Average Case Complexity under the Universal Distribution Equals Worst Case Complexity

Ming Li

University of Waterloo, Department of Computer Science
Waterloo, Ontario N2L 3G1, Canada

Paul M.B. Vitányi

Centrum voor Wiskunde en Informatica, Kruislaan 413
1098 SJ Amsterdam, The Netherlands
and
Universiteit van Amsterdam, Faculteit Wiskunde en Informatica

Abstract

The average complexity of any algorithm whatsoever (provided it always terminates) under the universal distribution is of the same order of magnitude as the worst-case complexity. This holds both for time complexity and for space complexity. To focus our discussion, we use as illustrations the particular case of sorting algorithms, and the general case of the average case complexity of NP-complete problems.

1 Introduction

[1] For many algorithms the average case running time under some distributions on the inputs is less than the worst-case running time. For instance, using Quicksort on a list of n items to be sorted gives under the Uniform Distribution on the inputs an average running time of $O(n \log n)$ while the worst-case running time is $\Omega(n^2)$. The worst-case running time of Quicksort is typically reached if the list is already sorted or almost sorted, that is, exactly in cases where we actually should not have to do much work at all. Since in practice the lists to be sorted occurring in computer computations are very likely to be sorted or almost sorted, programmers implementing systems involving sorting algorithms tend to resort

[1]The work of the first author was supported in part by NSERC Operating Grant OGP0036747. Part of the work was performed while he was with the Department of Computer Science, York University, North York, Ontario, Canada. The work of the second author was supported in part by NSERC International Scientific Exchange Award ISE0046203. A preliminary version of this work was presented at the *30th Annual IEEE Symposium on Foundations of Computer Science*, 1989, pp. 34-39.

to fast sorting algorithms of which the provable average run-time is of equal order of magnitude as the worst-case run-time, even though this average running time can only be proved to be $O(n \log^2 n)$ under the Uniform Distribution as in the case of Shellsort, or to some randomized version of Quicksort.

In the case of NP-complete problems the question arises whether there are algorithms that solve them in polynomial time "on the average". Whether this phenomenon occurs must depend on the combination of the particular NP-complete problem to be solved and the distribution of the instances. Obviously, some combinations are easy on the average, and some combinations are hard on the average, by tailoring the distribution to the ease or hardness of the individual instances of the problem. This raises the question of a meaningful definition of a "hard on the average" problem.

L.A. Levin [Le] has shown that for the Tiling problem with uniform distribution of instances there is no polynomial on the average algorithm, unless there exists such an algorithm for each combination of an NP-complete problem and polynomial time computable probability distribution.

Here it is shown that under the Universal Distribution *all* NP-complete problems are hard to compute on the average unless P = NP.

2 The Universal Distribution

Let N, Q, and R denote the set of nonnegative integers, nonnegative rational numbers, and nonnegative real numbers, respectively. A superscript '+' excludes zero. We consider countably infinite sample spaces, say $S = N \bigcup \{u\}$, where u is an 'undefined' element not in N. A function P from S into R, such that $\sum_{x \in S} P(x) \leq 1$ defines a *probability distribution* on S. (This allows us to consider defective probability distributions on the natural numbers, which sum to less than one, by concentrating the surplus probability on u.) A probability distribution P is called *enumerable*, if the set of points

$$\{(x, y) : x \in N,\ y \in Q,\ y \leq P(x)\}$$

is recursively enumerable. That is, $P(x)$ can be approximated from below by a Turing machine, for all $x \in N$. ($P(u)$ can be approximated from above. A probability distribution P is recursive if $P(x)$ can be approximated both from below and above by a Turing machine, for all x.)

Levin has shown that we can effectively enumerate all enumerable probability distributions, $P_1, P_2, \ldots$. In particular, there exists a *universal enumerable probability distribution*, denoted by, say, $\mathbf{m}$, such that

$$\forall_{k \in N} \exists_{c \in Q} \forall_{x \in N} [c\mathbf{m}(x) \geq P_k(x)]. \tag{1}$$

That is, $\mathbf{m}$ dominates each P_k multiplicatively. It is convenient to define

$$\mathbf{m}(x) = 2^{-K(x)}, \tag{2}$$

where $K(x)$ is the prefix variant of Kolmogorov complexity [G1]. In equation (1), the constant c can be set to

$$c = 2^{K(P_k)+O(1)} = 2^{K(k)+O(1)} = O(k \log^2 k). \tag{3}$$

This means that we can take c to be exponential in the length of the shortest self-delimiting binary program to compute P_k.

