On average polynomial time*

R. Schuler
Theoretische Informatik, Universität Ulm, D-89069 Ulm, Germany
email: rsc@informatik.uni-ulm.de

Keywords: computational complexity, average-case analysis.

Abstract

One important reason to consider average case complexity is whether all, or at least some significant part of NP can be solved efficiently on average for all reasonable distributions. Let $P_{P\text{-comp}}$ be the class of problems, which are solvable in average polynomial-time for every polynomial-time computable input distribution. Following the framework of Levin [7, 8] the above question can now be formulated as: Is NP included in $P_{P\text{-comp}}$? In this paper, it is shown that $P_{P\text{-comp}}$ contains P as a proper subset. Thus, even if $P \neq NP$, it is still possible that $NP \subseteq P_{P\text{-comp}}$. As a further step it is shown that $P_{P\text{-comp}}$ is different from NP. Therefore, either NP is not solvable efficiently on average, or (since $P_{P\text{-comp}} \subseteq E$) $E \not\subseteq NP$ and hence NP is a proper subset of Exp.

1 Introduction

Recently, "average-case complexity" has received considerable attention by researchers in several fields of computer science. Even if a problem is not (or may not be) solvable efficiently in the worst-case, it may be solvable efficiently on average. Indeed, several results have been obtained that show even simple algorithms work well on average (see e.g. [4, 5]). However, most of these results are about concrete problems, and not so much has been done for a more general study of average-case complexity. In particular, the following general question is important: Is every NP set decidable in polynomial time on average?

*This work was supported by the Deutsche Forschungsgemeinschaft, Grant Scho 302/4-1

Levin [7, 8] established a reasonable framework for discussing the above question formally. In this paper, we follow his framework.

When discussing average-case polynomial-time computability, one should be careful about the definition of "polynomial on average", because a natural but naive one is not appropriate for several reasons (see Gurevich [3] for a detailed discussion). Levin's definition for polynomial on average is robust and still natural.

In each concrete case, it may be reasonable to fix an input distribution. In a general discussion, however, we should not assume any fixed distribution. On the other hand, it may not be so realistic to discuss polynomial-time computability considering *any* input distribution. In this setting average-case complexity equals worst-case complexity [10, 9]. Levin proposed to consider polynomial-time computable distributions (in short, P-computable distributions) as "realistic" distributions.

Thus, in Levin's framework, the above questions is formally stated as follows: Is it the case that for every NP set L and every P-computable distribution μ, L is polynomial-time decidable on average (in Levin's sense) when the input is given under μ.

In [11], Schuler and Yamakami defined a class $\mathrm{P}_{\text{P-comp}}$. This is the class of sets that are decidable in polynomial-time on average under any P-computable distribution. Using this notation, the above question can be restated as follows: Does $\mathrm{NP} \subseteq \mathrm{P}_{\text{P-comp}}$ hold? Thus, in order to study this general question it would be helpful to know the structure of the class $\mathrm{P}_{\text{P-comp}}$.

In this paper we prove the following properties of $\mathrm{P}_{\text{P-comp}}$.

1. $\mathrm{P} \subsetneq \mathrm{P}_{\text{P-comp}} \subsetneq \mathrm{E}$
2. $\mathrm{NP} \neq \mathrm{P}_{\text{P-comp}}$

The first result (i.e. $\mathrm{P} \subsetneq \mathrm{P}_{\text{P-comp}}$) means that "average polynomial-time computability under P-computable distributions" is an essential generalization of "worst-case polynomial-time computability". Thus, even if $\mathrm{P} \neq \mathrm{NP}$, we may still have some hope that $\mathrm{NP} \subseteq \mathrm{P}_{\text{P-comp}}$, even if this implies NE=E [1]. The second result means that either $\mathrm{NP} \subsetneq \mathrm{P}_{\text{P-comp}}$, or $\mathrm{P}_{\text{P-comp}} \subsetneq \mathrm{NP}$ (or both). Notice that the former is the negative answer to our original question, and the latter implies that $\mathrm{E} \not\subseteq \mathrm{NP}$ and hence $\mathrm{NP} \subsetneq \mathrm{Exp}$.

