Towards a Theory of Average Case Complexity

Shai Ben-David, Benny Chor,
Oded Goldreich[1] and Michael Luby[2]

TR-89-004
February, 1989

This paper takes the next step in developing the theory of average case complexity, a study initiated by Levin. Previous works have focused on the existence of complete problems [Le, Gu, VL]. We widen the scope to other basic questions in computational complexity. For the first time in the context of average case complexity, we show the equivalence of search and decision problems, analyze the structure of NP under P reductions, and relate the NP versus average-P to non-deterministic versus deterministic (worst case) exponential time. We also present definitions and basic theorems regarding other complexity classes, such as average log-space.

[1]Ben-David, Chor and Goldreich are of the Department of Computer Science Technion, Haifa, Israel, supported by grant No. 86-003-1 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[2]International Computer Science Institute, Berkeley, California on leave of absence from the University of Toronto, research partially supported by a Natural Sciences and Engineering Research Council of Canada of Canada operating grant A8092 and by a University of Toronto Grant.

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 algorithms resarch – the probabilistic analysis of algorithms [J,K2]. However, this line of research has so far been applicable only to specific algiorithms and with respect to specific, typically uniform, probability distributions.

The general question of average case comlexity was addressed for the first time in [L84]. Levin's work can be viewed as the basis for a theory of average NP-completeness, much the same way as Cook [C] (and Levin [L73]) works are the basis for the theory of NP-completeness. Subsequently Gurevich [Gur] has presented several additional complete problems and gave a partial explenation of the difficulty in finding natural complete problems. Venkatesan and Levin [VL] showed the completeness of a graph coloring problem with respect to randomized reductions.

In this paper we widen the scope of investigation and consider basic computational questions in the context of average case complexity.

An average case complexity class consists of pairs, called distributional problems. Each such pair consists of a decision problem and a probability distribution on problem instances. Most of our work deals with the class (NP, P-computable), which is first defined by Levin [L84]. P-computable is the class of probability distributions μ such that there exists a polynomial time Turing machine which on input x computes $\mu(x)$, the total probability of all strings $y \leq x$. The easy distributional problems are those having an algorithm that solves the decision problem in average polynomial time, with respect to the probability distribution on problem instances. We denote this class by Average-P. 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-P, then so is Π_1.

A basic question regarding the theory of computational complexity is the relation between search and decision problems. Unfortunately, the standard Turing reduction of NP search problems to NP decision problems is not applicable in the distributional context. In Section 3, we present a randomized reduction of (NP, P-computable) search problems to (NP, P-computable) decision problems. Interestingly, this reduction can be carried out in RNC, yielding a reduction of NC search problems to NC decision problems that is different than the one in [KUW].

If (NP, P-computable) is not a subset of Average-P, then there are complete problems in (NP, P-computable) which are not in Average-P. A natural question is whether every (NP, P-computable) problem is either in Average-P or complete for (NP, P-computable). In Section 4, we resolve this question by showing that problems which are neither easy nor complete do exist. In fact, we show that the structural results of classical complexity theory [Lad] can be translated to the distributional context. Furthermore, we define a notion of one distribution being "harder" than another, and demonstrate a rich structure of distributions.

It is not clear whether (NP, P-computable) $\subseteq$ Average-P (even if P $\neq$ NP). In Section 5, we give strong indication that (NP, P-computable) is not a subset of Average-P by relating, for the first time, this question to a classical one in (worst case) complexity. Specifically, we prove that if (NP, P-computable) $\subseteq$ Average-P, then $\mathrm{NTime}(2^{O(n)}) = \mathrm{DTime}(2^{O(n)})$.

In Section 6, we present another natural family of distributions, P-samplable, for which random samples can be generated in time which is polynomial in the length of the sample generated. We define the class of distribution problems (NP, P-samplable) and present a complete problem for this class. We show that if one-way functions exist, then there are P-samplable distributions that are "very far" from any P-computable distribution.

