Average-Case Intractability vs. Worst-Case Intractability

Johannes Köbler and Rainer Schuler

Abteilung Theoretische Informatik
Universität Ulm
89069 Ulm, Germany
{koebler,schuler}@informatik.uni-ulm.de

Abstract

We use the assumption that all sets in NP (or other levels of the polynomial-time hierarchy) have efficient average-case algorithms to derive collapse consequences for $\mathcal{MA}$, $\mathcal{AM}$, and various subclasses of $\mathcal{P}$/poly. As a further consequence we show for $\mathcal{C} \in \{\mathcal{P}(\mathcal{PP}), \mathcal{PSPACE}\}$ that $\mathcal{C}$ is not tractable in the average-case unless $\mathcal{C} = \mathcal{P}$.

1 Introduction

In general, the average-case complexity of an algorithm depends (by definition) on the distribution on the inputs. In fact, there exist certain (so called malign or universal) distributions relative to which the average-case complexity of any algorithm coincides with its worst-case complexity [LV92]. Fortunately, these distributions are not recursive. Even for the class of polynomial-time bounded algorithms, malign distributions are not computable in polynomial time [Mil93].

In recent literature, it has been shown that several $\mathcal{NP}$-complete problems are solvable efficiently on average (i.e., in time polynomial on μ-average) with respect to certain natural distributions μ on the instances. However, this is not true for all $\mathcal{NP}$-complete problems, unless $\mathcal{E} = \mathcal{NE}$ [BCGL92]. In fact, some natural $\mathcal{NP}$ problems A are under a particular distribution μ complete for $\mathcal{NP}$ in the sense that A is not efficiently solvable on μ-average unless any $\mathcal{NP}$ problem is efficiently solvable with respect to any polynomial-time computable distribution [Lev86]. It is therefore one of the main open problems in average-case complexity theory whether $\mathcal{NP}$ problems can be solved efficiently on average with respect to natural, i.e., polynomial-time computable distributions.

Let $\mathcal{AP}_{\mathcal{FP}}$ denote the class of sets that are decidable in time polynomial on μ-average with respect to every polynomial-time computable distribution. As noted above, $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ implies that $\mathcal{E} = \mathcal{NE}$ [BCGL92]. This result provides an interesting connection between average-case complexity and worst-case complexity. Namely, if all $\mathcal{NP}$ problems can be decided in time polynomial on average, then all sets in $\mathcal{NE}$ can be decided in (worst-case) exponential time.

Similarly, as observed in [FF93], any random self-reducible set which can be decided in time polynomial on average (under the distribution induced by the random self-reduction) can be decided by a randomized algorithm in (worst-case) polynomial time. For example, Lipton [Lip91] used an idea of Beaver and Feigenbaum [BF90] to show that multivariate polynomials of low degree are (functionally) random self-reducible. In particular, it follows from Lipton's result that if there is an algorithm computing the permanent efficiently for all but a sufficiently small (polynomial) fraction of all $n \times n$ matrices (over GF(p) where $p > n + 1$ is prime), then it is possible to compute the permanent of any $n \times n$ matrix in expected polynomial time. Using this property it is not hard to show that $\mathcal{P}(\mathcal{PP}) \not\subseteq \mathcal{AP}_{\mathcal{FP}}$ unless $\mathcal{PP} = \mathcal{ZPP}$. From Corollary 3.3 below, $\mathcal{P}(\mathcal{PP}) \subseteq \mathcal{AP}_{\mathcal{FP}}$ even implies that $\mathcal{PP} = \mathcal{P}$ (in fact, it is easy to verify that $\mathcal{PP} = \mathcal{P}$ already follows from the assumption that the middle bit class $\mathcal{MP}$ [GKR$^+$95] is contained in $\mathcal{AP}_{\mathcal{FP}}$). This means that for $\mathcal{C} = \mathcal{P}(\mathcal{PP})$, $\mathcal{C}$ is not tractable on the average unless $\mathcal{C}$ is tractable in the worst-case. This rises the question whether a similar relationship holds for other classes $\mathcal{C}$ as, e.g., $\mathcal{C} = \mathcal{NP}$ or, more generally, for $\mathcal{C} = \Sigma_k^p$.

In contrast to worst-case complexity, where $\mathcal{NP} \subseteq \mathcal{P}$ implies that $\mathcal{PH} \subseteq \mathcal{P}$, it is not known whether $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ implies that all sets in $\Delta_2^p = \mathcal{P}(\mathcal{NP})$ are contained in $\mathcal{AP}_{\mathcal{FP}}$ (see [Imp95] for an exposition). Consider for example an $\mathcal{NP}$ optimization problem. It is not known whether an efficient average-case algorithm for the corresponding decision problem can be used to compute efficiently on average an optimal solution. To see the difficulty consider the computation of a deterministic Turing machine M with oracle A, where the distribution on the inputs of M is computable in polynomial time. Since the oracle queries can be adaptive, it depends on the oracle set A which queries are actually made. Hence, the distribution induced on the oracle queries is not necessarily computable in polynomial time. On the other hand, it is known that $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ implies $\Theta_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ (cf. Theorem 3.9). We refer the reader to [Imp95, SW95] for further discussions of this and related questions. As shown in [Sch96], the class $\mathcal{AP}_{\mathcal{FP}}$ is not closed under Turing reducibility, moreover, $\mathcal{AP}_{\mathcal{FP}}$ even contains Turing complete sets for $\mathcal{EXP}$ (note that $\mathcal{EXP}$ is not contained in $\mathcal{AP}_{\mathcal{FP}}$).

Our results are based on the following special properties of any set $A \in \mathcal{AP}_{\mathcal{FP}}$: Firstly, for any $\mathcal{P}$-printable domain D there is an algorithm that decides A efficiently on all inputs in the domain D. Secondly, since A is efficiently decidable on average with respect to the standard distribution μ_{st} (which is uniform on Σ^n), there is an algorithm for A that is polynomial in the worst case for all but a polynomial fraction of the strings of each length. Roughly speaking, we exploit these two properties in the following context: A serves as an oracle in a computation that generates oracle queries in such a way that it is sufficient to answer these queries either on some $\mathcal{P}$-printable domain or on any domain which contains a large fraction of the strings of each length.

In particular, we get the following collapse consequences. (The notion of instance complexity and the class IC[log,poly] of sets of strings with low instance complexity were introduced in [OKSW94]. As shown in [OKSW94], $\mathcal{P}/\log \subsetneq \mathrm{IC}[\log, \mathrm{poly}] \subsetneq \mathcal{P}/\mathrm{poly}$, and in

[AHH$^+$93] the following characterization of IC[log,poly] is given: A set A is in IC[log, poly] if and only if A and its complement are both conjunctively reducible to a tally set.)

- If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\mathcal{NP} \cap \mathcal{P}/\log = \mathcal{P}$.
- If $\Delta_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\Delta_2^p \cap \mathrm{IC}[\log, \mathrm{poly}] = \mathcal{P}$ and every self-reducible set in $\mathcal{P}/\mathrm{poly}$ is in $\mathcal{ZPP}$.
- If $\Sigma_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then all sets in $\Sigma_2^p \cap \Pi_2^p$ that conjunctively, disjunctively, or bounded truth-table reduce to some sparse set are in $\mathcal{P}$.
- If $\Delta_3^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\Sigma_2^p \cap \Pi_2^p \cap \mathcal{P}/\mathrm{poly} = \mathcal{P}$.
- If $\Sigma_3^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\Sigma_3^p \cap \Pi_3^p \cap \mathcal{P}/\mathrm{poly} = \mathcal{P}$.

Since $\mathcal{BPP}$ is contained in $\Sigma_2^p \cap \Pi_2^p$ [Sip83, Lau83] and in $\mathcal{P}/\mathrm{poly}$ [BG81] we get in particular:

- If $\Delta_3^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\mathcal{BPP} = \mathcal{P}$.

It is interesting to note that Corollary 4.5 implies stronger collapse consequences for the polynomial hierarchy. For example, if $\Delta_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\mathcal{NP} \subseteq \mathcal{P}/\mathrm{poly}$ implies $\mathcal{PH} = \mathcal{ZPP}$.

We also investigate the question whether problems in $\mathcal{NP}$ (or in other levels of PH) are solvable in time polynomial on average with respect to every distribution computable in $\mathcal{FP}(\Sigma_k^p)$ (in symbols: $\mathcal{NP} \subseteq ? \mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$). Note that $\mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$ is a (possibly strict) subclass of $\mathcal{AP}_{\mathcal{FP}}$. Hence, $\Sigma_k^p \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$ is a (possibly) stronger assumption than $\Sigma_k^p \subseteq \mathcal{AP}_{\mathcal{FP}}$.

Under the assumption that $\mathcal{NP}$ problems are solvable in time polynomial on average with respect to distributions in $\mathcal{FP}(\Sigma_2^p)$ we show that IP[$\mathcal{P}$/poly] is contained in $\mathcal{NP} \cap$ co-$\mathcal{NP}$, where IP[$\mathcal{P}$/poly] is the class of all sets that have an interactive proof with prover complexity restricted to $\mathcal{P}$/poly [BFL91, AKS95].

- If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_2^p)}$ then IP[$\mathcal{P}$/poly] $\subseteq \mathcal{NP} \cap$ co-$\mathcal{NP}$.

