Notes on Levin's Theory of Average-Case Complexity

Oded Goldreich*
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.
E-mail: `oded@wisdom.weizmann.ac.il`

November 1997

Abstract

In 1984, Leonid Levin has 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.

Acknowledgement and Warning: Much of the text was reproduced from expositionary material contained in [1], which in turn was based on [4]. Thus, much of the technical exposition is 10 years old; I would have written some things differently today.

*Written while visting LCS, MIT.

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 [11, 13]. However, this line of research has so far been applicable only to specific algorithms and with respect to specific, typically uniform, probability distributions.

The general question of average case complexity was addressed for the first time by Levin [15]. 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 [14]) works are the basis for the theory of NP-completeness. Subsequent works [7, 18, 8] have provided few 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 DistNP$\stackrel{\rm def}{=}\langle$NP, P-computable$\rangle$, defined by Levin [15], 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 Π_1 is reducible to Π_2 and Π_2 is in average polynomial-time, then so is Π_1. Finally, it is shown that the class DistNP contains a complete problem.

Levin's average-case theory, revisited. Levin's laconic presentation [15] hides the fact that choices has been done in the development of the average-case complexity theory. We discuss some of this choices here. Firstly, one better think of the motivation as to provide a theory of efficient computation (as suggested above), rather than a theory of infeasible ones (e.g., as in Cryptography). We note that a theory of useful-for-cryptography infeasible computations does exist (cf., [5, 6]). A key difference is that in Cryptography we needs problems for which one may generate instance-solution pairs so that solving the problem given only the instance is hard. In the theory of average-case complexity considered below, we consider problems which are hard to solve, but 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", but it is not clear if they exhaust all natural "simple" distributions. A much wider class, which is easier to defend, is the class of all distributions having an efficient algorithm for generting instances (according to the distribution). One may argue that the instances of any problem we may need to solve are generated efficiently by some process, and so the latter class of P-samplable distribution suffices for our theory [1]. Fortunately, it was show [10] that any distributional problem which 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 which, in our opinion, deem it inadequate. (For a detailed discussion of this point, the reader is referred to the Appendix.) 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 84], but the reader is advised to look for further explanations and motivating discussions elsewhere (e.g., [11, 9, 4]).

For sake of simplicity, we consider the standard lexicographic ordering of binary strings. Any fixed efficient enumeration will do. (An *efficient enumeration* is a 1-1 and onto mapping of strings to integers which can be computed and inverted in polynomial-time.) 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. encoding).

Definition 1 (Probability Distribution Function): *A* distribution function $\mu : \{0,1\}^* \to [0,1]$ *is a non-decreasing function from strings to the unit interval [0,1] which 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* μ *is denoted* μ' *and 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 $c \neq 1$. In all the cases where we use this convention it is easy to normalize the distribution, so that it converges to one. An important example is the *uniform* distribution function μ_0 defined as $\mu_0'(x) = \frac{1}{|x|^2} \cdot 2^{-|x|}$. (A minor modification which does converge to 1 is obtained by letting $\mu_0'(x) = \frac{1}{|x|\cdot(|x|+1)} \cdot 2^{-|x|}$.)

Definition 2 (A Distributional Problem): *A* distributional decision problem (resp., distributional search problem) *is a pair* (D,μ) *(resp.* (S,μ)*), 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].

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

It follows that the binary expansion of $\mu(x)$ has length polynomial in $|x|$. An 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 big 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 μ (as $\sum_x \mu'(x) \cdot t(|x|) = \sum_n 2^{-n} = 1$).

If the distribution function μ is in P-computable then the density function, μ', is computable in time polynomial in $|x|$. The converse, however, is false, unless P = NP (see [9]). 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 which corresponds to (the traditional) NP. Most of results in the literature refer to this class.

Definition 4 (The class DistNP): *A distributional problem (D, μ) belongs to the class* DistNP *if D is an NP-predicate and μ 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 μ 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 if 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. It is important to point out that the naive formalizations of these definitions 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.

Definition 5 (Polynomial on the Average): *A function $f : \{0,1\}^* \rightarrow \mathbf{N}$ is* polynomial on the average *with respect to a distribution μ 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. μ.*

Thus, a function is polynomial on the average if it is bounded by a polynomial in a function which is linear on the average. In fact, the basic definition is that of a function which is linear on the average; see [1, Def. 2].