In Section 7 we investigate notions of Average-logspace. We define average log-space reductions and the class (P, logspace-computable), and exhibit a complete distributional problem for this class. We further develop the theory and relate the question "is Dspace(n) = DTime($2^{O(n)}$)?" to the distributional question "is (P, logspace-computable) $\subseteq$ Average-logspace?". Structural properties of (P, logspace-computable) under logspace reductions, as well as the equivalence of search and decision, are also shown.

2 Definitions and Notations

In this section we give definitions of various concepts that are used throughout the paper. For sake of simplicity, we consider the lexicographic ordering of binary strings. Let $x-1$ denote the string preceding x in lexicographic order.

Definition: (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 convenience it is easy to normalize the distribution, so that it converges to one, by dividing by c.

Definition: (A Distributional Problem): A *distributional* decision *problem* (resp. *distributional* search *problem*) is a pair (D,μ) (resp. (R,μ)), where $D : \{0,1\}^* \to \{0,1\}$ is a Boolean predicate (resp. $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is a relation) and $\mu : \{0,1\}^* \to [0,1]$ is a distribution function.

In the sequel we mainly consider decision problems.

Definition: (The class 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 $|x|$ and in the number of bits of $\mu(x)$ produced).

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 [GMc, Gol]). In spite of this remark we usually present the density function and leave to the readr the verification that the corresponding distribution function is in P-computable.

We now present the class of distributional problems which coresponds to (the usual) NP. Most of the results in the paper apply to this class.

Definition: (The class (NP, P-computable) [L84]): A distributional problem (D, μ) belongs to the class (NP, P-computable) if D is an NP-predicate and μ is in P-computable.

The following definitions are due to Levin. As they may seem obscure at first glance, let us point out that naive formalizations of these definitions suffer from serious problems such as not being closed under functional composition of algorithms, being model dependent etc. More motivating arguments can be found in [J, GMc, Gol].

Definition: (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

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

Definition: (The class Average-P): A distributional problem (D, μ) is in the class Average-P if there exists an algorithm A solving D whose running time is polynomial on the average with respect to the distribution μ.

We view the classes Average-P and (NP, P-computable) as the average - case - complexity counterpart of P and NP (respectively). Another candidate, Average-NP – the class of distributional problems which can be solved by a non-deterministic machine running in average polynomial time with respect to P-computable, can be considered instead of (NP, P-computable). However, we feel that (NP, P-computable) is closer to the original motivation of investigating the average case complexity of NP. All results known (e.g. [L84, Gur, VL]) as well as the ones shown in this paper for the class (NP, P-computable) hold also for Average-NP.

2.1 Reducibility between Distributional Problems

We 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 problem does not necessarily yield such an algorithm for the first problem. For further discussions see [J, GMc, Gol].

Definition: (Deterministic Turing Reductions): We say that the deterministic oracle Turing machine M *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 with respect to μ_1. More precisely, let $t_M(x)$ be a bound on the running time of M on input x, then the function t_M is polynomial on the average with respect to μ_1.

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

$$M^{D_2}(x) = D_1(x)$$

where $M^{D_2}(x)$ denotes the output of the oracle machine M on input x and access to an 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 1 if machine M asks query y on input x and is 0 otherwise.

Definition: (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:* The same efficiency requirement as for Deterministic Turing Reductions.

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 that is the output of the oracle machine M on input x and access to oracle for D_2, with respect to the random choices of M.

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

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

where $Prob(Ask(M,x,y))$ is the probability that machine M asks query y on input x.

In the rest of the paper whenever we use the term *reduction* we mean a reduction of distributional problems, as defined above, and we use $\leq$ ($\leq_R$) to denote a deterministic (resp. randomized) reduction. It is easy to see that if $(D_1, \mu_1) \leq (D_2, \mu_2)$ 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) (see [GMc, Gol]). Similar comments hold with respect to $\leq_R$. Definitions of deterministic and randomized many-to-one reductions (i.e. Levin's original definition [L84] and the one used in [VL]) can be derived from these definitions as a special case.