Since, as we show, $\Sigma_k^p \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$ implies $\Delta_{k+1}^p \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$ and since $\Delta_k^p \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$ implies $\Delta_k^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ we get the following corollaries:

- If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{NP})}$ then $\Delta_2^p \cap \mathrm{IC}[\log, \mathrm{poly}] = \mathcal{P}$ and every self-reducible set in $\mathcal{P}$/poly is in $\mathcal{ZPP}$.
- If $\Sigma_2^p \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_2^p)}$ then $\Sigma_2^p \cap \Pi_2^p \cap \mathcal{P}/\mathrm{poly} = \mathcal{P}$ and $\mathcal{BPP} = \mathcal{P}$.

Finally, we extend a result in [Imp95] showing that $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ implies $\mathcal{BPP} = \mathcal{ZPP}$ as follows:

- If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$, then $\mathcal{MA} = \mathcal{NP}$.
- If $\Sigma_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$, then $\mathcal{AM} = \mathcal{NP}$.

Note that $\mathcal{AM} = \mathcal{NP}$ has some immediate strong implications as, for example, Graph Isomorphism is in $\mathcal{NP} \cap \text{co-}\mathcal{NP}$.

Recently a series of plausible consequences, not known to follow from the assumption $\mathcal{P} \neq \mathcal{NP}$, have been derived from the assumption that $\mathcal{NP}$ is not small in $\mathcal{EXP}$, see, e.g., [LM96, Lut97a, Lut97b, AK97]. It is interesting to note that the assumption $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ is contradictory to Lutz' hypothesis that $\mathcal{NP}$ is not small in $\mathcal{EXP}$, as follows directly from the fact that $\mathcal{AP}_{\mathcal{FP}}$ is small in $\mathcal{EXP}$ [SY95, CS96].

In this extended abstract most proofs are omitted; see [KS97] for a complete version.

2 Preliminaries

All languages are over the binary alphabet $\Sigma = \{0, 1\}$. The *length* of a string $x \in \Sigma^*$ is denoted by $|x|$. For a language A, let $A^{=n}$ denote the set of all strings in A of length n. A string is called *tally* if it is an element of 1^* and a set T is *tally* if $T \subseteq 1^*$. A set S is called *sparse* if the cardinality of $S^{=n}$ is bounded above by a polynomial in n. TALLY denotes the class of all tally sets, and SPARSE denotes the class of all sparse sets. The cardinality of a finite set A is denoted by $\|A\|$.

The *join* of two sets A and B is $A \oplus B = \{0x \mid x \in A\} \cup \{1x \mid x \in B\}$. The join of language classes is defined analogously. To encode pairs (or tuples) of strings we use a standard polynomial-time computable pairing function denoted by $\langle \cdot, \cdot \rangle$ whose inverses are also computable in polynomial time. We assume that this function encodes tuples of tally strings again as a tally string. $\mathbb{N}$ denotes the set of non-negative integers and by log we denote the function $\log x = \max\{1, \lceil \log_2 x \rceil\}$.

We assume that the reader is familiar with fundamental complexity theoretic concepts such as (oracle) Turing machines and the polynomial-time hierarchy (see, for example, [BDG95, Sch86]).

Let $\mathcal{C}$ be a complexity class. A set A is $\mathcal{P}^{\mathcal{C}}$-printable if there exists a set $C \in \mathcal{C}$ and a polynomial time bounded oracle Turing transducer T such that the output of T with oracle C and input 1^n is an enumeration of all strings in A of length n. An oracle Turing machine T is non-adaptive, if for all oracles C and all inputs x, the queries of T on input x are independent of C. T is honest if there exists a constant c such that $|x| \leq |y|^c$ for all x and for all oracle queries y of T on input x. A set A is $\mathcal{P}_{\text{,honest}}(\mathcal{C})$-printable if A is $\mathcal{P}(\mathcal{C})$-printable and the respective Turing transducer is honest and non-adaptive.

Next we review the notion of advice functions introduced by Karp and Lipton [KL80] to characterize non-uniform complexity classes. A function $h : 0^* \to \Sigma^*$ is called a *polynomial-length function* if for some polynomial p and for all $n \geq 0$, $|h(0^n)| = p(n)$. For a class $\mathcal{C}$ of sets, let $\mathcal{C}$/poly be the class of sets L such that there is a set $I \in \mathcal{C}$ and a polynomial-length function h such that for all n,

$$\forall x \in \Sigma^n : x \in L \;\Leftrightarrow\; \langle x, h(0^n) \rangle \in I.$$

The function h is called an *advice function* for L, whereas I is the corresponding *interpreter set.*

In the following we will also make use of multi-valued advice functions. A (total) multi-valued function h maps every string x to a non-empty subset of Σ^*, denoted by *set-*$h(x)$. We say that g is a refinement of h if for all x, *set-*$g(x) \subseteq$ *set-*$h(x)$.

A *multi-valued* advice function h has the property that for some polynomial p and all n, *set-*$h(0^n) \subseteq \Sigma^{p(n)}$ and for all $w \in$ *set-*$h(0^n)$,

$$\forall x \in \Sigma^n : x \in L \Leftrightarrow \langle x, w \rangle \in I.$$

Let $\mathcal{F}$ be a class of (possibly multi-valued) functions and let $L \in \mathcal{C}/\text{poly}$. Then L is said *to have an advice function in* $\mathcal{F}$ (with respect to interpreter class $\mathcal{C}$) if some $h \in \mathcal{F}$ is an advice function for L with respect to some interpreter set $I \in \mathcal{C}$.

Let μ be a probability distribution on Σ^*. Associated with μ are a distribution function that we also denote by μ and a density function, denoted by μ'. μ and μ' are functions from Σ^* to the interval $[0,1]$ such that $\sum_x \mu'(x) = 1$ and $\mu(x) = \sum_{y \leq x} \mu'(y)$ where, as usual, $\leq$ denotes the lexicographic ordering on Σ^*. Let t be a function from $\mathbb{N}$ to $\mathbb{N}$. A distribution μ t-dominates a distribution ν, if $\mu'(x) \cdot t(|x|) \geq \nu'(x)$ for all x. If t is a constant, then we say that μ dominates ν by a constant factor, similarly, if t is bounded by a polynomial, then we say that μ polynomially dominates ν.

Let μ be a distribution. A function $f : \Sigma^* \to \mathbb{N}$ is polynomial on μ-average [Lev86], if there exists a constant $\epsilon > 0$ such that

$$\sum_{x \neq \lambda} \frac{f^\epsilon(x)}{|x|} \mu'(x) < \infty.$$

The class of functions polynomial on μ-average has many closure properties that are known for polynomials [Lev86, Gur91]. A further important property is robustness under the polynomial domination of distributions [Lev86, Gur91], i.e., any function that is polynomial on ν-average is also polynomial on μ-average provided that ν dominates μ.

In the recent literature on average-case complexity basically two ways have been considered to formalize the intuitive notion of feasible (or natural) distributions. The more restrictive way is to consider only distributions as feasible that have *efficiently computable* distribution functions [Lev86, Gur91]. On the other hand, any efficiently samplable distribution (according to which instances can be *efficiently generated*) can be considered feasible. As shown in [BCGL92], every efficiently computable distribution is dominated by an efficiently samplable distribution. This implies that if a problem is solvable in time polynomial on average with respect to any efficiently samplable distribution then it is also solvable in time polynomial on average with respect to any efficiently computable distribution.

A distribution is said to be $\mathcal{P}$-computable if its distribution function μ is $\mathcal{P}$-computable, i.e., there exists a polynomial time bounded deterministic Turing transducer M such that for all x and all k it holds that $|M(x, 1^k) - \mu(x)| \leq 2^{-k}$. Here the output of M is interpreted as a rational number, in some appropriate way. For example, if $M(x, 1^k) = \langle p, q \rangle$, then $M(x, 1^k)$ computes the number p/q. As the following remark shows, requiring that the density function μ' of a distribution is $\mathcal{P}$-computable is a strictly weaker condition unless $\mathcal{P} \neq \mathcal{NP}$.

Remark. *As shown in [Gur91], if $\mathcal{P} \neq \mathcal{NP}$ then there exists a distribution whose density function μ' is $\mathcal{P}$-computable but whose distribution function μ is not $\mathcal{P}$-computable.*

As usual let $\mathcal{FP}$ denote the set of polynomial-time computable functions. An important subclass of the class of $\mathcal{P}$-computable distributions is the class of so-called $\mathcal{FP}$-computable distributions for which μ can be efficiently computed without error. For a complexity class $\mathcal{C}$, we say that a distribution is $\mathcal{FP}(\mathcal{C})$-computable (in symbols: $\mu \in \mathcal{FP}(\mathcal{C})$) if its distribution function μ is $\mathcal{FP}(\mathcal{C})$-computable, i.e., there exist functions $f \in \mathcal{FP}(\mathcal{C})$ and $g \in \mathcal{FP}$ such that for all x, $\mu(x) = f(x)/g(x)$.

As the following theorem shows, a problem is solvable in time polynomial on ν-average for every $\mathcal{FP}$-computable distribution ν if and only if it is solvable in time polynomial on μ-average for every $\mathcal{P}$-computable distribution μ.