2 Preliminaries

In the following let $\mu' : \Sigma^* \to [0,1]$ be a density function, μ be the corresponding

distribution, where $\mu(x) = \sum_{y \leq x} \mu'(y)$, and $\mu'_n(x) = \mu'(x)/\sum_{|y|=n} \mu'(y)$ be the conditional probability of an input x of length n. Given a set X, $\mu'(X)$ is defined as $\mu'(X) = \sum_{x \in X} \mu'(x)$. μ is a standard distribution, if for some constants k and c: $\mu'(x) = (c/|x|^k)2^{-|x|}$. A finite string $d_1 d_2 \cdots d_m$, i.e. an output of a transducer, is identified with the number $d_1 \cdot 2^{-1} + d_2 \cdot 2^{-2} + \cdots + d_m \cdot 2^{-m} \in [0,1]$.

A distribution μ is polynomial (exponential) time computable if there exists a transducer T such that for all x and all i: $|T(x, 1^i) - \mu(x)| \leq 2^{-i}$ and T is polynomial (exponential) in i and $|x|$ [6]. Let P-comp (E-comp) be the class of polynomial (exponential) time computable distribution functions [2]. E denotes the class of sets computable in exponential time, i.e. in time 2^{cn}, for some constant $c > 0$.

A transducer approximates a distribution μ, iff $|T(x) - \mu(x)| \leq 2^{-|x|}$. In the following we require that there exists an enumeration $T_1, T_2, \ldots$ of transducers which approximate the distributions in P-comp and the k-th Transducer is computable in time n^k. For completeness we include a construction here.

Let $S_1, S_2, \ldots$ be an enumeration of all transducers such that S_k is n^{k-1}/c time bounded, for a constant c determined below. Then T_k is given by the procedure below. On input x, the procedure checks that S_k is monotone increasing on 1^i, $i \leq |x|$. Then it checks that within length $|x|$ S_k is, in a certain way, monotone increasing. This is done by dividing the remaining set (starting with $\Sigma^{|x|}$) in a left and a right half, and checking that the values of S_k on the boundaries of the parts are increasing (from lexicographically smaller strings to larger strings). This testing continues on the set containing the input string. If a test fails, the procedure outputs the value of S_k for the smaller string. Thus if at some point a test fails then $T'_k(x) = 0$.

input x, $|x| = n$
  for $i = 1$ to n **do**
    if $S_k(1^{i-1}, i) > S_k(1^i, i)$ **then output** $S_k(1^{i-1}, i)$
  if $S_k(1^{n-1}, n-1) > S_k(0^n, n)$ **then output** $S_k(1^{n-1}, n-1)$
  let $y = \lambda$
  for $i = 1$ to n **do**
    if $S_k(y00^{n-i}, n) \leq S_k(y10^{n-i}, n) \leq S_k(y01^{n-i}, n) \leq S_k(y11^{n-i}, n)$
      does not hold **then output** $S_k(y00^{n-i}, n)$
    if $y1$ is a prefix of x **then** $y = y1$ **else** $y = y0$
output $S_k(x, n)$

If S_k indeed computes a distribution then all consistency tests are valid and the pro-

cedure outputs $S_k(x,n)$. On the other hand the procedure ensures that the output is non decreasing and smaller than 1. The time needed to compute $T_k(x)$ is bounded by $d \cdot n \cdot n^{k-1}/c$, for some constant d, and hence less than n^k, for $c > d$. Again, for every transducer T in the enumeration, let T' denote the corresponding density function. That is for all x let $T'(x) = T(x) - T(x^-)$, where x^- is the predecessor of x.

In the context of average case complexity a problem is always a pair, a decision problem together with a distribution function. For example the class DistNP [7, 1] contains all pairs (D, μ), where $D \in$ NP and μ is a polynomial-time computable distribution function. Thus, another way to pose our motivating question is whether all or part of DistNP can be solved efficiently on the average.

A first intuitive definition of polynomial on the average could require that the expected value of a function is bounded by a polynomial, i.e.:

$$\exists k \forall n : \sum_{|x|=n} \mu'_n(x) \cdot f(x) \leq O(n^k). \tag{1}$$

This definition has the disadvantage that there exist functions f and distributions μ such that f is average polynomial with respect to μ but f^2 is not. One easily verified example is

$$f(x) = \begin{cases} 2^n, \text{ if } x = 0^n \\ |x|, \text{ otherwise} \end{cases} \quad \text{and } \mu'_n(x) = 2^{-n}.$$

This anomaly (and others) is solved by the following definition. (One other problem is, that even if a problem is solvable in polynomial-time on average with some oracle, which is itself in P, then a procedure combining both algorithms is not necessarily in average polynomial time). For a detailed discussion see e.g. [3].