3 Search versus Decision Problems

In this section we show that search and decision distributional problems are equivalent with respect to randomized reductions.

Before presenting our reduction of distributional search problems to distributional decision problems we remark that the standard reduction of a search problems to the corresponding decision problem does not have the domination property. On input x, the oracle

machine tries to find a "solution" y by asking an oracle queries of the form "is y' a prefix of a solution to x". Choosing the distribution of this decision problem to be uniform on these prefixes violates the domination condition, since when (for example) $|y'| = |y|/2$ the reduction maps an instance of the search problem to a much more rare instance of the decision problem (the ratio of probabilities of these instances is $> 2^{-|y|/2}$). It's not clear how to construct *polynomial time computable* distributions for the above decision problem so that the reduction will satisfy the domination condition.

We reduce every distributional search problem to an appropriate (distributional) decision problem in two steps. First, we randomly reduce the (distributional) search problem to a related (distributional) problem with a unique solution. This reduction follows the underlying principles of the proof of Valiant and Vazirani [VV]. Next, we reduce such a search problem to a related decision. The reader may easily verify that these reductions can be performed in fast parallel time (RNC).

3.1 Reducing a Search Problem to a Search of Unique Solution

Theorem 1: Let $\Pi_1 = (R_1, \mu_1) \in$ (NP, P-computable). Then there exists a unique solution search problem $\Pi_2 = (R_2, \mu_2) \in$ (NP, P-computable) such that Π_1 is randomly reducible to Π_2. (Queries to R_2 which do not have a unique solution are answered arbitrarily.)

Proof Sketch: For sake of simplicity, assume $(x, y) \in R_1$ implies $|x| = |y|$. Consider x such that $|x| = n$. Let $H_{n,k}$ be a set of universal hash functions (e.g. $n \times k$ Boolean matrices) [CW]. Define R_2 as follows: $(x', y) \in R_2$ if $x' = (x, k, h, \alpha)$, $(x, y) \in R_1$, $h \in H_{n,k}$ and $h(y) = \alpha$. The density function μ_2' assigns (x, k, h, α) probability $\mu_1'(x) \cdot |x|^{-1} \cdot |H_{n,k}|^{-1} 2^{-k}$ if $h \in H_{n,k}$ and $\alpha \in \{0,1\}^k$, and 0 otherwise. The reduction, effected by the probabilistic oracle M proceeds as follows. On input $x \in \{0,1\}^n$, machine M tries the following for every value of $k \in \{1, 2, \ldots, n\}$. Selects independently at random with uniform probability distribution an h in $H_{n,k}$ and $\alpha \in \{0,1\}^k$, makes the oracle query (x, k, h, α) and tests the reply. This is repeated polynomially many times. For some k (i.e. so that the number of y's satisfying $(x, y) \in R_1$ falls in the interval $[2^{k-1}, 2^k)$) at least one of the polynomially many queries has a unique solution with high probability and therefore the oracle answers with the unique solution. To conclude the proof note that the reduction satisfies the domination condition. □

Almost the same construction will reduce any (NP, P-computable) decision problem to an (NP, P-computable) decision problem with unique witnesses.

3.2 Reducing a Search of Unique Solution to a Decision problem

Theorem 2: Let $(R_1, \mu_1) \in$ (NP, P-computable) be a distributional unique-search problem. Then there exists a distributional decision problem $(D_2, \mu_2) \in$ (NP, P-computable) such that (R_1, μ_1) (deterministically) reducible to (D_2, μ_2).