Theorem 2.1 *[Gur91] Every $\mathcal{P}$-computable distribution μ is dominated by a $\mathcal{FP}$-computable distribution ν by a constant factor. Furthermore, for all x, the binary representation of $\nu(x)$ is of length linear in the length of x.*

Following [Lev86, Gur91] we assume that all natural distributions are either $\mathcal{P}$-computable or dominated by a $\mathcal{P}$-computable distribution. In this sense, a set is efficiently decidable on average (under natural distributions) if it is decidable in time polynomial on μ-average with respect to every distribution $\mu \in \mathcal{FP}$.

Definition 2.2 *[SY96] Let $\mathcal{F}$ be a set of distributions. A set A is decidable in average polynomial time under distributions in $\mathcal{F}$ (in symbols, $A \in \mathcal{AP}_{\mathcal{F}}$) if for every distribution $\mu \in \mathcal{F}$ there exists a deterministic Turing machine M such that $A = L(M)$ and the running time of M is polynomial on μ-average.*

As noted by Ben-David et al., all sets in $\mathcal{AP}_{\mathcal{FP}}$ are decidable in polynomial time on tally inputs.

Theorem 2.3 *[BCGL92] Every tally set in $\mathcal{AP}_{\mathcal{FP}}$ is in $\mathcal{P}$.*

Proof. Assume that A is a tally set and in $\mathcal{AP}_{\mathcal{FP}}$. Consider a tally distribution μ_{tally} defined as follows:

$$\mu'_{tally}(x) = \begin{cases} 1/n(n+1), & \text{if } x = 0^n, n > 0; \\ 0, & \text{otherwise.} \end{cases}$$

It is not difficult to see that the distribution function μ_{tally} is $\mathcal{FP}$-computable (i.e., $\mu_{tally}(x) = f(x)/g(x)$ for some $\mathcal{FP}$ functions $f, g : \Sigma^* \to \mathbb{N}$).

Since A is in $\mathcal{AP}_{\mathcal{FP}}$ there exists a Turing machine M such that $A = L(M)$ and the running time of M is polynomial on μ_{tally}-average. This implies, according to part one of Proposition 2.6, that M is polynomial time bounded (in worst-case) on all tally strings. ■

In [Sch90], Schapire gives the following characterization of a function f being polynomial on μ-average.

Theorem 2.4 *[Sch90] Let μ be a distribution. Then $f : \Sigma^* \to \mathbb{N}$ is polynomial on μ-average if and only if there exists a polynomial p such that for all m,*

$$\mu\{x \mid f(x) > p(|x|, m)\} \;\leq\; \frac{1}{m}.$$

From this characterization it follows immediately that any function f that is polynomial on μ-average is in fact polynomially bounded on $\Sigma^{\leq n}$, except for a subset which has low probability under μ.

Proposition 2.5 *Let f be polynomial on μ-average. For every polynomial p there exists a polynomial p' such that for all n,*

$$\mu\{x \in \Sigma^{\leq p(n)} \mid f(x) > p'(n)\} \;\leq\; \frac{1}{p(n)}.$$

Proof. Assume that f is polynomial on μ-average. Choose constants c and $k > 1$ such that $\sum_{x \neq \lambda} \frac{f^{1/k}(x)}{|x|}\mu'(x) < c$. Now let H_n denote the set $\{x \in \Sigma^{\leq p(n)} \mid f(x) > c^k p^{2k}(n)\}$ and assume to the contrary that for some $n > 0$, $\mu'(H_n) > 1/p(n)$. Then,

$$\sum_{x \neq \lambda} \frac{f^{1/k}(x)}{|x|}\mu'(x) \;\geq\; \sum_{x \in H_n} \frac{cp^2(n)}{p(n)}\mu'(x) \;\geq\; \frac{1}{p(n)} \cdot c \cdot p(n) \;=\; c.$$

■

The above proposition gives us the following special cases which we will need in the following.

Proposition 2.6 *Let f be polynomial on μ-average.*

1. *Then for every polynomial p there exists a polynomial p' such that $f(x) \leq p'(|x|)$ holds for all x with $\mu'(x) \geq 1/p(|x|)$.*
2. *If $\mu = \mu_{st}$ is the standard distribution (where $\mu'_{st}(x) = \frac{1}{|x|(|x|+1)} \cdot 2^{-|x|}$ for all $x \neq \lambda$), then for every polynomial p there exists a polynomial p' such that for all $n > 0$,*

$$\|\{x \in \Sigma^n \mid f(x) > p'(n)\}\| \leq \frac{2^n}{p(n)}.$$

3 Eliminating tally and printable oracle queries

A first consequence of the assumption that $\mathcal{NP}$ problems are decidable in time polynomial on μ-average for any distribution $\mu \in \mathcal{FP}$ was given by Ben-David et.al. [BCGL92].

Theorem 3.1 *[BCGL92] If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$, then $\mathcal{E} = \mathcal{NE}$ (or, equivalently, $\mathcal{NP} \cap \mathrm{TALLY} \subseteq \mathcal{P}$).*

Proof. Recall that $\mathcal{E} = \mathcal{NE}$ if and only if every tally set in $\mathcal{NP}$ is in $\mathcal{P}$ [Boo74]. Since, by Theorem 2.3, every tally set in $\mathcal{AP}_{\mathcal{FP}}$ is already in $\mathcal{P}$ it follows that $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ implies $\mathcal{E} = \mathcal{NE}$. ■

Put in other words, if $\mathcal{NP}$ problems have efficient average-case decision algorithms, then $\mathcal{P}(\mathcal{NP} \cap \mathrm{TALLY})$, a subclass of $\mathcal{P}/\mathrm{poly}$, collapses downto $\mathcal{P}$. We observe that similar collapse consequences downto $\mathcal{P}$ can be derived for other subclasses of $\mathcal{P}/\mathrm{poly}$ (see Corollary 3.3). Some of these collapse consequences follow immediately from recent results investigating the complexity of sparse and tally descriptions for sets in $\mathcal{P}/\mathrm{poly}$ [BS92, Köb94, Gav95, AKM96]. For the others we can exploit an interesting connection between the worst-case complexity of a set L and the average-case complexity of oracles used in the computation of an advice function for L.

The following theorem shows that if an advice function h for some set L can be efficiently computed relative to some oracle which is efficiently decidable on average, then h is computable in polynomial time.

Theorem 3.2

1. *Any advice function that is computable in $\mathcal{FP}(D)$ where $\mathcal{P}(D) \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ is computable in $\mathcal{FP}$.*

2. *Any advice function that is computable in $\mathcal{FP}(D)$ where $\{A \mid A \leq_m^p D\} \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ is computable in $\mathcal{FP}$.*

Proof.

1. Let h be an advice function in $\mathcal{FP}(D)$ where D is an oracle for which $\mathcal{P}(D) \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$. Then h is computable in $\mathcal{FP}(T)$ where T is the tally set

   $$T = \{\langle 0^n, 0^i \rangle \mid \text{ the } i\text{th bit of } h(0^n) \text{ is one}\}.$$

   Since T is in $\mathcal{P}(D) \cap \mathrm{TALLY}$ and hence by assumption in $\mathcal{AP}_{\mathcal{FP}}$, it follows from Theorem 2.3 that $T \in \mathcal{P}$, implying that $h \in \mathcal{FP}$.

2. Assume that h is an advice function in $h \in \mathcal{FP}(D)$ via some oracle transducer M. Then h is computable in $\mathcal{FP}(T)$ where T is the tally set

   $$T = \{\langle 0^n, 0^i \rangle \mid \text{ the } i\text{th query of } M(0^n) \text{ is in } D\}.$$

   Since T many-one reduces to D it follows from the assumption and from Theorem 2.3 that $T \in \mathcal{P}$, implying that $h \in \mathcal{FP}$.

■

Now, using results from [BS92, AKM96, Köb94, Gav95], we can state similar collapse consequences as in Theorem 3.1 for several subclasses of $\mathcal{P}$/poly. We note that by using a different proof technique it has been shown in [BFNW93] that $\mathcal{BPP} = \mathcal{P}$ follows from the assumption that every tally set in Σ_4^p is contained in $\mathcal{P}$.

Corollary 3.3

1. *If $\mathcal{NP} \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\mathcal{NP} \cap \mathcal{P}/log = \mathcal{P}$.*

2. *If $\Delta_2^p \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\Delta_2^p \cap \mathrm{IC}[\log, \mathrm{poly}] = \mathcal{P}$.*

3. *If $\Sigma_2^p \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ then all sets in $\Sigma_2^p \cap \Pi_2^p$ that conjunctively, disjunctively, or bounded truth-table reduce to some sparse set are in $\mathcal{P}$.*

4. *If $\Delta_3^p \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\Sigma_2^p \cap \Pi_2^p \cap \mathcal{P}/poly = \mathcal{P}$ and hence $\mathcal{BPP} = \mathcal{P}$.*

5. *If $\Sigma_3^p \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\Sigma_3^p \cap \Pi_3^p \cap \mathcal{P}/poly = \mathcal{P}$.*

Proof.

1. As shown in [BS92], every set A in $\mathcal{NP} \cap \mathcal{P}/\log$ is contained in $\mathcal{P}(S)$ for some sparse set $S \in \mathcal{NP}$. Furthermore, as shown in [Har83], $\mathcal{P}(\mathcal{NP} \cap \mathrm{SPARSE}) = \mathcal{P}(\mathcal{NP} \cap \mathrm{TALLY})$, implying that $\mathcal{NP} \cap \mathcal{P}/\log \subseteq \mathcal{P}(\mathcal{NP} \cap \mathrm{TALLY})$. Assuming $\mathcal{NP} \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$, the collapse now follows by Theorem 2.3.