The universal distribution (rather, its continuous version) was originally discovered by R.J. Solomonoff in 1964, with the aim of predicting continuations of finite prefixes of infinite binary sequences. We can view the discrete probability density $\mathbf{m}$ as the *a priori* probability [2] of finite objects in absence of any knowledge about them [So]. Levin has shown that Solomonoff's definition, and the two definitions (1) and (2) given above, are equivalent up to a multiplicative constant. Thus, three very different formalizations turn out to define the same notion of universal probability. Such a circumstance is often taken as evidence that we are dealing with a fundamental concept. See [ZL] for the analogous notions in continuous sample spaces, [G2], and [LV1] or [LV2] for elaboration of the cited facts and proofs.

This universal distribution has many important properties. Under $\mathbf{m}$, easily describable objects have high probability, and complex or random objects have low probability. Other things being equal, it embodies Occam's Razor, which says we should prefer simple explanations over complicated ones. To give an example, with $x = 2^n$ we have $K(x) \leq \log n + 2 \log\log n + O(1)$ and $\mathbf{m}(x) = \Omega(1/n \log^2 n)$. If we generate the binary representation of y by n tosses of a fair coin, apart from the leading '1', then for the overwhelming majority of outcomes we shall have $K(y) \geq n$ and $\mathbf{m}(y) = O(2^{-n})$.

By Markov's inequality, for any two probability distributions P and Q, for all k, we have $Q(x) \geq P(x)/k$ with P-probability at least $1 - 1/k$. By equations (1) and (3) therefore, for each enumerable probability distribution $P(x)$ we have

$$\sum\{P(x) : 2^{K(P)+O(1)}\mathbf{m}(x) \geq P(x) \geq \mathbf{m}(x)/k\} \geq 1 - 1/k, \tag{4}$$

for all $k > 0$. In this sense, with high P-probability, $P(x)$ is close to $\mathbf{m}(x)$, for each enumerable P. The distribution $\mathbf{m}$ is the only enumerable one which has that property. If the problem instances are generated algorithmically, then the distribution is enumerable. In absence of any a priori knowledge of the actual distribution therefore, apart from that it is enumerable, studying the average behavior under $\mathbf{m}$ is considerably more meaningful than studying the average behavior under any other particular enumerable distribution.

3 Average Case Complexity

Let $x \in N$. Let $l(x)$ denote the *length* of the binary representation of x. Let $t(x)$ be the running time of algorithm A on problem instance x. Define the *worst-case time complexity* of A as $T(n) = \max\{t(x) : l(x) = n\}$. Define the *average time complexity* of A with respect to a a probability distribution P on the sample space S by

$$T^P_{average}(n) = \frac{\sum_{l(x)=n} P(x)\, t(x)}{\sum_{l(x)=n} P(x)}.$$

Example (Quicksort). Let us compare the average time complexity for Quicksort under the Uniform Distribution $L(x)$ and the one under the Universal distribution $\mathbf{m}(x)$. Define $L(x) = 2^{-2l(x)}$, such that the conditional probability $L(x|\, l(x) = n) = 2^{-n}$. We encode the list of elements to be sorted as nonnegative integers in some standard way.

[2]Consider an enumeration $T_1, T_2, \ldots$ of Turing machines with a separate binary one-way input tape. Let T be such a machine. If T halts with output x, then T has scanned a finite initial segment of the input, say p, and we define $T(p) = x$. The set of such p for which T halts is a prefix code: no such input is a proper prefix of another one. Assume the input is provided by tosses of a fair coin. The probability that T halts with output x is $P_T(x) = \sum_{T(p)=x} 2^{-l(p)}$, where $l(p)$ denotes the length of p. Then $\sum_{x \in N} P_T(x) \leq 1$, the deficit from one being the probability that T doesn't halt. Concentrate this surplus probability on $P_T(u)$, such that $\sum_{x \in S} P_T(x) = 1$. It can be shown that P is an enumerable probability distribution iff $P = \Theta(P_T)$ for some T. In particular, $P_U(x) = \Theta(\mathbf{m}(x))$ for a universal machine U. From this, properties (1), (2), and (3) can be derived.