Proof Sketch: For sake of simplicity, assume that $(x, y) \in R_1$ implies $|x| = |y|$. When discussing D_2, we write strings in the form (x, i), where $1 \leq i \leq |x|$. For all x such that there exists a unique y such that $(x, y) \in R_1$, $(x, i) \in D_2$ iff the $i^t h$ bit of y is equal to one. For all

other x, (x,i) is arbitrarily in D_2 or not. For each (x,i), we set $\mu_2'(x,i) = \mu_1'(x) \cdot frac1|x|$. This reduction has the domination property. □

4 On the Structure of (NP, P-computable)

Ladner has demonstrated the richness of the structure of NP under polynomial reductions [La]. The average case complexity counterpart, (NP, P-computable), is not less complex. There are several different ways to define a (average-polynomial) 'reducability order' on this class and they all enjoy structure theorems analogous to those of [Lad]. As the proofs, and even the precise statement of theorems, are quite lengthy, we present here only some characteristic examples.

The natural way to define a complexity ordering on the class (NP, P-computable) is through the reducibility of distributional problems. Another possibility is to fix a distribution $\mu \in$ P-computable and consider the relation $\leq_\mu$ defined on NP by $D_1 \leq_\mu D_2$ iff $(D_1,\mu) \leq (D_2,\mu)$. Theorem 5 implies that there is an infinite complexity hierarchy on NP with respect to $\leq_\mu$ for each $\mu \in$ P-computable such that there is a $D \in$ NP where (D,μ) is not in Average-P. It follows that a similar result applies to the 'natural' ordering of (NP, P-computable). It should be noted that in general the orderings $\leq_\mu$ and $\leq_P$ are different, i.e. $D_1, D_2 \in$ NP and $D_1 \leq_P D_2$ does not imply $D_1 \leq_\mu D_2$ for a specific or for any $\mu \in$ P-computable (P denotes the usual relation of many-one, (worst-case) polynomial-time reducibility).

There is yet another natural order which emerges in the context of distributional problems: an ordering of distributions. We define $\mu_1 \leq_{\text{DIST}} \mu_2$ if there exists a $D_2 \in$ NP such that for every $D_1 \in$ NP we have $(D_1,\mu_1) \leq (D_2,\mu_2)$. Note that the partial order induced on the equivalence classes derived from $\leq_{\text{DIST}}$ has a maximum element Max = $\{\mu : \exists D \in \text{NP}$ s.t. (D,μ) is complete for (NP, P-computable)$\}$. and a minimum element Min = $\{\mu : \forall D \in \text{NP}\ ((D,\mu) \in \text{Average-P}\ )\}$. Theorem 4 implies that there is an infinite complexity hierarchy on P-computable with respect to the ordering $\leq_{\text{DIST}}$ if (NP, P-computable) is not a subset of Average-P.

Theorem 3: For every distribution $\mu_1 >_{\text{DIST}}$ Min there exists a distribution μ_2 such that Min $<_{\text{DIST}} \mu_2 <_{\text{DIST}} \mu_1$.

Proof Sketch: For simplicity, we only construct here a distribution μ_2 such that the claim holds for a specific D'. Several other details are missing (e.g. we construct a μ_2 which does not converge to 1, etc.). Consider an enumeration of all candidate average polynomial time algorithms, $\{(A_i,\epsilon_i,c_i) : i \in \mathbf{N}\}$, where the A_i's range over all algorithms, ϵ_i's over the rationals, and c_i's over the integers. Let $\{(M_i,\epsilon_i,c_i) : i \in \mathbf{N}\}$, be a similar enumeration, where the M_i's are all oracle Turing machines.

In the sequel, we associate bit strings with their (integer) index in the standard lexicographic order. Define a function $\rho : \mathbf{N} \to \mathbf{N}$ and an algorithm for computing it and μ_2', recursively. On input x, compute the sequence $< \rho(1),\mu_2'(1),D(1),D'(1) >, < \rho(2),\mu_2'(2),D(2),D'(2) >, \ldots$ until reaching $n = |x|$ steps. Let r be the last integer for which $(\rho(r),\mu_2'(r),D(r),D'(r))$ is computed. If $r = 0$ set $\rho(x) = 1$, $\mu_2'(x) = \mu_1'(x)$ and stop.