2. As shown in [AKM96], every set A in $\mathrm{IC}[\log, \mathrm{poly}]$ is in $\mathcal{P}(T)$ for some tally set $T \in \mathcal{P}(\mathcal{NP} \oplus A)$. Therefore, if additionally $A \in \Delta_2^p$, then $A \in \mathcal{P}(T)$ for some tally set $T \in \Delta_2^p$. Since by Theorem 2.3 every tally set in $\mathcal{AP}_{\mathcal{FP}}$ is in $\mathcal{P}$ it follows by the assumption $\Delta_2^p \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ that $A \in \mathcal{P}$.

3. As shown in [AKM96], every set A that conjunctively, disjunctively, or bounded truth-table reduces to some sparse set is in $\mathcal{P}(S)$ for some sparse oracle S that can be decided in $\mathcal{NP}(A \oplus \mathcal{NP})$ with the help of an advice function h in $\mathcal{FP}(\mathcal{NP}(A))$. Therefore, if additionally $A \in \Sigma_2^p \cap \Pi_2^p$, then h is computable in $\mathcal{FP}(\Sigma_2^p)$ and thus, using the assumption $\Sigma_2^p \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ and part two of Theorem 3.2, in $\mathcal{FP}$. This implies that $S \in \Sigma_2^p$ and using an analogous reasoning as in the proof of part one it follows that $A \in \mathcal{P}$.

4. As shown in [Köb94], every set A in $\mathcal{P}$/poly has an advice function h computable in $\mathcal{FP}(\mathcal{NP}(A) \oplus \Sigma_2^p)$. Therefore, if additionally $A \in \Sigma_2^p \cap \Pi_2^p$, then $h \in \mathcal{FP}(\Sigma_2^p)$ and thus, using the assumption $\Delta_3^p \cap \mathrm{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ and part one of Theorem 3.2, in $\mathcal{FP}$. The consequence that $\mathcal{BPP} \subseteq \mathcal{P}$ is immediate since $\mathcal{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$ [Sip83, Lau83] and $\mathcal{BPP} \subseteq \mathcal{P}$/poly [BG81].

5. As shown in [Gav95], every set A in $\mathcal{P}/\text{poly}$ is in $\mathcal{P}(T)$ for some tally set T in $\mathcal{NP}(A \oplus \Sigma_2^p)$. Therefore, if additionally $A \in \Sigma_3^p \cap \Pi_3^p$, then $A \in \mathcal{P}(T)$ for some tally set $T \in \mathcal{NP}(\Sigma_3^p \cap \Pi_3^p) = \Sigma_3^p$. Since by Theorem 2.3 every tally set in $\mathcal{AP}_{\mathcal{FP}}$ is in $\mathcal{P}$ it follows by the assumption $\Sigma_3^p \cap \text{TALLY} \subseteq \mathcal{AP}_{\mathcal{FP}}$ that $A \in \mathcal{P}$.

■

In our next theorem we consider the complexity of sets in $\mathcal{P}/\text{poly}$ that have advice functions which can be computed by nondeterministic transducers under some oracle. For the proof we need the following proposition which shows as a special case that any set $A \in \mathcal{AP}_{\mathcal{FP}}$ is efficiently decidable on any $\mathcal{P}$-printable domain B.

Proposition 3.4 *Let $A \in \mathcal{AP}_{\mathcal{FP}(C)}$ and let B be a $\mathcal{P}(C)$-printable set for some oracle C. Then there exists a set $D \in \mathcal{P}$ such that $B \subseteq D$ and $D \cap A \in \mathcal{P}$.*

Proof. Assume that B is $\mathcal{P}(C)$-printable. Consider the distribution μ_B defined by

$$\mu'_B(x) = \begin{cases} \frac{1}{n(n+1)}, & \text{if } x \text{ is the lex. } n\text{-th string in } B; \\ 0, & \text{otherwise.} \end{cases}$$

Since B is $\mathcal{P}(C)$-printable it is not difficult to see that the *distribution* function μ_B is $\mathcal{FP}(C)$-computable. Furthermore, $\mu'_B(x) \geq 1/p(|x|)(p(|x|)+1)$ for all $x \in B$, where $p(n)$ is a polynomial bounding the number of all strings in B of length at most n. Since, by assumption, $A \in \mathcal{AP}_{\mathcal{FP}(C)}$ there is a Turing machine M that decides B and whose running time, denoted by $time_M$, is polynomial on μ_B-average. Now it follows from part one of Proposition 2.6 that for some polynomial q, $time_M(x) \leq q(|x|)$ for all $x \in B$. Thus, letting D be the set of all inputs $x \in \Sigma^*$ such that $time_M(x) \leq q(|x|)$, we can conclude that $D \in \mathcal{P}$, $B \subseteq D$, and that $D \cap A \in \mathcal{P}$. ■

For an oracle B, a (multivalued) function h is in $\mathcal{NPMV}(B)$ if there exists a nondeterministic polynomial-time transducer T such that *set-$h(x)$* consists of all output values of T^B on input x. h is in $\mathcal{NPMV}_{\text{honest}}(B)$ if, additionally, there exists a constant c such that $|y|^c > |x|$ for all oracle queries y of T^B on input x. In the case $B = \emptyset$, we simply write $\mathcal{NPMV}$ instead of $\mathcal{NPMV}(\emptyset)$.

Theorem 3.5 *If $A \in \mathcal{AP}_{\mathcal{FP}(\mathcal{NP}(A))}$, then any advice function $h \in \mathcal{NPMV}_{\text{honest}}(A)$ has a refinement in $\mathcal{NPMV}$, implying that any set $L \in (\mathcal{NP} \cap co\text{-}\mathcal{NP})/poly$ that has an advice function in $\mathcal{NPMV}_{\text{honest}}(A)$ belongs to $\mathcal{NP} \cap co\text{-}\mathcal{NP}$.*

Proof. Let h be a multi-valued advice function in $\mathcal{NPMV}_{\text{honest}}(A)$ and let T be a honest nondeterministic oracle transducer computing h under oracle A. Let q be a polynomial bounding the running time of T and let c be a constant such that $|y|^c > |x|$ for all oracle queries y of T^B on input x. Consider the following set B consisting of all queries y asked by T^A on input 0^n on its leftmost successful computation:

$$B = \{y \mid \exists n : T^A(0^n) \text{ asks query } y \text{ on the leftmost successful computation}\}.$$

Since T is honest it follows that B is $\mathcal{P}(\mathcal{NP}(A))$-printable. Since, by assumption, A belongs to $\mathcal{AP}_{\mathcal{FP}(\mathcal{NP}(A))}$ we can use Proposition 3.4 to conclude that there exists a set $D \in \mathcal{P}$ such that $B \subseteq D$ and $D \cap A \in \mathcal{P}$. This implies that h has a refinement in $\mathcal{NPMV}$ and hence, any set $L \in (\mathcal{NP} \cap \text{co-}\mathcal{NP})/\{h\}$ belongs to $\mathcal{NP} \cap \text{co-}\mathcal{NP}$. ■

Corollary 3.6 *If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_2^p)}$ then* IP$[\mathcal{P}/poly] \subseteq \mathcal{NP} \cap$ *co-*$\mathcal{NP}$.

Proof. Let $L \in$ IP$[\mathcal{P}/\text{poly}]$. Then L belongs to $\mathcal{P}/\text{poly}$. Let $I \in \mathcal{P}$ be an interpreter set for L and let h be an advice function, where $|h(0^n)| = p(n)$ for some polynomial p. An advice string can be checked by a co-$\mathcal{NP}$(IP$[\mathcal{P}/\text{poly}]$) computation as follows. On input $z \in \Sigma^{p(n)}$, verify for all $x \in \Sigma^n$ that $(x, z) \in I \Leftrightarrow x \in L$. Since IP$[\mathcal{P}/\text{poly}]$ is low for Σ_2^p [AKS95], i.e., $\Sigma_2^p(\text{IP}[\mathcal{P}/\text{poly}]) = \Sigma_2^{\text{p}}$, it follows that the correctness of an advice string can be checked by a Σ_2^p computation, and hence L has a multivalued advice function in $\mathcal{NPMV}_{\text{honest}}(\mathcal{NP})$. Thus the result is a consequence of part two of Theorem 3.5. ■

As mentioned in the introduction, if a decision problem L_2 is decidable in time polynomial on μ_2-average for any $\mathcal{FP}$-computable distribution μ_2, then this does not necessarily imply that also any set L_1 in $\mathcal{P}(L_2)$ is efficiently decidable on average with respect to any $\mathcal{FP}$-computable distribution μ_1. If however for any $\mathcal{FP}(\mathcal{NP})$-computable distribution μ_2, L_2 is decidable in time polynomial on μ_2-average, then we can show that indeed L_1 is efficiently solvable on average with respect to any $\mathcal{FP}$-computable distribution μ_1. For this it suffices to show that the distribution on the oracle queries induced by μ_1 and by the reduction of L_1 to L_2 is $\mathcal{FP}(\mathcal{NP})$-computable. Also, we will make use of the notion of Turing reducibility between distributional problems (L, μ) where L is a set and μ is a distribution.