For Quicksort, $T^L_{average}(n) = \Theta(n \log n)$. We may expect that $T^m_{average}(n) = \Omega(n \log n)$. But the Theorem will tell us much more, namely, $T^m_{average}(n) = \Omega(n^2)$! Let us give some intuition why this is the case. With the low average time-complexity under the Uniform Distribution, there can only be $o((\log n)2^n/n)$ strings x of length n with $t(x) = \Omega(n^2)$. Therefore, given n, each such string can be described by its sequence number in this small set, and hence for each such x we find $K(x|\,n) \leq n - \log n + 3 \log\log n$. (Since n is known, we can find each $n - k$ by coding k self-delimiting in 2 log k bits. The inequality follows by setting $k = \log n - \log\log n$.) Therefore, no really random x's, with $K(x|n) \geq n$, can achieve the worst-case run time $\Omega(n^2)$. Only strings x which are non-random, with $K(x|n)\ n$, among which are the sorted or almost sorted lists, and lists exhibiting other regularities, can have $\Omega(n^2)$ running time. Such lists x have relatively low Kolmogorov complexity $K(x)$ since they are regular (can be shortly described), and therefore $\mathbf{m}(x) = 2^{-K(x)}$ is very high. Therefore, the contribution of these strings to the average running time is weighted very heavily. This intuition can be made precise in a much more general form. We assume that all inputs to an algorithm are coded as integers according to some standard encoding.

Theorem. *Let A be any algorithm, provided it terminates for all inputs in N. Let the inputs to A be distributed according to* $\mathbf{m}$. *Then the average case time complexity is of the same order of magnitude as the corresponding worst-case time complexity.*

Proof. We define a probability distribution $P(x)$ on the inputs that assigns high probability to the inputs for which the worst-case complexity is reached, and zero probability for other cases.

Let A be the algorithm involved. Let $T(n)$ be the worst-case time complexity of A. Clearly, $T(n)$ is recursive (for instance by running A on all x's of length n). Define the probability distribution $P(x)$ by:

1 For each $n = 1, 2, \ldots$, define $a_n := \sum_{l(x)=n} \mathbf{m}(x)$;

2 if $l(x) = n$ and x is lexicographically least with $t(x) = T(n)$, then $P(x) := a_n$, else $P(x) := 0$.

It is easy to see that a_n is enumerable since $\mathbf{m}(x)$ is enumerable. Therefore, $P(x)$ is enumerable. Setting $P(u) = \mathbf{m}(u)$, we have defined $P(x)$ such that $\sum_{x \in S} P(x) = \sum_{x \in S} \mathbf{m}(x)$, and $P(x)$ is an enumerable probability distribution. The average case time complexity $T^m_{average}(n)$ with respect to the $\mathbf{m}(x)$ distribution on the inputs, using $c_P \mathbf{m}(x) \geq P(x)$ by (1), is obtained by:

$$
\begin{aligned}
T^m_{average}(n) &= \frac{\sum_{l(x)=n} \mathbf{m}(x)\, t(x)}{\sum_{l(x)=n} \mathbf{m}(x)} \\
&\geq \frac{1}{c_P} \sum_{l(x)=n} \frac{P(x)}{\sum_{l(x)=n} \mathbf{m}(x)} T(n) \\
&\geq \frac{1}{c_P} \sum_{l(x)=n} \alpha \frac{P(x)}{\sum_{l(x)=n} P(x)} T(n) \\
&\geq \frac{\alpha}{c_P} T(n),
\end{aligned}
$$

where

$$
\alpha = \frac{\sum_{l(x)=n} P(x)}{\sum_{l(x)=n} \mathbf{m}(x)} = 1.
$$

The proof of the theorem is finished by the observation that

$$
T(n) \geq T^m_{average}(n)
$$

holds vacuously.

If P in the proof is P_k in the standard effective enumeration $P_1, P_2, \ldots$ of enumerable semimeasures, then we can set $c_P \leq k \log^2 k$ by equation (3). Namely, considering the binary representations of positive integers, $\bar{c}k$ is a prefix code with $l(c(k)) = \log k + 2 \log\log k$. Since there is a Turing machine halting with output k iff the input is $c(k)$, the length $K(k)$ of the shortest prefix free program for k does not exceed $l(c(k))$. This gives an interpretation to the constant of proportionality between the **m**-average complexity and the worst-case complexity: if the algorithm to approximate $P(x)$ from below is the kth algorithm in the standard effective enumeration of all algorithms, then:

$$T^{m}_{average}(n) \ \geq \ \frac{T(n)}{k \log^2 k}.$$

Hence we must code the algorithm to compute P as compact as possible to get the most significant lower bound. That is, the ease with which we can describe (algorithmically) the strings which produce a worst case running time determines the closeness of the average time complexity to the worst-case time complexity.

