters*, 49:15–20, 1994.

[Gol88] O. Goldreich. Towards a theory of average case complexity. Technical report, Technion Haifa, Israel, 1988.

[Gur91] Y. Gurevich. Average case complexity. *Journal of Computer and System Sciences*, 42(3):346–398, 1991.

[HS66] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape turing machines. *Journal of the Association for Computing Machinery*, 13(4):533–546, 1966.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley, 1979.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proc. 10th Conference on Structure in Complexity Theory*, pages 134–147, 1995.

[Kar94] C. Karg. Strukturfragen im Umfeld der Durchschnittskomplexität. Master's thesis, Universität Ulm, 1994.

[KF82] K.I. Ko and H. Friedman. Computational complexity of real functions. *Theoretical Computer Science*, 20:323–352, 1982.

[Lev84] L. Levin. Problems, complete in "average" instance. In *Proc. 16th ACM Symposium on the Theory of Computing*, page 465, 1984.

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

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[RS93] R. Reischuk and Ch. Schindelhauer. Precise average case complexity. In *Proc. 10th Annual Symposium on Theoretical Aspects of Computer Science*, 1993.

[Sch90] R.E. Schapire. The emerging theory of average case complexity. Technical Report 431, MIT, 1990.

[Sch95] R. Schuler. Some properties of sets tractable under every polynomial-time computable distribution. *Information Processing Letters*, 1995.

[SW95] R. Schuler and O. Watanabe. Towards average-case complexity analysis of NP optimization problems. In *Proc. 10th Conference on Structure in Complexity Theory*, pages 148–159, 1995.

[SY92] R. Schuler and T. Yamakami. Structural average case complexity. In *Proc. 12th Foundations of Software Technology and Theoretical Computer Science*, pages 128–139, 1992.

[SY95] R. Schuler and T. Yamakami. Sets computable in polynomial time on average. In *Proc. 1st International Computing and Combinatorics Conference*, 1995.

[WB93] J. Wang and J. Belanger. On average P vs. average NP. In K. Ambos-Spies, S. Homer, and U. Schöning, editors, *Complexity Theory—Current Research*. Cambridge University Press, 1993.