Definition 6 (The class Average-P): *A distributional problem (D, μ) 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 μ.*

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 (cf., [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 Turing Reductions): *We say that the probabilistic oracle Turing machine M* randomly reduces *the distributional problem (D_1, μ_1) to the distributional problem (D_2, μ_2) if the following three conditions hold.*

1) Efficiency: *Machine M is polynomial time on the average taken over x with distribution μ_1 and the internal coin tosses of M with uniform probability distribution (i.e., let $t_M(x,r)$ be the running time of M on input x and internal coin tosses r, then 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 μ_0 is the uniform distribution).*

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

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

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

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

$$\mu_2'(y) \geq \frac{1}{|y|^c} \cdot \sum_{x \in \{0,1\}^*} Ask_M(x,y) \cdot \mu_1'(x)$$

where $Ask_M(x,y)$ is the probability (taken over M's internal coin tosses) that "machine M asks query y on input x".

In the definition of deterministic Turing reductions $M^{D_2}(x)$ is determined by x (rather than being a random variable) and $Ask_M(x,y)$ is either 0 or 1 (rather than being any arbitrary rational in $[0,1]$). In case of a many-to-one deterministic reduction, for every x, we have $Ask_M(x,y) = 1$ for a unique y.

It can be proven that if (D_1, μ_1) is deterministically (resp., randomly) reducible to (D_2, μ_2) and if (D_2, μ_2) is solvable by a deterministic (resp., randomized) algorithm with running time polynomial on the average then so is (D_1, μ_1).

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.

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 8 (distributional Bounded Halting):

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

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

Note that μ'_{BH} is very different from the uniform distribution on binary strings (e.g., consider relatively large k). Yet, as noted by Levin, one can easily modify Π_{BH} so that has a "uniform" distribution and is DistNP-complete with respect to randomized reduction. (Hint: replace the unary time bound by a string of equal length, assigning each such string the same probability.)

3 DistNP-completeness of Π_{BH}

The proof, presented here, is due to Guretich [7] (an alternative proof is implied by Levin's original paper [15]).

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.

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 problem is to determine whether there exists a computation of M on input x which 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). Applying reductions as above we will end up with many distributional versions of Bounded Halting, and furthermore the corresponding distribution functions will be very different and will not necessarily dominate one another. Instead, one should reduce a distributional problem, (D,μ), with an arbitrary P-computable distribution to a distributional problem with a fixed (P-computable) distribution (e.g. Π_{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 Π_{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 (recall $\mu'_{BH}(M_D, x, 1^{P_D(|x|)}) < 2^{-|x|}$). 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 which have comparable probability mass under a fixed distribution. This encoding will map x into a code of length bounded above by the logarithm of $1/\mu'(x)$. Accordingly, the reduction will map x to a triple $(M_{D,\mu}, x', 1^{|x|^{O(1)}})$, where $|x'| < O(1) + \log_2 1/\mu'(x)$, and $M_{D,\mu}$ is a non-deterministic Turing machine which 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 μ on the target distribution μ_{BH}, the reduction will incorporate the structure of μ into the the reduced instance.
The following technical lemma is the basis of the reduction.

Coding Lemma: Let μ be a polynomial-time computable distribution function. Then there exist a coding function C_μ satisfying the following three conditions.

1) *Compression:* $\forall x$

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

2) *Efficient Encoding:* The function C_μ is computable in polynomial-time.

3) *Unique Decoding:* The function C_μ is one-to-one (i.e. $C_\mu(x) = C_\mu(x')$ implies $x = x'$).

Proof: The function C_μ 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). 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 $\mu(x)$ (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 $(\mu(x-1), \mu(x)]$ (i.e., $\mu(x-1) < 0.z1 \leq \mu(x)$).

We now verify that C_μ so defined 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. Clearly, C_μ can be computed in polynomial-time by computing $\mu(x-1)$ and $\mu(x)$. Finally, note that C_μ is one-to-one by considering the two cases, $C_\mu(x) = 0x$ and $C_\mu(x) = 1z$. (In the second case, use the fact that $\mu(x-1) < 0.z1 \leq \mu(x)$). $\square$

Using the coding function presented in the above proof, we introduce a non-deterministic machine $M_{D,\mu}$ so that the distributional problem (D, μ) is reducible to $\Pi_{BH} = (BH, \mu_{BH})$ in a way that all instances (of D) are mapped to triples with first element $M_{D,\mu}$. 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) which solves D.