Definition 2.1 *[7, 8] A function $f : \Sigma^* \to R^+$ is polynomial on average with respect to a distribution μ (polynomial on μ-average) if $\exists \delta > 0$ such that*

$$\sum_{x \in \Sigma^*} \mu'(x) \cdot \frac{f(x)^\delta}{|x|} < \infty.$$

Note that if a function satisfies equation (1) then it is also polynomial time on average. The class of problems solvable in average polynomial time is denoted by AP. In general, for a set of distribution functions $\mathcal{F}$ let AP($\mathcal{F}$) denote the set of decision problems (D, μ) such that $\mu \in \mathcal{F}$ and there exists a Turing machine M such that $L(M) = D$ and M is polynomial time on μ average.

3 Efficiently solvable on the average extends P

If a decision problem D is in P, then a problem (D, μ) is in $\mathrm{AP}(\mathcal{F})$ for every set of distribution functions $\mathcal{F}$ which contains μ. On the other hand, one might expect that for specific classes $\mathcal{F}$, if a problem (D, μ) is in $\mathrm{AP}(\mathcal{F})$ for all density functions $\mu \in \mathcal{F}$, then D is in P. To make the idea precise we use the following notion introduced in [11].

Definition 3.1 *Let $\mathcal{F}$ be a class of distribution functions, then $\mathrm{P}_{\mathcal{F}}$, is the class of languages D such that $(D, \mu) \in \mathrm{AP}(\mathcal{F})$ for every $\mu \in \mathcal{F}$.*

This definition allows to compare classes of problems, defined in terms of average-case complexity, directly with (well studied) worst-case complexity classes.

As was observed in [11] if $(A, \mu) \in \mathrm{AP}$ and μ is a standard distribution, then $A \in \mathrm{E}$, and hence, $\mathrm{P}_{\text{P-comp}}$ is contained in E. Therefore $\mathrm{P} \subseteq \mathrm{P}_{\text{P-comp}} \subseteq \mathrm{E}$. In [11] it was proved that $\mathrm{P}_{\mathrm{E-comp}} = \mathrm{P}$. This follows from the existence of polynomial-time complexity cores computable in E. But for the class $\mathrm{P}_{\text{P-comp}}$ we can prove the following strict inclusions, which show that $\mathrm{P}_{\text{P-comp}}$ is properly located between P and E. In some sense, one can conclude that $\mathrm{P}_{\text{P-comp}}$ enlarges the class of problems which can be solved efficiently. Note, that this is different to probabilistic classes like BPP, where it is still open whether BPP really extends P.

Theorem 3.2

$$\mathrm{P} \subsetneq \mathrm{P}_{\text{P-comp}} \subsetneq \mathrm{E}.$$

Proof. To prove that $\mathrm{E} \not\subseteq \mathrm{P}_{\text{P-comp}}$ let A be a Tally set in E which is not in P. Let μ be a distribution where $\mu(0^n) = c/n^2$ and 0 otherwise. Then μ is in P-comp. And for every Turing machine M, with $L(M) = A$, and for every constant $\delta > 0$ there exist infinitely many n such that $\mathrm{time}_M(0^n) > n^{3/\delta}$ and therefore:

$$\sum_x \mu'(x) \cdot \frac{(\mathrm{time}_M(x))^\delta}{|x|} \geq \sum_{i=1}^{\infty} \frac{c}{n_i^2} \cdot \frac{n_i^3}{n_i} = \infty.$$

This shows that the set A is not in $\mathrm{P}_{\text{P-comp}}$.

To see that there are sets A in $\mathrm{P}_{\text{P-comp}}$ which are not in P, note that for every distribution for every length n there are more than $2^n - 2^{n^\epsilon}$ strings with probability smaller than 2^{-n^ϵ}, for every $\epsilon < 1$. The idea (to define A) is to select strings which have low probability for every distribution in P-comp. These strings can then be used to diagonalize against P.

