# Notes on Levin's Theory of Average-Case Complexity

Oded Goldreich

**Abstract.** In 1984, Leonid Levin initiated a theory of average-case complexity. We provide an exposition of the basic definitions suggested by Levin, and discuss some of the considerations underlying these definitions.

**Keywords:** Average-case complexity, reductions.

This survey is rooted in the author's (exposition and exploration) work [4], which was partially reproduded in [1]. An early version of this survey appeared as TR97-058 of *ECCC*. Some of the perspective and conclusions were revised in light of a relatively recent work of Livne [21], but an attempt was made to preserve the spirit of the original survey. The author's current perspective is better reflected in [7, Sec. 10.2] and [8], which advocate somewhat different definitional choices (e.g., focusing on *typical* rather than *average* performace of algorithms).

## 1 Introduction

The average complexity of a problem is, in many cases, a more significant measure than its worst case complexity. This has motivated the development of a rich area in algorithmic research – the probabilistic analysis of algorithms [14, 16]. However, historically, this line of research focuses on the analysis of specific algorithms *with respect to specific, typically uniform, probability distributions*.

The general question of average case complexity was addressed for the first time by Levin [18]. Levin's work can be viewed as the basis for a theory of average NP-completeness, much the same way as Cook's [2] (and Levin's [17]) works are the basis for the theory of NP-completeness. Subsequent works [9, 22, 10, 21] have provided additional complete problems. Other basic complexity problems, such as decision versus search, were studied in [1].

*Levin's average-case complexity theory in a nutshell.* An average case complexity class consists of pairs, called distributional problems. Each such pair consists of a decision (resp., search) problem and a probability distribution on problem instances. We focus on the class $\mathrm{DistNP} \stackrel{\mathrm{def}}{=} \langle \mathrm{NP}, \text{P-computable} \rangle$, defined by Levin [18], which is a distributional analogue of NP: It consists of NP decision problems coupled with distributions for which the accumulative measure is polynomail-time computable. That is, P-computable is the class of distributions

for which there exists a polynomial time algorithm that on input $x$ computes the total probability of all strings $y \leq x$. The easy distributional problems are those solvable in "average polynomial-time" (a notion which surprisingly require careful formulation). Reductions between distributional problems are defined in a way guaranteeing that if $\Pi_1$ is reducible to $\Pi_2$ and $\Pi_2$ is in average polynomial-time, then so is $\Pi_1$. Finally, it is shown that the class DistNP contains a complete problem.

*Levin's average-case theory, revisited.* Levin's laconic presentation [18] hides the fact that several non-trivial choices were in the development of the average-case complexity theory. We discuss some of this choices here. Firstly, we note that our motivation is providing a theory of efficient computation (as suggested above), rather than a theory of infeasible computations (e.g., as in Cryptography). Recall that a theory of useful-for-cryptography infeasible computations does exist (cf., [5, 6]). A key difference between these two theories is that in Cryptography we seek problems for which one may generate instance-solution pairs such that solving the problem given only the instance is hard. In the theory of average-case complexity (considered below), we wish to draw the line between problems that are easy to solve and problems that are hard to solve (on the average), but we do not require an efficient procedure for generating hard (on the average) instances coupled with solutions.

Secondly, one has to admit that the class DistNP (i.e., specifically, the choice of distributions) is somewhat problematic. Indeed, P-computable distributions seem "simple" (albeit one may reconsider this view in light of [21]), but it is not clear if they exhaust all natural distributions. A much wider class, which certainly contains all natural distributions, is the class, denoted P-samplable, of all distributions having an efficient algorithm for generting instances (according to the distribution): Arguably, the instances of any problem that we may need to solve in real life are generated by some efficient process, and so the latter class of distributions (i.e., P-samplable) suffices as the scope of our theory [1]. But in this case it becomes even harder to argue that a distributional problem that refers to (a computational problem coupled wityh) an *arbitrary* P-samplable distribution is natural. Fortunately, it was showed in [13] that any distributional problem that is complete for DistNP=$\langle$NP, P-computable$\rangle$, is also complete with respect to the class $\langle$NP, P-samplable$\rangle$. Thus, in retrospect, Levin's choice only makes the theory stronger: It requires to select complete distributional problems from the restricted class $\langle$NP, P-computable$\rangle$, whereas hardness holds with respect to the wider class $\langle$NP, P-samplable$\rangle$.

As hinted above, the definition of average polynomial-time is less straightforward than one may expect. The obvious attempt at formulation this notion leads to fundamental problems that, in our opinion, deem it inadequate. (For a detailed discussion of this point, the reader is referred to Appendix A.) We believe that once the failure of the obvious attempt is understood, Levin's definition (presented below) does look a natural one.

## 2 Definitions and Notations

In this section we present the basic definitions underlying the theory of average-case complexity. Most definitions originate from Levin's work [18], but the reader is advised to look for further explanations and motivating discussions elsewhere (e.g., [14, 11, 4]). An alternative formulation, which uses probability ensembles (rather than a single infinite distribution) as a pivot, is presented in [7, Sec. 10.2.1] and [8].

For sake of simplicity, we consider the standard lexicographic ordering of binary strings, but any other efficient enumeration of strings will do.[1] By writing $x < y$ we mean that the string $x$ precedes $y$ in lexicographic order, and $y - 1$ denotes the immediate predecessor of $y$. Also, we associate pairs, triples etc. of binary strings with single binary strings in some standard manner (i.e., a standard encoding).