It would seem that the result has implications for algorithm design. For large n, average case analysis is misleading because real inputs tend to be distributed according to the universal distribution, not according to the uniform distribution. But the constant of proportionality in the high order term is something like $2^{-K(P)}$. Consider Quicksort again. It runs in $n \log n$ time under the uniform distribution but n^2 time worst case. So its real average time complexity might be something like $n \log n + n^{2 - K(P)}$. As long as the input size n satisfies $n \log n \geq n^{2 - K(P)}$, like when $K(P) \geq \log n$, experimental testing of the average running time of Quicksort must show a considerably improvement over the n^2 worst case behavior, corresponding to the analysis for the uniform distribution. Here $K(P)$ is the size of the shortest program to generate the pseudo uniform distribution over the sample. Frequently people use pseudo random permutations in order to kill off the worst case behavior, or to choose the 'pivot' in the algorithm randomly. This results in randomized Quicksort. Again, the Kolmogorov complexity of the random number generator must be at least $\log n$ in order to drive the high order term down to n. Thus, random number generators should be selected with the input size to the final algorithm in mind. An interesting question is whether any random number generator of Kolmogorov complexity $\log n$ is sufficient – or are they all sufficient?

We finish with some immediate corollaries.

Corollary. The analogue of the Theorem holds for other complexity measures (like *space* complexity), by about the same proof.

Corollary. The **m**-average time complexity of Quicksort is $\Omega(n^2)$.

Corollary. For each NP-complete problem, if the problem instances are distributed according to **m**, then the average running time of any algorithm that solves it is superpolynomial unless P = NP. (A result related to this corollary is suggested in [BCGL], apparently using different arguments.)

Following the work reported here, related questions with respect to more feasible classes of probability distributions (like polynomial time computable ones) have been studied in [Mi].

Acknowledgements.

Richard Beigel, Benny Chor, Gloria Kissin, John Tromp, and Vladimir Uspenskii commented on the manuscript. Mike O'Donnell raised the question we address here during a lecture given by the second author.

References

{BCGL] S. Ben-David, B. Chor, O. Goldreich, M. Luby, On the theory of average case complexity, Proc. 21th STOC, 1989, pp. 204-216.

[G1] P. Gács, On the symmetry of algorithmic information, *Soviet Math. Dokl.*, **15**(1974), pp. 1477-1481, (Correction, *Ibid.*, 15(1974), p. 1481)

[G2] P. Gács, Lecture notes on descriptional complexity and randomness, Manuscript, Boston University, Boston, Mass., October 1987 (Unpublished).

[LV1] M. Li and P. Vitanyi, Kolmogorov complexity and its applications, in *Handbook for Theoretical Computer Science, Vol. 1*, Jan van Leeuwen, Managing Editor, North-Holland, 1990.

More up-to-date:

M. Li and P. Vitanyi, *An Introduction to Kolmogorov complexity and Its Applications*, Springer-Verlag, New York, 1997 (2nd Edition).

[LV2] M. Li and P. Vitanyi, Inductive reasoning and Kolmogorov complexity, 4th IEEE Structure in Complexity Theory conference, 1989, pp. 165-185. Final version: *J. Comp. System Sciences*, 44:2(1992), 343-384.

See also: P.M.B. Vitanyi and M. Li, Minimum Description Length Induction, Bayesianism, and Kolmogorov Complexity, *IEEE Trans. Inform. Theory*, IT-46:2(2000), 446–464.

[Le] L.A. Levin, Average case complete problems, *SIAM J. Comp.*, 15(1986), pp. 285,286.

[Mi] P.B. Milterson, The complexity of malign ensembles, Tech. Rept. PB-335, DAIMI, Aarhus University, September 1990.

[ZV] A.K. Zvonkin and L.A. Levin, The complexity of finite objects and development of the concepts of information and randomness by means of the theory of algorithms, *Russian Math. Surveys*, 25:6(1970), pp. 83-124.