Let $M_1, M_2, \ldots$ be an enumeration of the polynomial-time bounded Turing machines such that the i-th machine runs in time n^i. Then P $= \{L(M_i) \mid i > 0\}$. Similarly let $T_1, T_2, \ldots$ be the above given enumeration of transducers which approximate the distributions in P-comp. That is, T_k is computable in time n^k, and for every $\mu \in$P-comp there is some T_k such that for all x: $|\mu'(x) - T'_k(x)| \leq 2^{-|x|}$. For every n and y, $|y| < n$ let $X = \{z \mid |z| = n \text{ and } y \text{ is a prefix of } z\}$. Then $|\mu'(X) - T'_k(X)| \leq 2 \cdot 2^{-n}$, since $T'_k(X) = T_k(y1^{n-|y|}) - T_k(y0^{n-|y|})$, and $T'_k(X)$ can be computed in time $O(n^k)$. Now consider the following procedure which defines a set A.

input x, $|x| = n$
  $y = \lambda$.
Phase I
  for $l = 1$ to $\log n$ **do**
    repeat $\log^2 n$ **times**
      let $L = \{z \mid |z| = n \text{ and } y0 \text{ is a prefix of } z\}$,
        $R = \{z \mid |z| = n \text{ and } y1 \text{ is a prefix of } z\}$,
(6) **if** $T'_l(L) < T'_l(R)$ **then** $y = y0$ **else** $y = y1$.
      if y is not a prefix of x **then reject**.
Phase II
  repeat $n - \log^3 n$ **times**
      let $L = \{z \mid |z| = n \text{ and } y0 \text{ is a prefix of } z\}$,
        $R = \{z \mid |z| = n \text{ and } y1 \text{ is a prefix of } z\}$,
    if $\max_{i \leq \log n}(T'_i(L)) < \max_{i \leq \log n}(T'_i(R))$ **then** $y = y0$ **else** $y = y1$.
    if y is not a prefix of x **then reject**.
Phase III
  simulate $M_{n^{\epsilon/2}}$ on input x and
  if $M_{n^{\epsilon/2}}$ accepts **then reject else accept**.

To see that A is in $\mathrm{P}_{\text{P-comp}}$ and not in P, we prove the following claims.

Claim 1 *For every n there exists one x of length n that is used to diagonalize against $M_{n^{\epsilon/2}}$, i.e. that passes Phase I and Phase II.*

Claim 2 *(A, μ) is computable in polynomial time on μ average, for every $\mu \in$*P-comp.

Proof of Claim 1: The loop in Phase I is started with $y = \lambda$, i.e. $L \cup R = \Sigma^n$. In each pass of the loop in line (6), the computation continues either on strings in L or in

R (which are of the same size) by selecting $y0$ or $y1$ resp. as prefix. The computation stops if x is not in the selected set L or R resp. Thus, after $\log^3 n$ iterations in Phase I, the computation continues on at least $2^{n-\log^3 n}$ strings.

Phase II is basically a prefix search, that, in $n - \log^3 n$ iterations, selects exactly one string which has the string y, $|y| = \log^3 n$, computed in Phase I as prefix. Note that for the string selected here it holds that $T_i'(x) \leq 2^{-n^\epsilon}$, for all $i \leq \log n$. To see that this is true observe that in every iteration (of Phase II) the computation continues on L, if for some T_l, $T_l'(R)$ is larger than every $T_i'(L)$, $i \leq \log n$. Otherwise the computation continues on R. Thus after $\log n$ iterations the maximal value of any T_i , $i \leq \log n$, on the remaining set is divided by two. Therefore after $n - \log^3 n$ iterations it holds that $T_i'(y) < 2^{-(n-\log^3 n)/\log n} < 2^{-n^\epsilon}$.

Proof of Claim 2: Let $X_0, X_1, \ldots, X_{\log n}$ be the subsets of Σ^n such that X_k is the set that remains after the kth transducer T_k is considered in Phase I. Then, for all $i > k$ it holds that $T_k'(X_i) \leq 2^{-\log^2 n} = n^{-\log n}$. To see this let $l = k$ in Phase I. Then T_k is considered, and in the repeat loop the computation continues either on the set L or on the set R, whichever has the smaller probability. In the beginning $T_k'(X_{k-1})$ is smaller than $T_k'(\Sigma^n) \leq 1$, and in each iteration the value is divided by two. This gives after $\log^2$ iterations the desired result.