**Definition 1** (probability distribution functions): *A* distribution function $\mu : \{0,1\}^* \to [0,1]$ *is a non-decreasing function from strings to the unit interval* $[0,1]$ *that converges to one (i.e.,* $\mu(0) \geq 0$*,* $\mu(x) \leq \mu(y)$ *for each* $x < y$*, and* $\lim_{x\to\infty} \mu(x) = 1$*). The* density function *associated with the distribution function* $\mu$ *is denoted* $\mu'$ *and is defined by* $\mu'(0) = \mu(0)$ *and* $\mu'(x) = \mu(x) - \mu(x-1)$ *for every* $x > 0$*.*

Clearly, $\mu(x) = \sum_{y \leq x} \mu'(y)$. For notational convenience, we often describe distribution functions converging to some positive constant $c \neq 1$. In all the cases where we use this convention, it is easy to normalize the distribution such that it converges to one. An important example is the *uniform* distribution function $\mu_0$ defined as $\mu'_0(x) = \frac{1}{|x|^2} \cdot 2^{-|x|}$. (A minor variation that does converge to 1 is obtained by letting $\mu'_0(x) = \frac{1}{|x|\cdot(|x|+1)} \cdot 2^{-|x|}$.)

**Definition 2** (distributional problems): *A* distributional decision problem (resp., distributional search problem) *is a pair* $(D, \mu)$ (resp. $(S, \mu)$)*, where* $D : \{0,1\}^* \to \{0,1\}$ (resp., $S \subseteq \{0,1\}^* \times \{0,1\}^*$) *and* $\mu : \{0,1\}^* \to [0,1]$ *is a distribution function.*

In the sequel we consider mainly decision problems. Similar formulations for search problems can be easily derived.

### 2.1 Distributional-NP

Simple distributions are identified with the P-computable ones. The importance of restricting attention to simple distributions (rather than allowing arbitrary ones) is demonstrated in [1, Sec. 5.2].

[1] An efficient enumeration of strings is a 1-1 and onto mapping of strings to integers that can be computed and inverted in polynomial-time.

**Definition 3** (P-computable): *A distribution $\mu$ is in the class* P-computable *if there is a deterministic polynomial time Turing machine that on input $x$ outputs the binary expansion of $\mu(x)$. (Indeed, the running time is polynomial in $|x|$.)*

It follows that the binary expansion of $\mu(x)$ has length polynomial in $|x|$. A necessary condition for distributions to be of interest is their putting noticeable probability weight on long strings (i.e., for some polynomail, $p$, and sufficiently large $n$ the probability weight assigned to $n$-bit strings should be at least $1/p(n)$). Consider to the contrary the density function $\mu'(x) \stackrel{\text{def}}{=} 2^{-3|x|}$. An algorithm of running time $t(x) = 2^{|x|}$ will be considered to have constant on the average running-time w.r.t this $\mu$ (since $\sum_x \mu'(x) \cdot t(|x|) = \sum_n 2^{-n} = 1$).

If the distribution function $\mu$ is in P-computable, then the corresponding density function, $\mu'$, is computable in time polynomial in $|x|$. The converse, however, is false, unless P = NP (see [11]). In spite of this remark we usually present the density function, and leave it to the reader to verify that the corresponding distribution function is in P-computable.

We now present the class of distributional problems that corresponds to (the traditional) class NP. Most of results in the literature refer to this class.

**Definition 4** (DistNP): *A distributional problem $(D, \mu)$ belongs to the class* DistNP *if $D$ is an NP-predicate and $\mu$ is in* P-computable. DistNP *is also denoted* $\langle$*NP,* P-computable$\rangle$.

A wider class of distributions, denoted P-samplable, gives rise to a wider class of distributional NP problems, which was discussed in the Introduction: A distribution $\mu$ is in the class P-samplable if there exists a polynomial $p$ and a probabilistic algorithm $A$ that outputs the string $x$ with probability $\mu'(x)$ within $p(|x|)$ steps. That is, elements in a P-samplable distribution are generated in time polynomial in their length. We comment that any P-computable distribution is P-samplable, whereas the converse is false (provided one-way functions exist). For a detailed discussion see [1].

### 2.2 Average Polynomial-Time

The following definitions, regarding average polynomial-time, may seem obscure at first glance. Thus, it is important to point out that the naive formalizations of the corresponding notions suffer from serious problems such as not being closed under functional composition of algorithms, being model dependent, encoding dependent, etc. For a more detailed discussion, see Appendix A.

**Definition 5** (polynomial on the average): *A function $f : \{0,1\}^* \to \mathbb{N}$ is* polynomial on the average *with respect to a distribution $\mu$ if there exists a constant $\epsilon > 0$ such that*

$$\sum_{x \in \{0,1\}^*} \mu'(x) \cdot \frac{f(x)^\epsilon}{|x|} < \infty.$$

*The function $l(x) = f(x)^\epsilon$ is* linear on the average *w.r.t. $\mu$.*

Thus, *a function is polynomial on the average if it is bounded by a polynomial in a function that is linear on the average*. In fact, the basic definition is that of a function that is linear on the average (cf. [1]). The notion of polynomial on the average is the basis of the complexity class of distributional problems that are solvable in time that is polynomial on the average.