The reduction maps an instance x (of D) to the triple $(M_{D,\mu}, C_\mu(x), 1^{P(|x|)})$, where $P(n) \stackrel{\mathrm{def}}{=} P_D(n) + P_C(n) + n$, $P_D(n)$ 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 above mapping constitutes a reduction of (D, μ) to (BH, μ_{BH}).

Proof: We verify the three requirements.

- The transformation can be computed in polynomial-time. (Recall that C_μ 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, 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 μ_{BH}, we first note 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 μ_{BH}
$$\mu'_{BH}(M_{D,\mu}, C_\mu(x), 1^{P(|x|)}) = c \cdot \frac{1}{P(|x|)^2} \cdot \frac{1}{|C_\mu(x)|^2 \cdot 2^{|C_\mu(x)|}}$$
where $c = \frac{1}{|M_{D,\mu}|^2 \cdot 2^{|M_{D,\mu}|}}$ is a constant depending only on (D, μ).
By virtue of the coding Lemma
$$\mu'(x) \leq 2 \cdot 2^{-|C_\mu(x)|}$$
It thus follows that
$$\begin{aligned} \mu'_{BH}(M_{D,\mu}, C_\mu(x), 1^{P(|x|)}) &\geq c \cdot \frac{1}{P(|x|)^2} \cdot \frac{1}{|C_\mu(x)|^2} \cdot \frac{\mu'(x)}{2} \\ &> \frac{c}{2 \cdot |M_{D,\mu}, C_\mu(x), 1^{P(|x|)}|^2} \cdot \mu'(x) \end{aligned}$$

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 which are *easy* on the average;
3. notions of reducibility which 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 [7, 18, 1, 10], 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 [10] 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.

2. The definition of average polynomial-time does seem strange at first glance, but it seems that it (or similar alternative) does captures the intuitive meaning of “easy on the average”.

3. The notions of reducibility are both natural and adequate.

4. Results did follow, but here indeed much more is expected. Currently, DistNP-complete problems are known for the following areas: Computability (e.g., Bounded-Halting) [7], Combinatorics (e.g., Tiling [15] and a generalization of graph coloring [18]), Formal Languages (cf., [7, 4]), and Algebra (e.g., of matrix groups [8]). However the challenge of finding a really natural distributional problem which is complete in DistNP (e.g., subset sum with uniform distribution), 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 which preserve uniform (or very simple) instance distribution are of general interest. Such reductions, unlike most known reductions used in the theory of NP-completeness, have a range which is a non-negligible part of the set of all possible instances of the target problem (i.e. a part which cannot be claim to be only a “pathological subcase”).

Levin views the results in his paper [15] as an indication that all “simple” (i.e., P-computable) distributions are in fact related (or similar). Additional support to this statment is provided by his latter work [17].

Acknowledgements

I'm very grateful to Leonid Levin for many inspiring discussions.

Appendix: 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:

- "This is *the* natural definition".
- "This definition is *not important* for the results in my paper; only the definitions of reduction and completeness matter (and also they can be modified in many ways preserving the results)".
- "Any definition which *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 here by arguing that the definition which 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,μ) 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 $\exists c>0 \forall n$:*

$$\sum_{x\in\{0,1\}^n} \mu'_n(x)\cdot t_A(x) < n^c$$

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 problem which we consider to be 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 uniform probability distribution on inputs of each length and an oracle Turing machine M which given access to oracle B solves A. Suppose that M^B runs $2^{\frac{n}{2}}$ steps on $2^{\frac{n}{2}}$ of the inputs of length n, and n^2 steps on all other inputs of length n; and furthermore that M when making t steps asks a single query of length $\sqrt{t}$. (Note that machine M, given access to oracle for B, is polynomial-time on the average.) Finally, suppose that the algorithm for B has cubic running-time. The reader can now verify that although M 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 which 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 underlying definition of polynomial-on-the-average is not closed under application of polynomials. Namely, if $t:\{0,1\}^*\to \mathbb{N}$ is polynomial on the average, 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). This technical problem is also the source of the following problem, that Levin considers most upsetting: Definition X is *not* machine independent. This is the case since 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).