Definition 3.7 *[BCGL92] A distributional problem (L_1, μ_1)* Turing reduces *to a distributional problem (L_2, μ_2) via some polynomial-time oracle machine M and a distribution μ if*

1. *μ polynomially dominates μ_1,*

2. *M Turing reduces L_1 to L_2, and*

3. *$\mu_2'(y) \geq \sum \mu'(x)/p(|y|)$ where the sum is taken over all strings x such that query y is asked by M on input x when using oracle L_2 (in symbols: $y \in Q(x, M, L_2)$).*

As stated in [BCGL92] the class of efficiently decidable distributional problems is closed under Turing reducibility.

Theorem 3.8 *[BCGL92] If (L_1, μ_1) is* Turing reducible *to (L_2, μ_2) and if L_2 is decidable in time polynomial on μ_2-average, then L_1 is decidable in time polynomial on μ_1-average.*

Using this closure property we can now easily show that all sets in Δ_{k+1}^p are efficiently decidable on average with respect to any $\mathcal{FP}(\Sigma_k^p)$-computable distribution provided that this is true for all sets in Σ_k^p.

Theorem 3.9 *Let $\mathcal{C}$ and $\mathcal{D}$ be language classes where $\mathcal{C}$ is closed under polynomial-time many-one equivalence. Then $\mathcal{C} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{C}\oplus\mathcal{D})}$ implies $\mathcal{P}(\mathcal{C}) \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{C}\oplus\mathcal{D})}$ and $\mathcal{C} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{D})}$ implies $\mathcal{P}(\mathcal{C}) \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{D})}$.*

Proof. Assume that $\mathcal{C} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{C}\oplus\mathcal{D})}$ and consider an arbitrary set $L \in \mathcal{P}(A)$ and an arbitrary distribution $\mu_1 \in \mathcal{FP}(\mathcal{C} \oplus \mathcal{D})$ where $A \in \mathcal{C} \oplus \mathcal{D}$. We have to show that L is efficiently decidable on average with respect to μ_1. Letting $\mathit{pad}(A) = \{1^{|x|}0xw10^i \mid x \in \Sigma^*, i \geq 0, w \in A\}$ it is not difficult to see that $\mathit{pad}(A)$ and A are equivalent under polynomial-time many-one reductions. Moreover, we can assume that $L = L(M, \mathit{pad}(A))$ for a polynomial-time oracle Turing machine M which for some polynomial p asks on input x only queries y of the form $1^{|x|}0xw10^i$ where $w \in \Sigma^*$ and $i = p(|x|) - |w|$. Hence it is easy to determine for any query y of M the corresponding input string x_y (which is unique). Furthermore, we assume that on any input x, M asks at least one query.

Now consider the distribution μ_2 induced by μ_1 on the queries of M:

$$\mu_2'(y) = \begin{cases} \frac{\mu_1'(x_y)}{\|Q(x_y,M,\mathit{pad}(A))\|}, & \text{if } y \in Q(x_y, M, \mathit{pad}(A)), \\ 0, & \text{otherwise.} \end{cases}$$

It is not difficult to see that (L, μ_1) Turing reduces to $(\mathit{pad}(A), \mu_2)$ via M and μ_1. Thus it only remains to show that μ_2 is $\mathcal{FP}(\mathcal{C} \oplus \mathcal{D})$-computable. By the monotonicity of the oracle queries of M it follows that

$$\mu_2(y) \;=\; \sum_{z \leq y} \mu_2'(z) \;=\; \left(\sum_{x < x_y} \mu_1'(x) \right) + \frac{l}{m} \cdot \mu_1'(x_y) \;=\; \mu_1(x_y^-) + \frac{l}{m} \cdot \mu_1'(x_y)$$

where $m = \|Q(x_y, M, \mathit{pad}(A))\|$, l is the number of strings in $Q(x_y, M, \mathit{pad}(A))$ less than or equal to y, and x_y^- is the predecessor of x_y in lexicographic order. But this proves that $\mu_2 \in \mathcal{FP}(\mathcal{C} \oplus \mathcal{D})$.

Now, assume that $\mathcal{C} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{D})}$ and consider an arbitrary set $L \in \mathcal{P}(\mathcal{D})$ and an arbitrary distribution $\mu_1 \in \mathcal{FP}(\mathcal{D})$. Then it follows exactly as above that L is efficiently decidable on average with respect to μ_1 since the distribution μ_2 induced by μ_1 on the queries of M is now easily seen to be $\mathcal{FP}(\mathcal{D})$-computable. ■

Corollary 3.10

1. *If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{NP})}$ then $\Delta_2^p \cap \mathrm{IC}[\log, \mathrm{poly}] = \mathcal{P}$.*

2. *If $\Sigma_2^p \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_2^p)}$ then $\Sigma_2^p \cap \Pi_2^p \cap \mathcal{P}/poly = \mathcal{P}$ and in particular, $\mathcal{BPP} \subseteq \mathcal{P}$.*

Proof. By Theorem 3.9, $\Sigma_k^p \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$ implies $\mathcal{P}^{\Sigma_k^p} \subseteq \mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)}$. Since $\mathcal{AP}_{\mathcal{FP}(\Sigma_k^p)} \subseteq \mathcal{AP}_{\mathcal{FP}}$ the results follow from Theorem 3.3 and Corollary 3.6. ■

4 Eliminating random oracle queries

As mentioned in the introduction, any random self-reducible set which can be decided in time polynomial on average (under the distribution induced by the random self-reduction) can be decided by a randomized algorithm in expected polynomial time. As shown by Feigenbaum and Fortnow, many complexity classes like $\mathcal{PP}$, $\mathcal{M}\mathrm{od}_k\mathcal{P}$, and $\mathcal{PSPACE}$ have complete sets that are random self-reducible. By combining the results stated in [FF93] with Corollary 4.3 below, it is not hard to verify that for $\mathcal{K} \in \{\mathcal{P}(\mathcal{PP}), \mathcal{MP}, \mathcal{M}\mathrm{od}_k\mathcal{P}, \mathcal{M}\mathrm{od}\mathcal{P}, \mathcal{PSPACE}\}$, $\mathcal{K}$ is not contained in $\mathcal{AP}_{\mathcal{FP}}$ unless $\mathcal{K} \subseteq \mathcal{ZPP}$ where the middle bit class $\mathcal{MP}$, the classes $\mathcal{M}\mathrm{od}_k\mathcal{P}$, $k \geq 2$, and the generalized Mod class $\mathcal{M}\mathrm{od}\mathcal{P}$ have been introduced and studied in [GKR$^+$95], [CH90, Her90, BG92], and [KT96], respectively.

In Theorem 4.4 below we show a similar collapse for the subclass of $\mathcal{P}/\mathrm{poly}$ consisting of all sets L for which a multivalued advice function can be computed by a randomized algorithm under an oracle that is easily decidable on average. Let $h \in \mathcal{NPMV}(B)$. Then we say that $h \in \mathcal{FZPP}(B)$ if h is computable by an $\mathcal{NPMV}(B)$ transducer that, when considered as a probabilistic Turing machine, on any input x produces with probability at least $1/2$ some output y.

Let M be a randomized Turing machine. If we fix a sequence $r \in \{0,1\}^*$ of the probabilistic choices of M, then the computation of M on input x is deterministic. We use $M_r(x)$ to denote the output of M on input x and computation path r. Assuming that M uses a functional oracle $f : \Sigma^* \to \Sigma^*$ and p is a polynomial bounding the running time of M, we define for any input x the distribution $\mu_{M,f,x}$ induced by M on input x with oracle f,

$$\mu'_{M,f,x}(y) = \frac{\|R_y(x)\|}{\|R(x)\|}$$

where

$$R_y(x) = \{(r,i) \mid r \in \{0,1\}^{p(n)} \text{ and } y \text{ is the } i\text{th query of} M_r^f(x)\}$$

and $R(x) = \bigcup_{y \in \Sigma^*} R_y(x)$.

Assume that $g \in \mathcal{FZPP}^f$ via some transducer M. Then we say that the computation of M^f is dominated by a distribution μ (in symbols: $g \in \mathcal{FZPP}^f_\mu$ via M) if μ dominates the ensemble $(\mu_{M,f,x})_{x \in \Sigma^*}$, i.e., there exists a polynomial s such that for all x and y, $\mu'(y) \geq \mu'_{M,f,x}(y)/s(|x|)$. By $\mathcal{ZPP}^f_\mu$ we denote the class of all languages whose characteristic function belongs to $\mathcal{FZPP}^f_\mu$.