**Definition 6** (Average-P): *A distributional problem $(D, \mu)$ is in the class* Average-P *if there exists an algorithm $A$ solving $D$, so that the running time of $A$ is polynomial on the average with respect to the distribution $\mu$.*

We view the classes Average-P and DistNP as the average-case analogue of P and NP (respectively). We mention that if EXP $\neq$ NEXP (i.e., DTime$(2^{O(n)}) \neq$ NTime$(2^{O(n)})$), then Average-P does not contain all of DistNP (see [1]).

### 2.3 Reducibility Between Distributional Problems

We now present definitions of (average polynomial time) reductions of one distributional problem to another. Intuitively, such a reduction should be efficiently computable, yield a valid result and "preserve" the probability distribution. The purpose of the last requirement is to ensure that the reduction does not map very likely instances of the first problem to rare instances of the second problem. Otherwise, having a polynomial time on the average algorithm for the second distributional problem does not necessarily yield such an algorithm for the first distributional problem. Following is a definition of randomized Turing reductions. Definitions of deterministic and many-to-one reductions can be easily derived as special cases.

**Definition 7** (randomized reductions): *We say that the probabilistic oracle Turing machine $M$* randomly reduces *the distributional problem $(D_1, \mu_1)$ to the distributional problem $(D_2, \mu_2)$ if the following three conditions hold.*

Efficiency: *Machine $M$ is polynomial time on the average, where the average is taken over the distribution $\mu_1$ and the internal coin tosses of $M$; that is, letting $t_M(x, r)$ denote the running time of $M$ on input $x$ and internal coin tosses $r$, we require that there exists $\epsilon > 0$ such that*

$$\sum_{x,r} \mu_1'(x)\mu_0'(r) \cdot \frac{t_M(x,r)^\epsilon}{|x|} < \infty,$$

*where $\mu_0$ is the uniform distribution.*

Validity: *For every $x \in \{0,1\}^*$,*

$$\Pr[M^{D_2}(x) = D_1(x)] \geq \frac{2}{3}$$

*where $M^{D_2}(x)$ is the random variable (determined by $M$'s internal coin tosses) that denotes the output of the oracle machine $M$ on input $x$ and access to oracle for $D_2$.*

Domination: *There exists a constant $c > 0$ such that for every $y \in \{0,1\}^*$ it holds that*

$$\mu'_2(y) \geq \frac{1}{|y|^c} \cdot \sum_{x \in \{0,1\}^*} \mathtt{Ask}_M(x,y) \cdot \mu'_1(x)$$

*where $\mathtt{Ask}_M(x,y)$ is the probability (taken over $M$'s internal coin tosses) that machine $M$ asks query $y$ on input $x$.*

In the special case of *deterministic* Turing reductions the value of $M^{D_2}(x)$ is fully determined by $x$ (rather than being a random variable) and $\mathtt{Ask}_M(x,y)$ is either 0 or 1 (rather than being any arbitrary rational in $[0,1]$). In the case of a many-to-one deterministic reduction, for every $x$, there exists a unique $y$ such that $\mathtt{Ask}_M(x,y) = 1$ holds.

**Proposition:** *If $(D_1, \mu_1)$ is deterministically* (resp., randomly) *reducible to $(D_2, \mu_2)$ and $(D_2, \mu_2)$ is solvable by a deterministic* (resp., randomized) *algorithm of running time that is polynomial on the average, then so is $(D_1, \mu_1)$.*

**Proof:** Given any reduction of $(D_1, \mu_1)$ to $(D_2, \mu_2)$ we consider the distribution $\mu_I$ of the queries of the reduction on random instances distributed according to $\mu_1$; that is,

$$\mu_I(y) \stackrel{\text{def}}{=} \sum_{x \in \{0,1\}^*} \mathtt{Ask}_M(x,y) \cdot \mu'_1(x).$$

We next decouple the original reduction to a reduction of $(D_1, \mu_1)$ to $(D_2, \mu_I)$ (via the original transformation) and a reduction by the identity transformation of $(D_2, \mu_I)$ to $(D_2, \mu_2)$. (Note that each of these reductions satisfies Definition 7.) Thus, it suffices to establish the proposition for each of these two reductions.

1. Considering the reduction of $(D_1, \mu_1)$ to $(D_2, \mu_I)$, we note that when this reduction is invoked on inputs distributed according to $\mu_1$, it makes queries that are distributed according to $\mu_I$. Thus, if $(D_2, \mu_I)$ is solvable in polynomial-time on the avearge, then so is $(D_1, \mu_1)$.
2. Considering the reduction (by the identity transformation) of $(D_2, \mu_I)$ to $(D_2, \mu_2)$, it suffices to show that *if $t : \{0,1\}^* \to \mathbb{N}$ is polynomial on the average w.r.t $\mu_2$, then $t$ is polynomial on the average w.r.t $\mu_I$.*
   By the hypothesis regarding $t$ and $\mu_2$, for some for $\epsilon > 0$, it holds that $\sum_y \mu'_2(y) \frac{t(y)^\epsilon}{|y|} = O(1)$, whereas by the hypothesis that $\mu_2$ dominates $\mu_I$ it holds that $\mu'_I(y) \leq |y|^c \mu'_2(y)$ (for all $y$). Let $G \stackrel{\text{def}}{=} \{y : t(y) \leq |y|^{2c/\epsilon}\}$, and split the sum $\sum_y \mu'_I(y) \frac{t(y)^{\epsilon/2c}}{|y|}$ according to whether $y \in G$ or not. The sum $\sum_{y \in G} \mu'_I(y) \frac{t(y)^{\epsilon/2c}}{|y|}$ is upper bounded by 1 (by using $t(y)^{\epsilon/2c} \leq |y|$ for $y \in G$); whereas