If $\rho(r)$ is odd, let $i = (\rho(r)+1)/2$ and consider (A_i, ϵ_i, c_i). Use A_i, again with time bound n, to compute a sequence of pairs $< A_i(1), t_i(1) >, \ldots, < A_i(s), t_i(s) >$, where $t_i(j)$ is the running time of A_i on input j. Algorithm A_i is ruled out if it either runs too slow or errs. Namely, if either $\sum_{1 \leq y \leq s} \mu_2'(y) \cdot \frac{t_i(y)^{\epsilon_i}}{|y|} > c_i$ or $\exists y\ (\leq r, s)$ such that $D(y) \neq A_i(y)$ (and $\mu_2'(y) \neq 0$) then set $\rho(x) = \rho(r)+1$ else set $\rho(x) = \rho(r)$. In the first case set $\mu_2'(x) = 0$ while in the second case set $\mu_2'(x) = \mu_1'(x)$. Eventually, for some x the value of ρ will increase since no algorithm can solve D in time which is polynomial on the average with respect to a distribution which differs from μ_1 only on a finite portion.

If $\rho(r)$ is even, let $i = \rho(r)/2$ and consider (M_i, ϵ_i, c_i). Use M_i (and the above computed values of $D'(\cdot)$) to compute a sequence $t_i(1), \ldots, t_i(s)$, where $t_i(j)$ is the running time of M_i on input j, until n steps are reached or M_i asks a query q the answer to which has not been computed above (i.e. $q > r$). The oracle machine M_i is ruled out if it either runs too slow or asks queries violating the domination condition or errs. Namely, if either $\sum_{1 \leq y \leq s} \mu_2'(y) \cdot \frac{t_i(y)^{\epsilon_i}}{|y|} > c_i$ or $\exists y\ (\leq r, s)\ (\mu_1'(y) \neq 0)$ such that either all queries have even ρ value or $D(y) \neq M_i^{D'}(y)$ then set $\rho(x) = \rho(r)+1$ else set $\rho(x) = \rho(r)$. In the first case set $\mu_2'(x) = \mu_1'(x)$ while in the second case set $\mu_2'(x) = 0$. Eventually, for some x the value of ρ will increase since no machine having access to a finite oracle can solve D' in time which is polynomial on the average with respect to μ_2. It should be noticed that μ_2 is polynomial time computable. □

Corollary: If (NP, P-computable) is not a subset of Average-P then there exists a distributional problem in (NP, P-computable) that is not in Average-P and not complete for (NP, P-computable).

Theorem 4: For every problem (D_1, μ) in (NP, P-computable) but not in Average-P there exists a problem (D_2, μ) such that $(D_2, \mu) \in$ (NP, P-computable) but not in Average-P, (D_1, μ) is not (Turing) reducible to (D_2, μ), and (D_2, μ) is (many-to-one) reducible to (D_1, μ).

Proof Idea: Similar in its underlying principles to the proof of Theorem 4, except that we use the value of $\rho(x)$ to modify D instead of μ. □

5 Relations to Worst-Case Complexity

In this section we present theorems which relate questions about worst-case complexity classes to questions about average-case complexity classes. For lack of space, we omit all proofs. The first question we address is how likely is it that every NP problem has an easy on the average solution with respect to every P-computable distribution.

Theorem 5: If $\mathrm{NTime}(2^{O(n)}) \neq \mathrm{DTime}(2^{O(n)})$ then (NP, P-computable) is not a subset of Average-P.

We can get an 'if and only if' result by extending the class of distributions allowed from P-computable to exponential-time-computable.

Theorem 6: $\mathrm{NTime}(2^{O(n)}) \neq \mathrm{DTime}(2^{O(n)})$ if and only if there exists a unary language L such that there is no algorithm which decides membership in L and has running time polynomial on the average with respect to the distribution on unary strings $\mu'(1^n) = n^{-2}$.