In the following we use A_f to denote the set that contains for any argument y of f all strings yz such that the ith bit of $f(y)$ is one (in the context of the present paper we can always assume that $|f(y)| < q(|y|)$ for some fixed polynomial q),

$$A_f = \{y\mathrm{bin}_{q(|y|)}(i) \mid i = |f(y)| + 1 \text{ or } 1 \leq i \leq |f(y)| \text{ and the } i\text{th bit of } f(y) \text{ is one }\}.$$

Lemma 4.1 *Let f be a function and q be a polynomial such that $|f(y)| < q(|y|)$. Then $\mathcal{FZPP}^f_\mu \subseteq \mathcal{FZPP}^{A_f}_\nu$ where ν is the distribution defined as*

$$\nu'(u) = \begin{cases} \mu'(y)/q(|y|), & \text{if } u = y\mathrm{bin}_{q(|y|)}(i) \text{ for some } i \in \{1,\ldots,q(|y|)\}, \\ 0, & \text{otherwise.} \end{cases}$$

Proof. Let $g \in \mathcal{FZPP}^f_\mu$ via some machine M and let s be a polynomial such that for all x and y, $\mu'_{M,f,x}(y)/s(|x|) \le \mu'(y)$. Let M' be an oracle machine that simulates M by substituting each oracle query y by the sequence $y\mathrm{bin}_{q(|y|)}(i)$, $i = 1, \cdots, q(|y|)$ of oracle queries to A_f. Then $g \in \mathcal{FZPP}^{A_f}$ via M'. Furthermore, it is not hard to see that the distribution $\mu_{M',A_f,x}$ on the oracle queries induced by M' on input x fulfills for all strings $u = y\mathrm{bin}_{q(|y|)}(i)$ where $1 \le i \le q(|y|)$ the inequality

$$\mu'_{M',A_f,x}(y\mathrm{bin}_{q(|y|)}) = \mu'_{M,f,x}(y)/q(|y|) \le \mu'(y)s(|x|)/q(|y|) = \nu'(y\mathrm{bin}_{q(|y|)})s(|x|).$$

Since $\mu'_{M',A_f,x}(u) = 0$ for all other strings u, this shows that ν dominates the ensemble $(\mu_{M',A_f,x})_{x\in\Sigma^*}$. ■

Theorem 4.2 *If $A_f \in \mathcal{AP}_{\mathcal{FP}}$ and if μ is a distribution in $\mathcal{FP}$, then $\mathcal{FZPP}^f_\mu \subseteq \mathcal{FZPP}$ and, in particular, $\mathcal{ZPP}^f_\mu \subseteq \mathcal{ZPP}$.*

Proof. Let $g \in \mathcal{FZPP}^f_\mu$. From Lemma 4.1 it follows that g belongs to $\mathcal{FZPP}^{A_f}_\nu$ via some probabilistic oracle Turing machine M and some distribution $\nu \in \mathcal{FP}$. Since the induced distribution on the oracle queries remains the same if we run an oracle machine an arbitrary number of times, we can assume that on any input x, M outputs "?" with probability at most 1/8. Let p be a polynomial time bound for M and let s be a polynomial such that for all x and y, $\mu_{M,A_f,x}(y)/s(|x|) \le \nu(y)$. Since by assumption $A_f \in \mathcal{AP}_{\mathcal{FP}}$, it follows from Proposition 2.5 that there exists a machine M_{A_f} deciding A_f such that for some polynomial t and all n, $\nu'\{y \in \Sigma^{\le p(n)} \mid time_{M_{A_f}}(y) > t(n)\} \le \frac{1}{8p(n)\cdot s(n)}$. Hence, for all x it holds that

$$\mu'_{M,A_f,x}\{y \in \Sigma^{\le p(|x|)} \mid time_{M_{A_f}}(y) > t(|x|)\} \ \le \ \frac{1}{8p(|x|)}.$$

Recall that $\mu'_{M,A_f,x}(y) = \|R_y(x)\|/\|R_x\|$, where $R_y(x)$ is the set of positions (r,i) in the computation tree of M^{A_f} on input x where M^{A_f} queries y and $R_x = \bigcup_y R_y(x)$ is the set of all query positions of M^{A_f} on input x. Therefore, each position (r,i) at which y is asked as a query contributes an amount of $1/\|R_x\|$ to the probability $\mu_{M,A_f,x}(y)$.

This implies that the number of query positions (r,i) such that $time_{M_{A_f}}(y) > t(|x|)$ where y is the ith query of $M_r^{A_f}(x)$, is bounded by $\|R_x\|/8p(|x|)$. Hence, the number of computations r of $M^{A_f}(x)$ such that $time_{M_{A_f}}(y) > t(|x|)$ for at least one query y of $M_r^{A_f}(x)$ is at most $\|R_x\|/8p(|x|) \le 2^{p(|x|)}/8$.

Therefore g can be decided as follows.

On input x, $|x| = n$, guess a string r of length $p(n)$ and simulate $M_r(x)$ where each oracle query y is answered by running $M_{A_f}(y)$ for at most $t(n)$ steps. If for some query y, $M_{A_f}(y)$ runs for more than $t(n)$ steps, then output "?".

Since this algorithm does only output strings in *set*-$g(x)$ and since the probability of outputting "?" is easily verified to be at most $1/4$, it follows that a refinement of g is in $\mathcal{FZPP}$. ■

Since the standard distribution μ_{st} is easily seen to be in $\mathcal{FP}$, we immediately get the following corollary.

Corollary 4.3 *If $A_f \in \mathcal{AP}_{\mathcal{FP}}$, then any function $h \in \mathcal{FZPP}^f_{\mu_{st}}$ has a refinement in $\mathcal{FZPP}$ and, in particular, $\mathcal{ZPP}^f_{\mu_{st}} \subseteq \mathcal{ZPP}$.*

Now we are ready to show that any advice function which can be computed by a randomized algorithm under an oracle that is easily decidable on average is computable in the same way without the help of an oracle.

Theorem 4.4 *Any advice function that is computable in $\mathcal{FZPP}(D)$ where $\mathcal{P}(D) \subseteq \mathcal{AP}_{\mathcal{FP}}$ has a refinement in $\mathcal{FZPP}$.*

Proof. Assume that h is a multi-valued advice function in $\mathcal{FZPP}(D)$ and let T be a probabilistic oracle transducer computing h under oracle D. Consider the function f defined by $f(r) = T^D_r(0^n)$ if $r \in \{0,1\}^{p(n)}$ for some n and $T^D_r(0^n)$ is a successfull computation, and $f(r) = 0$ otherwise. Note that $h \in \mathcal{FZPP}^f_{\mu_{\mathrm{st}}}$ via a probabilistic transducer M that on input 0^n randomly guesses a string $r \in \Sigma^{p(n)}$ and outputs $f(r)$. Since A_f in $\mathcal{P}(D)$ and since by assumption, $\mathcal{P}(D) \subseteq \mathcal{AP}_{\mathcal{FP}}$, it follows from Corollary 4.3 that the result follows from the assumption that $\mathcal{P}(D) \subseteq \mathcal{AP}_{\mathcal{FP}}$. ■

By using results from [BCG$^+$96, KW95, Lip91, FF93, GKR$^+$95, KT96] it is easy to derive the following corollary.

Corollary 4.5

1. *[Wat96] If $\Delta^p_2 \subseteq \mathcal{AP}_{\mathcal{FP}}$ then every self-reducible set in $\mathcal{P}$/poly is in $\mathcal{ZPP}$.*
2. *If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{NP})}$ then every self-reducible set in $\mathcal{P}$/poly is in $\mathcal{ZPP}$.*
3. *For $\mathcal{K} \in \{\mathcal{P}(\mathcal{PP}), \mathcal{MP}, \mathcal{M}od\mathcal{P}, \mathcal{PSPACE}\}$, $\mathcal{K}$ is not contained in $\mathcal{AP}_{\mathcal{FP}}$ unless $\mathcal{K} = \mathcal{P}$.*

Proof.

1. Since, as shown in [BCG$^+$96, KW95], every self-reducible set $L \in \mathcal{P}$/poly has an advice function in $\mathcal{FZPP}(\mathcal{NP})$, the result follows from Theorem 4.4.

2. By Theorem 3.9, $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{NP})}$ implies $\Delta_2^p \subseteq \mathcal{AP}_{\mathcal{FP}(\mathcal{NP})}$. Since $\mathcal{AP}_{\mathcal{FP}(\mathcal{NP})} \subseteq \mathcal{AP}_{\mathcal{FP}}$ the result follows from Corollary 4.5.

3. We first consider the case that $\mathcal{K} = \mathcal{P}(\mathcal{PP})$. Under the assumption $\mathcal{P}(\mathcal{PP}) \subseteq \mathcal{AP}_{\mathcal{FP}}$ it is easy to design an algorithm that computes the permanent efficiently for all but a sufficiently small (polynomial) fraction of all $n \times n$ matrices (over $\mathrm{GF}(p)$ where $p > n+1$ is prime). As shown in [Lip91] this implies that the permanent (over $\mathrm{GF}(p)$) of any $n \times n$ matrix can be computed in expected polynomial time. Since computing the permanent (over $\mathrm{GF}(p)$ where p is given as part of the input) is hard for $\mathcal{P}(\mathcal{PP})$ [Val79] we can conclude that $\mathcal{PP} = \mathcal{ZPP}$. Since $\Delta_3^p \subseteq \mathcal{P}(\mathcal{PP})$ [Tod91], it follows by Corollary 3.3 that $\mathcal{PP} = \mathcal{P}$.

   Next assume that $\mathcal{K} \in \{\mathcal{MP}, \mathcal{M}\mathrm{od}\mathcal{P}\}$ is contained in $\mathcal{AP}_{\mathcal{FP}}$. Then by Theorem 3.9 it follows that $\mathcal{P}(\mathcal{K})$ is contained in $\mathcal{AP}_{\mathcal{FP}}$. Since $\mathcal{P}(\mathcal{PP}) \subseteq \mathcal{P}(\mathcal{K})$ [GKR$^+$95, KT96] we get that $\mathcal{PP} = \mathcal{P}$, implying that $\mathcal{K} \subseteq \mathcal{P}$.