$$\sum_{y \notin G} \mu'_I(y) \frac{t(y)^{\epsilon/2c}}{|y|} \leq \sum_{y \notin G} |y|^c \mu'_2(y) \frac{t(y)^{\epsilon/2}}{|y|}$$

$$\begin{aligned}
&\leq \sum_{y \notin G} t(y)^{\epsilon/2} \mu'_2(y) \frac{t(y)^{\epsilon/2}}{|y|} \\
&= \sum_{y \notin G} \mu'_2(y) \frac{t(y)^{\epsilon}}{|y|} \\
&= O(1)
\end{aligned}$$

where the first inequality uses $\mu'_I \leq |y|^c \mu'_2(y)$ and the second inequality uses $|y|^c \leq t(y)^{\epsilon/2}$ (for $y \notin G$).

The proposition follows. □

We also mention that reductions are transitive in the special case in which they are *honest*; that is, on input $x$ they ask queries of length at least $|x|^\epsilon$, for some constant $\epsilon > 0$. All known reductions have this property. Finally, we spell out the resulting definition of DistNP-completeness.

**Definition 8** (DistNP-completeness): *A distributional problem $\Pi$ is* DistNP-complete *if $\Pi \in$ DistNP and every problem in* DistNP *is reducible to $\Pi$.*

We shall actually use the most restricted notion of a reduction; that is, unless stated otherwise, all the reductions we discuss are deterministic many-to-one reductions.

### 2.4 A Generic DistNP Complete Problem

The following distributional version of *Bounded Halting*, denoted $\Pi_{BH} = (BH, \mu_{BH})$, is known to be DistNP-complete (see Section 3).

**Definition 9** (distributional Bounded Halting): *The distributional problem $\Pi_{BH} = (BH, \mu_{BH})$ consists of the following*

- Decision: $BH(M, x, 1^k) = 1$ *iff there exists a computation of the non-deterministic machine $M$ on input $x$ that halts within $k$ steps.*
- Distribution: *The distribution $\mu_{BH}$ is defined in terms of its density function*

$$\mu'_{BH}(M, x, 1^k) \stackrel{\text{def}}{=} \frac{1}{|M|^2 \cdot 2^{|M|}} \cdot \frac{1}{|x|^2 \cdot 2^{|x|}} \cdot \frac{1}{k^2}$$

Note that $\mu'_{BH}$ is very different from the uniform distribution on binary strings (e.g., consider relatively large $k$), but this seems fair since part of its input is not binary. Nevertheless, as noted by Levin, one can obtain a variant of $\Pi_{BH}$ that refers to the uniform distribution and is DistNP-complete with respect to randomized reduction. Specifically, we replace the unary time bound by a string of equal length, and assign each such string the same probability (see [7, §10.2.1.3] or [8, Sec. 2.3]).

## 3 DistNP-completeness of $\Pi_{BH}$

The proof, presented here, is due to Guretich [9]. (An alternative proof is implied by Levin's original paper [18].)

*Perspective.* In the traditional theory of $\mathcal{NP}$-completeness, the *mere* existence of complete problems is almost immediate. For example, it is extremely simple to show that the *Bounded Halting* problem is $\mathcal{NP}$-complete. Recall that Bounded Halting ($BH$) is defined over triples $(M, x, 1^k)$, where $M$ is a non-deterministic machine, $x$ is a binary string and $k$ is an integer (given in unary). The decision problem is to determine whether there exists a computation of $M$ on input $x$ that halts within $k$ steps. Clearly, Bounded Halting is in $\mathcal{NP}$ (here its crucial that $k$ is given in unary). Let $D$ be an arbitrary $\mathcal{NP}$ problem, and let $M_D$ be the non-deterministic machine solving it in time $P_D(n)$ on inputs of length $n$, where $P_D$ is a fixed polynomial. Then, the reduction of $D$ to $BH$ consists of the transformation $x \to (M_D, x, 1^{P_D(|x|)})$.

In the case of distributional-NP an analogous theorem is much harder to prove. The difficulty is that we have to reduce all DistNP problems (i.e., pairs consisting of decision problems and simple distributions) to one single distributional problem (i.e., Bounded Halting with a single simple distribution). If we apply reductions as above (and consider the induced probability distributions), then we will end up with many distributional versions of Bounded Halting. Furthermore the corresponding distribution functions will be very different and will not necessarily dominate one another. Instead, one should reduce a distributional problem, $(D, \mu)$, with an arbitrary P-computable distribution to a distributional problem with a fixed (P-computable) distribution (e.g. $\Pi_{BH}$). The difficulty in doing so is that the reduction should have the domination property.