Another two problems with Definition X have to do with the fact that it deals separately with inputs of different length. The first problem is that Definition X is very dependent on the particular encoding of the problem instance. Consider, for example, a problem on simple undirected graphs for which there exist an algorithm A with running time $t_A(G) = f(n,m)$, where n is the number of vertices in G and m is the number of edges (in G). Suppose that if $m < n^{\frac{3}{2}}$ then $f(n,m) = 2^n$ and else $f(n,m) = n^2$. Consider the distributional problem which consists of the above graph problem with the uniform probability distribution on all graphs with the same number of vertices. Now, if the graph is given by its (incident) matrix representation then Definition X implies that A solves the problem in average polynomial-time (the average is taken on all graphs with n nodes). On the other hand, if the graphs are represented by their adjacency lists then the modified algorithm A (which transforms the graphs to matrix representation and applies algorithm A) is judged by Definition X to be non-polynomial on the average (here the average is taken over all graphs of m edges). This of course will not happen when working with the definition presented in Section 2. The second problem with dealing separately with different input lengths is that it does not allow one to disregard inputs of a particular length. Consider for example a problem for which we are only interested in the running-time on inputs of odd length.

After pointing 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 $\exists c > 0\ \forall n$:

$$\forall x \in \{0,1\}^n \ : \quad t_A(x) < n^c$$

Definition X was derived by applying the expectation operator to the above inequality. But why not make a very simple algebraic manipulation of the inequality before applying the expectation operator? How about taking the c-th root of both sides and dividing by n; this yields $\exists c > 0\ \forall n$:

$$\forall x \in \{0,1\}^n \ : \quad \frac{t_A(x)^{\frac{1}{c}}}{n} < 1$$

Applying the expectation operator to the above inequality leads to the definition presented in Section 2... We believe that this definition 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. So is robustness under efficiently effected transformation of problem encoding. These are one of the primary reasons for the acceptability of P as capturing problems which 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.

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, April 1992, pp. 193–219.

[2] Cook, S.A., "The Complexity of Theorem Proving Procedures", *Proc. 3rd ACM Symp. on Theory of Computing*, pp. 151–158, 1971.

[3] Garey, M.R., and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[4] Goldreich, O., "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 – Fragments of a Book.* February 1995. Available from `http://theory.lcs.mit.edu/~oded/frag.html`.

[6] O. Goldreich. On the Foundations of Modern Cryptography (essay). Proceedings of *Crypto97*, Springer LNCS, Vol. 1294, pp. 46–74.

[7] Gurevich, Y., "Complete and Incomplete Randomized NP Problems", *Proc. of the 28th IEEE Symp. on Foundation of Computer Science*, 1987, pp. 111–117.

[8] Gurevich, Y., "Matrix Decomposition Problem is Complete for the Average Case", *Proc. of the 31st IEEE Symp. on Foundation of Computer Science*, 1990, pp. 802-811.

[9] Gurevich, Y., and D. McCauley, "Average Case Complete Problems", preprint, 1987.

[10] Impagliazzo, R., 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*, 1990, pp. 812–821.

[11] Johnson, D.S., "The NP-Complete Column – an ongoing guide", *Jour. of Algorithms*, 1984, Vol. 4, pp. 284–299.

[12] Karp, R.M., "Reducibility among Combinatorial Problems", *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher (eds.), Plenum Press, pp. 85–103, 1972.

[13] Karp, R.M., "Probabilistic Analysis of Algorithms", manuscript, 1986.

[14] Levin, L.A., "Universal Search Problems", *Problemy Peredaci Informacii 9*, pp. 115–116, 1973. Translated in *problems of Information Transmission 9*, pp. 265–266.

[15] Levin, L.A., "Average Case Complete Problems", *SIAM Jour. of Computing*, 1986, Vol. 15, pp. 285–286. Extended abstract appeared in *Proc. 16th ACM Symp. on Theory of Computing*, 1984, p. 465.

[16] Levin, L.A., "One-Way Function and Pseudorandom Generators", *Proc. 17th ACM Symp. on Theory of Computing*, 1985, pp. 363–365.

[17] Levin, L.A., "Homogeneous Measures and Polynomial Time Invariants", *Proc. 29th IEEE Symp. on Foundations of Computer Science*, 1988, pp. 36–41.

[18] Venkatesan, R., and L.A. Levin, "Random Instances of a Graph Coloring Problem are Hard", *Proc. 20th ACM Symp. on Theory of Computing*, 1988, pp. 217–222.