   For the case $\mathcal{K} = \mathcal{PSPACE}$ we use the result in [FF93] that $\mathcal{PSPACE}$-complete sets are random self-reducible. From the proof given in [FF93] it is easy to verify that there is a function $f \in \mathcal{FP}(\mathcal{PSPACE})$ such that $\mathcal{PSPACE} \subseteq \mathcal{ZPP}^f_{\mu_{\mathrm{st}}}$. Since the assumption $\mathcal{PSPACE} \subseteq \mathcal{AP}_{\mathcal{FP}}$ implies that $A_f \in \mathcal{AP}_{\mathcal{FP}}$, it follows by Corollary 4.3 that $\mathcal{ZPP}^f_{\mu_{\mathrm{st}}} \subseteq \mathcal{ZPP}$, implying that $\mathcal{PSPACE} = \mathcal{ZPP}$. Since $\Delta_3^p \subseteq \mathcal{PSPACE}$, it follows by Corollary 3.3 that $\mathcal{PSPACE} = \mathcal{P}$.

■

It is interesting to note that Corollary 4.5 implies stronger collapse consequences for the polynomial hierarchy. For example, if $\Delta_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\mathcal{NP} \subseteq \mathcal{P}/\mathrm{poly}$ implies $\mathcal{PH} = \mathcal{ZPP}$.

Finally, by applying a technique used to show that $\mathcal{MA} \subseteq \mathcal{ZPP}(\mathcal{NP})$ [AK97] we extend a result in [Imp95] showing that $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ implies $\mathcal{BPP} = \mathcal{ZPP}$. More specifically, we derive under the same assumption $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ that $\mathcal{MA}$ can be derandomized, i.e., $\mathcal{MA} = \mathcal{NP}$, whereas under the stronger assumption $\Sigma_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ also $\mathcal{AM}$ can be derandomized, i.e., $\mathcal{AM} = \mathcal{NP}$. Note that $\mathcal{AM} = \mathcal{NP}$ has some immediate strong implications as, for example, Graph Isomorphism is in $\mathcal{NP} \cap \text{co-}\mathcal{NP}$.

A *nondeterministic circuit* c has two kinds of input gates: in addition to the actual inputs $x_1, \ldots, x_n$, c has a series of distinguished *guess inputs* $y_1, \ldots, y_m$. The value computed by c on input $x \in \Sigma^n$ is 1 (in symbols, $c(x) = 1$) if there exists a $y \in \Sigma^m$ such that $c(xy) = 1$, and 0 otherwise [SV85].

Next we recall the notion of hardness of boolean functions. We denote the class of boolean functions that can be computed by some (non)deterministic circuit c of size at most s by $\mathcal{CIR}(s)$ ($\mathcal{NCIR}(s)$, respectively).

Definition 4.6 *(cf. [Yao82, NW94]) Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function, $\mathcal{C}$ be a set of boolean functions, and let $r \in \mathbb{N}$ be a positive integer. f is said to be r-*hard for $\mathcal{C}$

if for all n-ary boolean functions g in $\mathcal{C}$,

$$\frac{1}{2} - \frac{1}{r} < \frac{\|\{x \in \{0,1\}^n \mid f(x) = g(x)\}\|}{2^n} < \frac{1}{2} + \frac{1}{r}.$$

*f is called $\mathcal{CIR}(r)$-*hard *($\mathcal{NCIR}(r)$-*hard*) if f is r-hard for $\mathcal{CIR}(r)$ ($\mathcal{NCIR}(r)$, respectively).*

Theorem 4.7

1. *If $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\mathcal{MA} = \mathcal{NP}$.*
2. *If $\Sigma_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ then $\mathcal{AM} = \mathcal{NP}$.*

Proof.

1. Based on the Nisan-Wigderson design of a pseudorandom generator [NW94] it has been recently shown that any set L in $\mathcal{MA}$ can be decided in $\mathcal{ZPP}(\mathcal{NP})$ [AK97]. Basically, the $\mathcal{ZPP}(\mathcal{NP})$ computation proceeds as follows:

   > On input x, the $\mathcal{ZPP}$ base machine randomly guesses some function $g : \{0,1\}^{O(\log n)} \to \{0,1\}$ and asks a co-$\mathcal{NP}$ oracle whether g is hard to approximate by circuits of some suitably chosen (polynomial) size. If so, g is used to build a pseudorandom generator G that can be used to derandomize the $\mathcal{MA}$ decision procedure for L. Consequently, one further $\mathcal{NP}$ query suffices to decide whether x belongs to L.

   A crucial point for this algorithm to work is that the randomly chosen function g is hard to approximate with high probability. Hence, under the assumption that $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ we can turn the above $\mathcal{ZPP}(\mathcal{NP})$ computation into an $\mathcal{NP}$ computation as follows. Consider a suitable efficient on average algorithm M (which exists by the assumption that $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$) which verifies that g is not hard to approximate. Then M rejects most of its inputs within a polynomial time bound p. Consequently, an $\mathcal{NP}$ algorithm can guess some boolean function g, verify the hardnes of g by running $M(g)$, and then use the corresponding pseudorandom generator G to decide whether x belongs to L.

   We proceed by giving a formal proof. By the way $\mathcal{MA}$ is defined, there exist a polynomial p and a set $B \in \mathcal{P}$ such that for all x, $|x| = n$,

   $$\begin{aligned} x \in L &\Rightarrow \exists y, |y| = p(n) : \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle x, y, r\rangle \in B] \geq 3/4, \\ x \notin L &\Rightarrow \forall y, |y| = p(n) : \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle x, y, r\rangle \in B] \leq 1/4 \end{aligned}$$

   where the subscript $r \in_R \{0,1\}^{p(n)}$ means that the probability is taken by choosing r uniformly at random from $\{0,1\}^{p(n)}$.

For fixed strings x and y, the decision procedure for B on input x, y, r can be simulated by some circuit $c_{x,y}$ with inputs $r_1, \ldots, r_{p(n)}$, implying that

$$\begin{array}{lcl} x \in L & \Rightarrow & \exists y, |y| = p(n) : \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[c_{x,y}(r) = 1] \geq 3/4, \\ x \notin L & \Rightarrow & \forall y, |y| = p(n) : \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[c_{x,y}(r) = 1] \leq 1/4 \end{array}$$

where w.l.o.g. we can assume that the size of $c_{x,y}$ is bounded by $p^2(|x|)$ and that $p(n) > 4n$. As shown in [NW94] (see also [AK97]) there is an $\mathcal{FP}$ function G having the following property: For any $\mathcal{CIR}(p^3(n))$-hard boolean function $g : \{0,1\}^{m(n)} \to \{0,1\}$, where $m(n) = 12 \log p(n)$, and for every $p(n)$-input circuit c of size at most $p^2(n)$ it holds that

$$\left| \mathit{Prob}_{y \in_R \{0,1\}^{p(n)}}[c(y) = 1] - \mathit{Prob}_{s \in_R \{0,1\}^{l(n)}}[c(G(g,s)) = 1] \right| \leq 1/p(n) \qquad (1)$$

where $l(n) = 24m(n)$. Furthermore, for all sufficiently large n, a randomly chosen boolean function $g : \{0,1\}^{m(n)} \to \{0,1\}$ is $\mathcal{CIR}(p^3(n))$-hard with probability at least $1 - e^{-p^3(n)}$. Since the set

$$A = \{g : \{0,1\}^{m(n)} \to \{0,1\} \mid n \geq 1, g \text{ is not } \mathcal{CIR}(p^3(n))\text{-hard } \}$$

belongs to $\mathcal{NP}$, we can use the assumption $\mathcal{NP} \subseteq \mathcal{AP}_{\mathcal{FP}}$ to get an algorithm M for A that is efficient on average w.r.t. the standard distribution. Exploiting the fact that at least a fraction of $1 - e^{-p^3(n)}$ of the strings of length $2^{m(n)}$ are rejected by M, it follows from part two of Proposition 2.6 that there is a polynomial q such that M rejects at least one string of length $2^{m(n)}$ within $q(n)$ steps. Now we are ready to give the $\mathcal{NP}$ decision procedure for L:

```
input $x$, $|x| = n$;
guess $g : \{0,1\}^{m(n)} \to \{0,1\}$;
if $M(g)$ rejects within $q(n)$ steps then
    guess $y \in \Sigma^{p(n)}$;
    $k := \sum_{s \in \{0,1\}^{l(n)}} c_{x,y}(G(g,s))$;
    if $k \geq 2^{l(n)-1}$ then accept else reject
else reject
```

Using inequality 1 above it is easy to verify that this algorithm decides L correctly.