Consider, for example, an attempt to reduce each problem in DistNP to $\Pi_{BH}$ by using the standard transformation of $D$ to $BH$ (sketched above). This transformation fails when applied to distributional problems in which the distribution of (infinitely many) strings is much higher than the distribution assigned to them by the uniform distribution. In such cases, the standard reduction maps an instance $x$ having probability mass $\mu'(x) \gg 2^{-|x|}$ to a triple $(M_D, x, 1^{P_D(|x|)})$ with much lighter probability mass (since $\mu'_{BH}(M_D, x, 1^{P_D(|x|)}) < 2^{-|x|}$). This violates the domination condition, and thus an alternative reduction is required.

The key to the alternative reduction is an (efficiently computable) encoding of strings taken from an *arbitrary* polynomial-time computable distribution by strings that have comparable probability mass under a *fixed* distribution (i.e., the uniform one). Specifically, this encoding will map $x$ into a codeword of length that is at most the logarithm of $1/\mu'(x)$, which means that under the uniform distribution the codeword of $x$ has probability weight approximately $\mu'(x)$. Accordingly, the reduction will map $x$ to a triple $(M_{D,\mu}, x', 1^{|x|^{O(1)}})$, where $|x'| < \log_2(1/\mu'(x)) + O(1)$, and $M_{D,\mu}$ is a non-deterministic Turing machine that first retrieves $x$ from $x'$, and then applies the standard non-deterministic machine (i.e., $M_D$) of the problem $D$. Such a reduction will be shown to satisfy all

three conditions (i.e. efficiency, validity, and domination). Thus, instead of forcing the structure of the original distribution $\mu$ on the target distribution $\mu_{BH}$, the reduction will incorporate the structure of $\mu$ into the the reduced instance (i.e., in $M_{D,\mu}$). The following technical lemma is the basis of the reduction.

**Coding Lemma:** *Let $\mu$ be a polynomial-time computable distribution function. Then there exist a coding function $C_\mu$ satisfying the following three conditions.*

Compression: *For every $x$, it holds that*

$$|C_\mu(x)| \leq \min\left\{|x|, \log_2 \frac{1}{\mu'(x)}\right\} + 1.$$

Efficient Encoding: *The function $C_\mu$ is computable in polynomial-time.*
Unique Decoding: *The function $C_\mu$ is one-to-one (i.e., $C_\mu(x) = C_\mu(x')$ implies $x = x'$).*

**Proof:** The function $C_\mu$ is defined as follows. If $\mu'(x) \leq 2^{-|x|}$, then $C_\mu(x) = 0x$ (i.e., in this case $x$ serves as its own encoding). Otherwise, if $\mu'(x) > 2^{-|x|}$, then $C_\mu(x) = 1z$, where $z$ is the longest common prefix of the binary expansions of $\mu(x-1)$ and $\mu(x)$; that is, if $\mu(x-1)$ and $\mu(x)$ have binary expansions $0.\sigma_1\sigma_2\cdots$ and $0.\tau_1\tau_2\cdots$, respectively, then $z = \sigma_1\cdots\sigma_\ell$ such that $\sigma_1\cdots\sigma_\ell = \tau_1\cdots\tau_\ell$ whereas $\sigma_{\ell+1} = 0$ and $\tau_{\ell+1} = 1$ (e.g., if $\mu(1010) = 0.10000$ and $\mu(1011) = 0.10101111$, then $C_\mu(1011) = 1z$ with $z = 10$). Consequently, $0.z1$ is in the interval $(\mu(x-1), \mu(x)]$; that is, $\mu(x-1) < 0.z1 \leq \mu(x)$.

We now verify that $C_\mu$ satisfies the conditions of the lemma. We start with the compression condition. Clearly, if $\mu'(x) \leq 2^{-|x|}$, then $|C_\mu(x)| = 1+|x| \leq 1+\log_2(1/\mu'(x))$. On the other hand, suppose that $\mu'(x) > 2^{-|x|}$ and let $z = z_1\cdots z_\ell$ be as above (i.e., the longest common prefix of the binary expansions of $\mu(x-1)$ and $\mu(x)$). Then,

$$\mu'(x) \;=\; \mu(x) - \mu(x-1) \;\leq\; \left(\sum_{i=1}^{\ell} 2^{-i} z_i + \sum_{i=\ell+1}^{\mathrm{poly}(|x|)} 2^{-i}\right) - \sum_{i=1}^{\ell} 2^{-i} z_i \;<\; 2^{-|z|}$$

and $|z| \leq \log_2(1/\mu'(x))$ follows. Thus, $|C_\mu(x)| \leq 1+\log_2(1/\mu'(x))$ in both cases, and if $\log_2(1/\mu'(x)) \geq |x|$ (i.e., $\mu'(x) \leq 2^{-|x|}$), then $|C_\mu(x)| = |x|+1$. Clearly, $C_\mu$ can be computed in polynomial-time (by computing $\mu(x-1)$ and $\mu(x)$). Finally, note that $C_\mu$ is one-to-one by considering the two cases (i.e., $C_\mu(x) = 0x$ and $C_\mu(x) = 1z$), while using the fact that $\mu(x-1) < 0.z1 \leq \mu(x)$ (in the second case ). □

*Towards the reduction.* For every distributional problem $(D, \mu)$ in DistNP, we introduce a non-deterministic machine $M_{D,\mu}$ that will be used in the following reduction of $(D, \mu)$ to $\Pi_{BH} = (BH, \mu_{BH})$ such that all instances (of $D$) are mapped to triples with first element $M_{D,\mu}$. The description of this machine, $M_{D,\mu}$, refers to the aforementioned coding function $C_\mu$ (as well as to the non-deterministic machine $M_D$ associated with $D$). On input $y = C_\mu(x)$, machine

$M_{D,\mu}$ computes $D(x)$, by first retrieving $x$ from $C_\mu(x)$ (e.g., guess and verify), and next running the non-deterministic polynomial-time machine (i.e., $M_D$) that solves $D$.