The following theorem provides a connection between a tighter bound on the running time for NP problems and distributional complexity.

Definition: A probability distribution μ is *exponentially linear vanishing* if there exists constants $c, \delta > 0$ such that for every $x \in \{0,1\}^*$, $\mu'(x) \leq c \cdot 2^{-(1+\delta)|x|}$. (Recall that NP $\subseteq$ DTime($2^{n^{O(1)}}$)).

Theorem 7: NP $\subseteq$ DTime($2^{O(n)}$) if and only if every problem in (NP, P-computable) with an exponentially linear vanishing distribution is in Average-P.

Corollary: If NP is not a subset of DTime($2^{O(n)}$) then there are problems in (NP, P-computable) with exponentially linear vanishing distributions that are neither in Average-P nor complete for (NP, P-computable) with respect to deterministic reductions.

6 A Wider Class of Distributions

In this section we consider a natural extension of the class of P-computable disributions – the class of P-samplable distributions. Distributions in this class arise in polynomial time probabilistic computations. We show that under "modest" cryptographic assumptions, there are P-samplable distributions that are "very far" from any P-computable distribution. We proceed to present a complete distributional problem in (NP, P-samplable). The proof of completness here is very different then for the (NP, P-computable) case.

Definition: (The class P-samplable): A distribution μ is in the class P-samplable if

1. The weight of Σ^n, $\mu_n \stackrel{\text{def}}{=} \mu(1^n) - \mu(1^{n-1})$ is polynomial time (in n) computable.
2. There is an efficient sampling algorithm for Σ^n under μ_n. That is, a probabilistic polynomial time Turing machine M that on input n (in unary) outputs a string $x \in \Sigma^n$ with probability $\mu'(x)/\mu_n$.

Theorem 8:

1. Every P-computable distribution is also in P-samplable.
2. If one-way permutation exists, then there is a P-samplable distribution μ such that for any distribution η, if η' is polynomial time computable, then η does not dominate μ.

Proof Sketch: 1. Given n, M picks at random with uniform distribution a truncated rational ρ in $[\mu(1^{n-1}), \mu(1^n))$ (the length of expansion depends on μ). It then finds, via binary search and queries to μ, the unique string $x \in \Sigma^n$ satisfying $\mu(x-1) < \rho \leq \mu(x)$.

2. Let f be a one-way permutation of $\{0,1\}^n$. Define the distribution μ on $\{0,1,2\}^n$ by

$$\mu'(y,z) = \begin{cases} \frac{1}{n^3 \cdot 2^n} & \text{if } |y| = |z| = n,\ \exists x\ y = f(x),\ z = s2\ldots2,\ s \text{ prefix of } x \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that μ is P-samplable. Suppose η dominates μ. For most y's of length n, there could be only polynomially many w's such that $\eta'(y,w) \geq \frac{1}{n^c \cdot 2^n}$ (otherwise η would sum up to more than 1). We say that such w is heavy for y, and we call y with only polynomially many heavy w's "common". Given a common y, an oracle for η' can be used to find $f^{-1}(y)$ in polynomial time: The search focuses on heavy nodes, of which there are only polynomially many, and ignores extensions in "light" directions. Because of the domination condition, all nodes on the branch with prefixes of $f^{-1}(y)$ are heavy. □

The above Theorem leads to the definition of the following class of distribution problems.

Definition: (The class (NP, P-samplable)): A problem (D,μ) belongs to the class (NP, P-samplable) if D is an NP-predicate and μ is in P-samplable.

Intuitively, the question of whether there exists a problem in (NP, P-samplable) which is not in Average-P can be interpreted as asking whether one party can efficiently find instances of an NP problem which will be hard to solve for another party. This should be contrasted with the question of whether there exists a problem in (NP, P-computable) which is not in Average-P, which can be interpreted as asking whether one can efficiently find instances of an NP problem which will be hard to solve, even for the person who has constructed them! For further discussion see [Gol].

Theorem 9: There exist problems which are complete in (NP, P-samplable).