Now let μ be any distribution in P-comp. Since $\mu \in$ P-comp, there exists a k such that T_k approximates μ, i.e. $|\mu'(x) - T_k'(x)| \leq 2^{-n}$. Then for every set X considered as L or R in the above procedure it holds that $|\mu'(X) - T_k'(X)| \leq 2 \cdot 2^{-n}$. (This error is small enough and will be ignored in the following estimation). Let $\text{time}_A^j(x)$ denote the time spend on input x to compute the for loop in Phase I the jth time. Then $\text{time}_A^j(x)$ is in $O(\log^2 n \cdot n^j)$ and hence, for all $n > 2^k$:

$$\forall j \leq k: \ \ \mu'(X_j) \leq \mu'(\Sigma^n) \quad \text{and} \quad \forall j > k: \ \ \mu'(X_j) \leq n^{-\log n}.$$

$$\sum_{j=1}^{k} \text{time}_A^j(x) \leq O(n^{1+k}) \quad \text{and} \quad \sum_{j=k}^{\log n} \text{time}_A^j(x) \leq O(n^{1+\log n}).$$

Phase II is computed only on strings in $X_{\log n}$. Since the loop is executed $n - \log^3 n$ times and each time for all T_i, $i \leq \log n$, it can be done in time $O(\log n \cdot n^{\log n}) < O(n^{2+\log n})$. The diagonalization in Phase III against a machine $M_{n^{\epsilon/2}}$ (which runs in time $n^{n^{\epsilon/2}} = 2^{n^{\epsilon/2} \log n}$) can be done in time $O(2^{n^\epsilon})$. Hence, we get the following estimation of the

average computation time f, choosing $\delta < 1/(2+k)$:

$$\sum_{x\in\Sigma^*} \mu'(x)\cdot\frac{f(x)^\delta}{|x|} \leq \sum_n \mu'(\Sigma^n)\cdot\frac{n^{\delta(1+k)}}{n} + \sum_n n^{-\log n}\cdot n^{\delta(1+\log n)} + \sum_n n^{-\log n}\cdot n^{\delta(2+\log n)} + 2^{-n^\epsilon}\cdot 2^{\delta n^\epsilon} \leq c.$$

□

Actually we proved that the set A, defined above, cannot be deceided by any $2^{n^{\epsilon/2}}$ time bounded machine, and that the only string of length n used for diagonalization can be generated in time $n^{O(\log n)}$.

4 Average polynomial time is different from NP

Let A be any language. Then $L(A) = \{0^n \mid \exists x, |x| = n \text{ and } x \in A\}$ is called the test language of A. A class $\mathcal{C}$ contains all its test languages if for every $A \in \mathcal{C}$ the test language $L(A)$ is also in $\mathcal{C}$. In this section it is shown that $\mathrm{P}_{\text{P-comp}}$ does not contain all its test languages and is therfore not equal to NP.

Lemma 4.1 *There exists a set A in $\mathrm{P}_{\text{P-comp}}$ such that $L(A)$ is not in $\mathrm{P}_{\text{P-comp}}$.*

Proof. Let A be a set in $\mathrm{P}_{\text{P-comp}} - \mathrm{P}$, as defined in the proof of Theorem 3.2. Then A has the following property: for each length n, A contains at most one string which can be generated in time $n^{O(\log n)}$. These strings are used to diagonalize against all sets computable in time $2^{n^{\epsilon/2}}$. Therefore A is not in DTIME(2^{n^δ}), $\delta < \epsilon/2$.

Assume that $\mathrm{P}_{\text{P-comp}}$ contains all its test languages. Then the test language $L(A)$ is also in $\mathrm{P}_{\text{P-comp}}$. Since $L(A)$ is tally, there exists a polynomial-time bounded machine M which computes $L(A)$. Then, A can be decided as follows:

On input x, verify if $0^{|x|} \in L(M)$ and if x is the string used for diagonalization in the construction of A. If both is true then x is accepted, otherwise x is rejected.

This implies that A is in DTIME($n^{O(\log n)}$), a contradiction to the choice of A. □

Since for all sets $A \in$ NP the test languages $L(A)$ are also in NP, we immediately get the following theorem:

Theorem 4.2 $\mathrm{P}_{\text{P-comp}} \neq \mathrm{NP}$.

Theorem 4.2 has an interesting consequence. If the set A constructed in Theorem 4.1 is in NP, then P $\subsetneq$ NP. However, if A is not in NP, then E $\subsetneq$ NP and hence NP $\subsetneq$ Exp.

Acknowledgments. The author would like to thank Osamu Watanabe for pointing out an error in the proof of Theorem 3.2 in an earlier version and for helpful comments.

References

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

[2] Y. Gurevich. Complete and incomplete randomized NP problems. *Proc. 28th IEEE Symposium on Foundations of Computer Science*, 111–117, 1987.

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

[4] D.S. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 5:284-299, 1984.

[5] E. Koutsoupias and C.H. Papadimitriou. On the greedy algorithm for satisfiability. *Information Processing Letters*, 43:53–55, 1992.

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

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

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

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

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

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