*The reduction itself.* An instance $x$ (of $D$) is mapped by the reduction to the triple $(M_{D,\mu}, C_\mu(x), 1^{P(|x|)})$, where $P(n) \stackrel{\text{def}}{=} P_D(n) + P_C(n) + n$ such that $P_D(n)$ is a polynomial bounding the running time of $M_D$ on (acceptable) inputs of length $n$, and $P_C(n)$ is a polynomial bounding the running time of an algorithm for encoding inputs (of length $n$).

**Proposition:** *The foregoing mapping constitutes a reduction of* $(D, \mu)$ *to* $(BH, \mu_{BH})$.

**Proof:** We verify the three requirements from a reduction.

- The transformation can be computed in polynomial-time.
  (Indeed, we rely on the fact that $C_\mu$ is polynomial-time computable.)
- By construction of $M_{D,\mu}$ it follows that $D(x) = 1$ if and only if there exists a computation of machine $M_{D,\mu}$ that on input $C_\mu(x)$ halts outputting 1 within $P(|x|)$ steps.
  (Recall that, on input $C_\mu(x)$, machine $M_{D,\mu}$ non-deterministically guesses $x$, verifies in $P_C(|x|)$ steps that $x$ is encoded by $C_\mu(x)$, and non-deterministically "computes" $D(x)$.)
- To see that the distribution induced by the reduction is dominated by the distribution $\mu_{BH}$, we first recall that the transformation $x \to C_\mu(x)$ is one-to-one. It suffices to consider instances of $BH$ which have a preimage under the reduction (since instances with no preimage satisfy the condition trivially). All these instances are triples with first element $M_{D,\mu}$. By the definition of $\mu_{BH}$

$$\mu'_{BH}(M_{D,\mu}, C_\mu(x), 1^{P(|x|)}) = c \cdot \frac{1}{P(|x|)^2} \cdot \frac{1}{|C_\mu(x)|^2 \cdot 2^{|C_\mu(x)|}} \tag{1}$$

  where $c = \frac{1}{|M_{D,\mu}|^2 \cdot 2^{|M_{D,\mu}|}}$ is a constant depending only on $(D, \mu)$.
  By virtue of the Coding Lemma it holds that

$$\mu'(x) \leq 2 \cdot 2^{-|C_\mu(x)|}. \tag{2}$$

  Combing Eq. (1) and (2), we get

$$\begin{aligned} \mu'_{BH}(M_{D,\mu}, C_\mu(x), 1^{P(|x|)}) &\geq c \cdot \frac{1}{P(|x|)^2} \cdot \frac{1}{|C_\mu(x)|^2} \cdot \frac{\mu'(x)}{2} \\ &> \frac{c}{2 \cdot |M_{D,\mu}, C_\mu(x), 1^{P(|x|)}|^2} \cdot \mu'(x). \end{aligned}$$

The proposition follows. □

## 4 Conclusions

In general, a theory of average case complexity should provide

1. a specification of a broad class of *interesting* distributional problems;
2. a definition capturing the subclass of (distributional) problems that are *easy* on the average;
3. notions of reducibility that allow to infer the easiness of one (distributional) problem from the easiness of another;
4. and, of course, results...

It seems that the theory of average case complexity, initiated by Levin and further developed in [9, 22, 1, 13, 21], satisfies these expectations to some extent. Following is my evaluation regarding its "performance" with respect to each of the above.

1. The scope of the theory, originally restricted to P-computable distributions, has been significantly extended to cover all P-sampleable distributions (as suggested in [1]). The key result here is by Impagliazzo and Levin [13] whow proved that every language which is ⟨NP, P-computable⟩-complete is also ⟨NP, P-samplable⟩-complete. This important result makes the theory of average case very robust: It allows to reduce distributional problems from an utmost wide class to distributional problems with very restricted/simple type of distributions.
   Till Livne's result [21], my feeling was that the set of P-computable distributions may be too restricted in the sense that it may not allow to present DistNP-complete problems that refer to most natural NP decision problems. However, Livne showed that all natural NP decision problems do have distributional versions that are DistNP-complete, alas these versions turned out to be somewaht unnatural (i.e., the distribution does not seem simple and/or natural).[2]
2. The definition of average polynomial-time does seem strange at first glance, but it seems that it (or a similar alternative) does captures the intuitive meaning of "easy on the average".
   We mention that an alternative notion that refers to the *typical* running time rather than the *average* of (a quantity that is polynomially related to) the running time is considered in [7, Sec. 10.2.1] and [8]. Specifically, we may say that $f : \{0,1\}^* \to \mathbb{N}$ is typically polynomial with respect to a distribution $\mu$ if there exists a positive polynomial $p$ such that, for every polynomial $q$, it holds that
   $$\sum_{x \in \{0,1\}^* : f(x) > p(|x|)} q(|x|) \cdot \mu'(x) < \infty.$$
3. The notions of reducibility are both natural and adequate.