Proof Sketch: It is possible to effectively enumerate all polynomial time sampling Turing machines, and hence all P-samplable distributions $\mu_1, \mu_2, \ldots$. (Such effective enumeration is apparently not possible for P-computable distributions.) Define a *universal distribution*

$$\mu'(x) \stackrel{\text{def}}{=} \sum_{i=1}^{\infty} \frac{\mu_i'(x)}{i^2} .$$

Then μ is in P-samplable, and for any NP complete problem L, (L,μ) is complete in P-samplable. (A similar idea is used in [L86].)

We can modify the construction so that, if one-way functions exist, then the resulting problem is complete in (NP, P-samplable) but is not in (NP, P-computable).

7 Average logspace

The natural adaptation of the definition of Average-P fails for Average-logspace. We present an alternative definition of Average-logspace, which satisfies some desired properties, such as Average-logspace $\subseteq$ Average-P. We define the class (P, logspace-computable), and give an appropriate version of the bounded halting problem, together with a distribution in logspace-computable, which are shown to be complete in (P, logspace-computable) with respect to logspace reductions.

The first attempt at the definition is the following: An algorithm A is logspace on the

average with respect to distribution μ if

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

where $s_A(x)$ denotes the work space used by algorithm A on input x. Unfortunately, this definition has some serious handicaps, the most upsetting of which is that for every $0<\alpha<1$, algorithms that use work space n^α on every input of length n, will be in average logspace with respect to the uniform distribution. (As a consequence, average logspace will not necessarily be contained in average-P.) Instead, we propose the following definitions.

Definition: (Logarithmic on the Average): A function $f:\{0,1\}^*\to\mathbf{N}$ is *logarithmic on the average* with respect to a distribution μ if there exists a constant $\epsilon>0$ such that

$$\sum_{x\in\{0,1\}^*} \mu'(x)\cdot\frac{(2^{f(x)})^\epsilon}{|x|}<\infty$$

Definition: (The class Average-logspace): A distributional problem (D,μ) is in the class Average-logspace if there exists an algorithm A solving D using work space s_A, which is logarithmic on the average with respect to the distribution μ. In other words, the exponent of the work space, $2^{s_A(x)}$, is polynomial on the average.

This revised definition overcomes the above mentioned difficulties. In addition, the notion of domination of probability distributions will still be applicable, and Average-logspace is closed under average logspace (many-to-one) reductions.

This approach can be generalized to the definition of the class Average-Uniform-NC. To do this, we use the characterization of Uniform-NC by alternating logspace and poly-log time Turing machines [R]. We will now require that both the exponent of the work space (i.e. $2^{\epsilon\cdot s_A(x)}$) and the exponent of the time to some power $\delta>0$ (i.e. $2^{\epsilon\cdot t_A(x)^\delta}$) be polynomial on the average.

The class (P, logspace-computable) is defined analogously to the definition of (NP, P-computable). Namely, (P, logspace-computable) consists of distributional problems, (D,μ), with $D\in$ P and the distribution function μ is logspace computable. It should be noticed that many natural distributions, including the uniform distribution, are logspace computable.

Deterministic Bounded Halting (*DBH*) is defined over triples $(M,x,1^k)$, where M is a deterministic machine, x is a binary string and k is an integer (given in unary). The problem is to determine whether M accepts x within k steps. Clearly, $DBH\in$ P, and it is not hard to see that it is P-complete with respect to logSpace reductions. The distribution μ_0 is defined in terms of its density function

$$\mu_0'(M,x,1^k)=\left(\frac{1}{|M|^2}\cdot 2^{-|M|}\right)\cdot\frac{1}{k^2}\cdot\left(\frac{1}{|x|^2}\cdot 2^{-|x|}\right)$$

Theorem 10: The distributional problem (DBH, μ_0) is complete in (P, logspace-computable).