2. The proof is similar to the one above. The only difference is that now a Σ_2^p oracle has to be used to check whether g is hard to approximate by nondeterministic circuits. Let $L \in \mathcal{AM}$. Then there exist a polynomial p and a set $D \in \mathcal{NP}$ such that for all x, $|x| = n$

$$\begin{array}{lcl} x \in L & \Rightarrow & \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle x, r \rangle \in D] \geq 3/4, \\ x \notin L & \Rightarrow & \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle x, r \rangle \in D] \leq 1/4. \end{array}$$

For a fixed input x, the decision procedure for D on input x, r can be simulated by some nondeterministic circuit c_x with input r, implying that

$$\begin{aligned} x \in L &\Rightarrow \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[c_x(r) = 1] \geq 3/4, \\ x \notin L &\Rightarrow \mathit{Prob}_{r \in_R \{0,1\}^{p(n)}}[c_x(r) = 1] \leq 1/4 \end{aligned}$$

where again we can assume that the size of c_x is bounded by $p^2(|x|)$. As shown in [AK97] there is an $\mathcal{FP}$ function G having the following property: For any $\mathcal{NCIR}(p^3(n))$-hard boolean function $g : \{0,1\}^{m(n)} \to \{0,1\}$, where $m(n) = 12 \log p(n)$, and for every $p(n)$-input nondeterministic circuit c of size at most $p^2(n)$ it holds that

$$\left|\mathit{Prob}_{y \in_R \{0,1\}^{p(n)}}[c(y) = 1] - \mathit{Prob}_{s \in_R \{0,1\}^{l(n)}}[c(G(g,s)) = 1]\right| \leq 1/p(n)$$

where $l(n) = 24m(n)$. Furthermore, for all sufficiently large n, a randomly chosen boolean function $g : \{0,1\}^{m(n)} \to \{0,1\}$ is $\mathcal{NCIR}(p^3(n))$-hard with probability at least $1 - e^{-p^3(n)}$. Since the set

$$A = \{g : \{0,1\}^{m(n)} \to \{0,1\} \mid n \geq 1, g \text{ is not } \mathcal{NCIR}(p^3(n))\text{-hard }\}$$

belongs to Σ_2^p, we can use the assumption $\Sigma_2^p \subseteq \mathcal{AP}_{\mathcal{FP}}$ to get an algorithm M for A that is efficient on average w.r.t. the standard distribution. Exactly as in part one above it follows that there is a polynomial q such that M rejects at least one string of length $2^{m(n)}$ within $q(n)$ steps. Hence, L can be decided by the following $\mathcal{NP}$ algorithm:

```
input $x$, $|x| = n$;
guess $g : \{0,1\}^{m(n)} \to \{0,1\}$;
if $M(g)$ rejects within $q(n)$ steps then
   if $\sum_{s \in \{0,1\}^{l(n)}} c_x(G(g,s)) \geq 2^{l(n)-1}$ then accept else reject
else reject
```

Note that the condition of the second if-statement can be evaluated in $\mathcal{NP}$ by guessing for each $s \in \{0,1\}^{l(n)}$ some assignment for the guess inputs of the nondeterministic circuit c_x on actual input $G(g,s)$ and checking whether the sum over the corresponding output bits exceeds $2^{l(n)-1}$.

■

Note that the above proof shows that in order to derive $\mathcal{MA} = \mathcal{NP}$ ($\mathcal{AM} = \mathcal{NP}$) it suffices to assume that for any set L in co-$\mathcal{NP}$ (respectively, Π_2^p) and any $\mathcal{FP}$-computable distribution μ there is some nondeterministic Turing machine for L whose running time is polynomial on μ-average.

References

[AHH$^+$93] V. Arvind, Y. Han, L. A. Hemachandra, J. Köbler, A. Lozano, M. Mundhenk, M. Ogiwara, U. Schöning, R. Silvestri, and T. Thierauf. Reductions to sets of low information content. In K. Ambos-Spies, S. Homer, and U. Schöning, editors, *Complexity Theory, Current Research*, 1–45. Cambridge U. Press, 1993.

[AK97] V. Arvind and J. Köbler. On resource-bounded measure and pseudorandomness. In *Proc. 17th Conference on Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science #1346, 235–249. Springer-Verlag, 1997.

[AKM96] V. Arvind, J. Köbler, and M. Mundhenk. Upper bounds for the complexity of sparse and tally descriptions. *Mathematical Systems Theory*, **29**(1):63–94, 1996.

[AKS95] V. Arvind, J. Köbler, and R. Schuler. On helping and interactive proof systems. *International Journal of Foundations of Computer Science*, **6**(2):137–153, 1995.

[BCG$^+$96] N. Bshouty, R. Cleve, R. Gavaldà, S. Kann an, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, **52**:421–433, 1996.

[BCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, **44**:193–219, 1992.

[BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, second edition, 1995.

[BF90] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proc. 7th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science #415, 37–48. Springer-Verlag, 1990.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, **1**:1–40, 1991.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, **3**:307–318, 1993.

[BG81] C.H. Bennet and J. Gill. Relative to a random oracle A, $\mathrm{P(A)} \neq \mathrm{NP(A)} \neq \text{co-NP(A)}$ with probability 1. *SIAM Journal on Computing*, **10**:69–113, 1981.

[BG92] R. Beigel and J. Gill. Counting classes: thresholds, parity, mods, and fewness. *Theoretical Computer Science*, **103**:3–23, 1992.

[Boo74] R. Book. Tally languages and complexity classes. *Information and Control*, **26**:186–193, 1974.

[BS92] J.L Balcázar and U. Schöning. Logarithmic advice classes. *Theoretical Computer Science*, **99**:279–290, 1992.

[CH90] J. Cai and L. A. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, **23**:95–106, 1990.

[CS96] J.-Y. Cai and A. Selman. Fine separation of average time complexity classes. In *Proc. 13th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science #1046, 331–343. Springer-Verlag, 1996.

[FF93] J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, **22**:994–1005, 1993.

[Gav95] R. Gavaldà. Bounding the complexity of advice functions. *Journal of Computer and System Sciences*, **50**(3):468–475, 1995.

[GKR$^+$95] F. Green, J. Köbler, K. Regan, T. Schwentick, and J. Torán. The power of the middle bit of a #P function. *Journal of Computer and System Sciences*, **50**(3):456–467, 1995.

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

[Har83] J. Hartmanis. On sparse sets in NP–P. *Information Processing Letters*, **16**:55–60, 1983.

[Her90] U. Hertrampf. Relations among MOD-classes. *Theoretical Computer Science*, **74**:325–328, 1990.

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

[KL80] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium on Theory of Computing*, 302–309. ACM Press, 1980.

[Köb94] J. Köbler. Locating P/poly optimally in the extended low hierarchy. *Theoretical Computer Science*, **134**(2):263–285, 1994.

[KS97] J. Köbler and R. Schuler. Using efficient average-case algorithms to collapse worst-case complexity classes. Technical Report UIB-97-18, University of Ulm, 1997.

[KT96] J. Köbler and S. Toda. On the power of generalized MOD-classes. *Mathematical Systems Theory*, **29**(1):33–46, 1996.

[KW95] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. In *Proc. 22nd International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science #944, 196–207. Springer-Verlag, 1995.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, **17**:215–217, 1983.

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

[Lip91] R. J. Lipton. New directions in testing. In J. Feigenbaum and M. Merritt, editors, *Distributed Computing and Cryptography*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science #2. American Mathematical Society, 1991.

[LM96] J. H. Lutz and E. Mayordomo. Cook versus Karp-Levin: separating reducibilities if NP is not small. *Theoretical Computer Science*, **164**:141–163, 1996.

[Lut97a] J. H. Lutz. Observations on measure and lowness for Δ_2^{P}. *Theory of Computing Systems*, **30**:429–442, 1997.

[Lut97b] J. H. Lutz. The quantitative structure of exponential time. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, 225–260. Springer-Verlag, 1997.

[LV92] M. Li and P.M.B. Vitányi. Average case complexity under the universal distribution equals worst-case complexity. *Information Processing Letters*, **42**:145–149, 1992.

[Mil93] P.B. Milterson. The complexity of malign measures. *SIAM Journal on Computing*, **22**(1):147–156, 1993.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, **49**:149–167, 1994.

[OKSW94] P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance complexity. *Journal of the ACM*, **41**(1):96–121, 1994.

[Sch86] U. Schöning. *Complexity and Structure*, Lecture Notes in Computer Science #211. Springer-Verlag, 1986.

[Sch90] R. E. Schapire. The emerging theory of average-case complexity. Technical Report TM-431, Massachusetts Institut of Technology, 1990.

[Sch96] R. Schuler. Truth-table closure and Turing closure of average polynomial time have different measures in EXP. In *Proc. 11th Annual IEEE Conference on Computational Complexity*, 190–197. IEEE Computer Society Press, 1996.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proc. 15th ACM Symposium on Theory of Computing*, 330–335. ACM Press, 1983.

[SV85] S. Skyum and L. G. Valiant. A complexity theory based on boolean algebra. *Journal of the ACM*, **32**:484–502, 1985.

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

[SY95] R. Schuler and T. Yamakami. Sets computable in polynomial time on average. In *Proc. 1st International Computing and Combinatorics Conference*, Lecture Notes in Computer Science #959, 400–409. Springer-Verlag, 1995.

[SY96] R. Schuler and T. Yamakami. Structural average case complexity. *Journal of Computer and System Sciences*, **52**:308–327, 1996.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, **20**:865–877, 1991.

[Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, **8**:189–201, 1979.

[Wat96] O. Watanabe, 1996. Personal communication.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symposium on the Foundations of Computer Science*, 80–91. IEEE Computer Society Press, 1982.