[2] For a discussion of the notion of a natural problem, the interested reader is referred to Appendix B.

4. Results did follow, but here indeed much more is expected. Currently, quite natural DistNP-complete problems are known for the following areas: Computability (e.g., Bounded-Halting) [9], Combinatorics (e.g., Tiling [18] and a generalization of graph coloring [22]), Formal Languages (cf., [9, 4]), and Algebra (e.g., of matrix groups [10]). Furthermore, the aforementioned result of Livne [21] asserts that all natural NP problems have DistNP-complete versions. However, the challenge of finding a really natural distributional problem (e.g., subset sum with uniform distribution) that is complete in DistNP has not been met so far. It seems that what is still lacking are techniques for design of "distribution preserving" reductions.

In addition to their central role in the theory of average-case complexity, reductions that preserve uniform (or very simple) instance distribution are of general interest. Such reductions, unlike most known reductions used in the theory of NP-completeness, cannot map all instances to some "pathological" subcase.

Levin views the results in his paper [18] as an indication that all P-computable distributions are in fact related (or similar). Additional support to this statment is provided by his latter work [20].

## Acknowledgements

I'm very grateful to Leonid Levin for many inspiring discussions. Special thanks also to Noam Livne for reconfiguring my views regarding P-computable distributions.

## Appendix A: Failure of a Naive Formulation

When asked to motivate his definition of average polynomial-time, Leonid Levin replies, non-deterministically, in one of the following three ways:

1. "This is *the* natural definition".
2. "This definition is *not important* for the results in my paper; only the definitions of reduction and completeness matter (and even they can be modified in many ways while preserving the results)".
3. "Any definition that *makes sense* is either equivalent or weaker".

For further elaboration on the first argument the reader is referred to Leonid Levin. The second argument is, off course, technically correct but unsatisfactory. We will need a definition of "easy on the average" when motivating the notion of a reduction and developing useful relaxations of it. The third argument is a thesis, which should be interpreted along Wittgenstein's suggestion to the teacher: "say nothing and restrict yourself to pointing out errors in the students' attempts to say something". We will follow this line of argument here by showing that the definition that seems natural to an average computer scientist suffers from serious problems and should be rejected.

**Definition X** (naive formulation of the notion of easy on the average): *A distributional problem $(D, \mu)$ is polynomial-time on the average if there exists an algorithm $A$ solving $D$ (i.e., on input $x$ outputs $D(x)$) such that the running time of algorithm $A$, denoted $t_A$, satisfies*

$$\exists c > 0 \; \forall n \qquad \sum_{x \in \{0,1\}^n} \mu'_n(x) \cdot t_A(x) < n^c \tag{3}$$

*where $\mu'_n(x)$ is the conditional probability that $x$ occurs given that an $n$-bit string occurs* (i.e., $\mu'_n(x) = \mu'(x)/\sum_{y \in \{0,1\}^n} \mu'(y)$).

*The main problem with Definition X.* The problem that we consider most upsetting is that *Definition X is not robust under functional composition of algorithms.* Namely, if the distributional problem $A$ can be solved in average polynomial-time given access to an oracle for $B$, and problem $B$ can be solved in polynomial-time, then it does *not* follow that the distributional problem $A$ can be solved in average polynomial-time.

For example, consider the uniform probability distribution (on inputs of each length) and an oracle Turing machine $M$ that solves $A$ when given access to oracle $B$. Suppose that $M$ runs for $2^{\frac{n}{2}}$ steps on $2^{\frac{n}{2}}$ of the inputs of length $n$, and $n^2$ steps on all other inputs of length $n$. Furthermore, supposed that when $M$ makes $t$ steps, it asks a single query of length $\sqrt{t}$. Note that machine $M$ is polynomial-time on the average. But now suppose that the algorithm for $B$ has cubic running-time. The reader can verify that, although $M$ itself (when given access to the oracle $B$) is polynomial-time on the average, combining $M$ with the cubic running-time algorithm for $B$ *does not* yield an algorithm that is polynomial-time on the average according to Definition X. It is easy to see that this problem does not arise when using the definition presented in Section 2.

The source of the above problem with Definition X is the fact that the definition of polynomial-on-the-average that underlies it is not closed under application of polynomials. Namely, if $t : \{0,1\}^* \to \mathbb{N}$ is polynomial on the average (according to Eq. (3)), with respect to some distribution, it does not follow that also $t^2(\cdot)$ is polynomial on the average (with respect to the same distribution).

The foregoing technical problem is also the source of the following problem, which Levin considers most upsetting: Definition X is *not* machine independent. This is the case because some of the simulations of one computational model on another square the running time (e.g., the simulation of two-tape Turing machines on a one-tape Turing machine, or the simulation of a RAM (Random Access Machine) on a Turing machine).

Having pointed out several weaknesses of Definition X, let us also doubt its "clear intuitive advantage" over the definition presented in Section 2. Definition X is derived from the formulation of worst case polynomial-time algorithms, which requires that $\exists c > 0 \; \forall n$ such that

$$\forall x \in \{0,1\}^n \qquad t_A(x) \; < \; n^c. \tag{4}$$

Definition X was derived by applying the expectation operator to the Eq. (4), But why not make a very simple algebraic manipulation of Eq. (4) before applying

the expectation operator? How about taking the $c$-th root of both sides and dividing by $n$; indeed, this yields that $\exists c > 0\ \forall n$ it holds that

$$\forall x \in \{0,1\}^n \qquad \frac{t_A(x)^{\frac{1}{c}}}{n} < 1. \tag{5}$$

But now, applying the expectation operator to the Eq. (5) leads to the definition presented in Section 2...