Proof Sketch The proof uses ideas of the (NP, P-computable)-completeness proof of Distributional Bounded Halting, as presented in [Gur,GMc,Gol]. If μ is logspace computable,

then so is the coding function C_μ which used in the reduction. The decoding algorithm C_μ^{-1} is not logspace computable. However, decoding can can be done in deterministic polynomial time, which is sufficient for our purpose. □

We remark that it is possible to define a (deterministic) version of the tiling problem, so that with the distribution μ_0, it is complete in (NP, P-computable).

The following theorems are analogues of these appearing in Section 3. Again, for lack of space, we omit their proofs.

Theorem 11: If DTime($2^{O(n)}$) $\neq$ DSpace(n) then there exists a problem in (P, logspace-computable) which is not in Average-logspace.

Theorem 12: DTime($2^{O(n)}$) $\neq$ DSpace(n) if and only if there exists a unary language L such that deciding membership in L is not logspace on the average with respect to the uniform distribution on unary strings.

Theorem 13: P $\subseteq$ DSpace(n) if and only if every problem in (P, logspace-computable) having an exponentially-vanishing distribution is in Average-logspace.

All the theorems in Section 5, dealing with the structure of (NP, P-computable), can be modified to the context of (P, logspace-computable). In particular, we have the following.

Theorem 14: If (P, logspace-computable) $\neq$ Average-logspace then there exists a distributional problem in (P, logspace-computable) - Average-logspace which is not (P, logspace-computable)-complete.

As the reduction presented in Section 5 can be carried out in RNC, it follows that (NP, P-computable) search problems are RNC-reducible to (NP, P-computable) decision problems.

Acknowledgements

We would like to thank Shimon Even, Mauricio Karchmer, Hugo Krawczyk, Ronnie Roth, and Avi Wigderson for helpful discussions. We are grateful to Leonid Levin for very interesting discussions.

References

[CW] Carter, J., and M. Wegman, "Universal Classes of Hash Functions", *JCSS*, 1979, Vol. 18, pp. 143–154.

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

[GJ] 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.

[Gol] Goldreich, O., "Towards a Theory of Average Case Complexity (a survey)", TR-507, Computer Science Department, Technion, Haifa, Israel, March 1988.

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

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

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

[K] 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.

[K2] Karp, R.M., "Probabilistic Analysis of Algorithms", manuscript.

[KUW] Karp, R.M., E. Upfal, and A. Wigderson, "Are Search and Decision Problems Computationally Equivalent?", *Proc. 17th ACM Symp. on Theory of Computing*, 1985, pp. 464–475.

[Lad] Ladner, R.E., "On the Structure of Polynomial Time Reducibility", *Jour. of the ACM*, 22, 1975, pp. 155–171.

[L73] Levin, L.A., "Universal Search Problems", *Problemy Peredaci Informacii 9*, pp. 115–116, 1973. Translated in *problems of Information Transmission 9*, pp. 265–266.

[L84] 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.

[L85] Levin, L.A., "One-Way Function and Pseudorandom Generators", *Proc. 17th ACM Symp. on Theory of Computing*, 1985, pp. 363–365.

[R] Ruzzo, W.L., "On Uniform Circuit Complexity", *JCSS*, 22, 1981, pp. 365–385.

[Sip] Sipser, M., "A Complexity Theoretic Approach to Randomness", *Proc. 15th ACM Symp. on Theory of Computing*, 1983, pp. 330–335.

[Sto] Stockmeyer, L.J., "The Complexity of Approximate Counting", *Proc. 15th ACM Symp. on Theory of Computing*, 1983, pp. 118–126.

[VV] Valiant, L.G., and V.V. Vazirani, "NP is as Easy as Detecting Unique Solutions", *Proc. 17th ACM Symp. on Theory of Computing*, 1985, pp. 458–463.

[VL] Venkatesan, R., and L.A. Levin, "Random Instances of a Graph Coloring Problem are Hard", to appear in *Proc. 20th ACM Symp. on Theory of Computing*, 1988.