We conclude that both Definition X and the definition presented in Section 2 are obtained by applying the expectation operator to a worst-case inequality, where the two base inequalities (i.e., Eq. (4) and Eq. (5)) are easily related to one another. From this perspective it is hard to argue (a priori) that one application is more natural than another. However, a posteriori, it becomes evident that the definition presented in Section 2 demonstrates a better understanding of the effect of the expectation operator with respect to complexity measures.

*Summary:* Robustness under functional composition as well as machine independence seems to be essential for a coherent theory. These are among the primary reasons for the acceptability of P as capturing problems that can be solved efficiently. In going from worst case analysis to average case analysis we should not and would not like to lose these properties.

## Appendix B: On the Notion of Natural Problems

Throughout this article, we made several references to the undefined notion of a natural computational problem. While most researchers have some intuition regarding this notion, we feel that an attempt to articulate this notion is in place.

We comment that one should not expect to see a formal definition of intuitive notions such as "simple" or "natural"; yet, this does not mean that we should not try to articulate our intuition about them.

The first idea that comes to mind is to say that a problem is natural if most researchers would say so. This empirically oriented definition seems workable, but it leaves us wondering as to what makes some problems natural whereas other problems are not natural; that is, why would most researchers agree on the foregoing classification of problems?

An appealing criterion was proposed by Livne [21]: *The extent to which a computational problem is natural, with respect to some result, is proportional to the amount of references to the said problem that are prior to the said result and occur in a different context.* Thus, for example, Satisfiability is a very natural problem with respect to the Cook-Levin Theorem [2, 17], because this problem was defined and studied in numerous works and in different contexts (such as logic) prior to the Cook-Levin Theorem. To the contrary, the sequence of decision problems constructed in the proof of the Hierarchy Theorem of [12] is definitely unnatural, because these decision problems were first defined in this context (let alone that they are never mentioned outside the context of the Hierarchy Theorem).

## References


1. S. Ben-David, B. Chor, O. Goldreich, and M. Luby, "On the Theory of Average Case Complexity", *Journal of Computer and system Sciences*, Vol. 44, No. 2, pages 193–219, 1992.
2. S.A. Cook, "The Complexity of Theorem Proving Procedures", *Proc. 3rd ACM Symp. on Theory of Computing*, pages 151–158, 1971.
3. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.
4. O. Goldreich. Towards a Theory of Average Case Complexity (a survey). TR-531, Computer Science Department, Technion, Haifa, Israel, March 1988.
5. O. Goldreich. *Foundation of Cryptography: Basic Tools*. Cambridge University Press, 2001.
6. O. Goldreich. *Foundation of Cryptography: Basic Applications*. Cambridge University Press, 2004.
7. O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.
8. O. Goldreich. Average Case Complexity, Revisited. This volume.
9. Y. Gurevich, "Complete and Incomplete Randomized NP Problems", *Proc. of the 28th IEEE Symp. on Foundation of Computer Science*, pages 111–117, 1987.
10. Y. Gurevich, "Matrix Decomposition Problem is Complete for the Average Case", *Proc. of the 31st IEEE Symp. on Foundation of Computer Science*, pages 802–811, 1990.
11. Y. Gurevich and D. McCauley, "Average Case Complete Problems", preprint, 1987.
12. J. Hartmanis and R.E. Stearns. On the Computational Complexity of of Algorithms. *Transactions of the AMS*, Vol. 117, pages 285–306, 1965.
13. R. Impagliazzo and L.A Levin, "No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random", *Proc. of the 31st IEEE Symp. on Foundation of Computer Science*, pages 812–821, 1990.
14. D.S. Johnson, "The NP-Complete Column – an ongoing guide", *Jour. of Algorithms*, 1984, Vol. 4, pages 284–299.
15. R.M. Karp, "Reducibility among Combinatorial Problems", *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher (eds.), Plenum Press, pages 85–103, 1972.
16. R.M. Karp, "Probabilistic Analysis of Algorithms", manuscript, 1986.
17. L.A Levin, "Universal Search Problems", *Problemy Peredaci Informacii 9*, pages 115–116, 1973. Translated in *problems of Information Transmission 9*, pages 265–266.
18. L.A. Levin, "Average Case Complete Problems", *SIAM Jour. of Computing*, Vol. 15, pages 285–286, 1986. Extended abstract appeared in *16th STOC*, 1984.
19. L.A. Levin, "One-Way Function and Pseudorandom Generators", *Proc. 17th ACM Symp. on Theory of Computing*, pages 363–365, 1985.
20. L.A. Levin, "Homogeneous Measures and Polynomial Time Invariants", *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 36–41, 1988.
21. N. Livne. All Natural NPC Problems Have Average-Case Complete Versions. *Computational Complexity*, to appear. Preliminary version in *ECCC*, TR06-122, 2006.
22. R. Venkatesan and L.A Levin, "Random Instances of a Graph Coloring Problem are Hard", *Proc. 20th ACM Symp. on Theory of Computing*, pages 217–222, 1988.