Average Case Complete Problems[*]

Leonid A. Levin[†]
Boston University[‡]

Abstract

Many interesting combinatorial problems were found to be NP-complete. Since there is little hope to solve them fast in the worst case, researchers look for algorithms which are fast just "on average". This matter is sensitive to the choice of a particular NP-complete problem and a probability distribution of its instances. Some of these tasks were easy and some not. But one needs a way to distinguish the "difficult on average" problems. Such negative results could not only save "positive" efforts but may also be used in areas (like cryptography) where hardness of some problems is a frequent assumption. It is shown below that the Tiling problem with uniform distribution of instances has no polynomial "on average" algorithm, unless every NP-problem with every simple probability distribution has it. It is interesting to try to prove similar statements for other NP-problems which resisted so far "average case" attacks.

Conventions: A *random problem* is a pair (μ, R), where $R \subset \{1, 2, ...\}^2$ is an "instance - witness" (or input-output) relation, and $\mu: \{1, 2, ...\} \to [0, 1]$ is a probability *distribution function* on inputs (i.e. $\mu(x)$ is the probability of all instances not exceeding x). Its *density* $\mu'(x) = \mu(x) - \mu(x - 1)$ is the probability of a particular input. A problem is in NP, if both R and μ are computable in time polynomial in length $|x| = \lceil \log x \rceil$ of input. A machine independent notion of a *polynomial on average* problem (μ, R) assumes $\overline{R}(x) \iff \exists y R(x, y)$ to be computable in time polynomial in t where $t(x)/|x|$ is bounded by a constant on average i.e. $\sum \mu'(x) t(x)/|x| < \infty$. *Domination* $\mu \preceq \mu_1$ means $\exists k \forall x \;\; \mu'(x)/\mu_1'(x) < |x|^k$. A polynomial time algorithm f *reduces* a problem (μ_1, R_1) to $(f(\mu_2), R_2)$, if $\mu_1 \preceq \mu_2$ (thus, likely inputs of one problem are mapped into likely inputs of the other) and $\overline{R}_1(x) \iff \overline{R}_2(f(x))$. Here $f(\mu)$ is the distribution of outputs of f and maps x to $\sum\limits_{f(y) \leq x} \mu'(y)$.

Reductions are closed under composition, and if $A(x)$ is a fast on average algorithm for $(f(\mu_2), R_2)$ then $A(f(x))$ works at most polynomially slower for (μ_1, R_1). The polynomials $|x|^k$ in domination and in reduction time may be replaced by a polynomial on average $t^k(x)$ to get *weak* reducibility. The definitions can also be modified for a more elegant "inverting" formulation of NP problems: to actually find y for which $x = r(y)$.

Definition:
A random NP problem is complete, *if every random NP problem is reducible to it.*

Example: Tiling. A tile is a square with a latin letter in every node. Tiles with matching letters can be joined. An instance (u, v, s) of the Tiling Problem, has a subset u of tile types considered "legal", a string $v = 0^n$ of 0's, and a string s of matching legal tiles. The problem is to extend s to a square of n^2 matching legal tiles. The joint probability, of u, n and $k = |s| < n$ is, say, $O(n^{-3})$ and every tile in s is chosen sequentially with equal probability for all "legal" tiles matching the previous one.

[*] *SIAM J. Comput.* (1986) 15(1):285-286
[†] Supported by NSF grants MCS-8104211, 8304498.
[‡] Computer Science department, 111 Cummington St, Boston, MA 02215; (e-mail to Lnd@cs.bu.edu)

Proposition: *Tiling is an NP-complete random problem.*

Proof: Padding makes (μ_1, R_1) computable in time, say, $|x|^2$. We first reduce (μ_1, R_1) (no matter how special μ_1 may be) to a problem with "almost uniform" distribution. If $\mu \succeq \mu_1$ then $f(x) = x$ reduces (μ_1, R) to (μ, R). Thus, using $\mu_1(x) := \mu_1(x)/2 + 1/2 - 1/2x$, we get $\mu_1'(x) > 1/2x^2$. Now, by a linear number of successive roundings, μ_1 is replaced by a *perfectly rounded* $\mu > \mu_1/4$, which means that $\mu(x)$ is the shortest binary rational within $(\mu(x-1),\ \mu(x+1))$. As all perfectly rounded measures, μ has **integer** $m(x) = \mu(x)/\mu'(x)$ and $\log_2 \mu'(x)$. Monotone μ is computable and invertible (by binary search) in polynomial time. And so is m, since $\mu(x)/m(x) = \mu'(x)$ is a power of 2 and $1/2 < \mu(x) < 1$. The resulting probability of $z = m(x)$ is $\mu'(m^{-1}(z)) = \mu'(x) = \mu(x)/m(x) < 1/m(x) = 1/z$. Thus, $m(\mu)$ is "almost uniform".

Let p be the program for a universal Turing machine U with time bound $O(|x|^3)$ for which $R(x, y) = U(0^{|x|}1pm(x), y)$. Let $\lambda'(0^n 1s) = O(n^{-3})/s$ for $|s| < 3n$. Then $f(x) = 0^{|x|}1pm(x)$ reduces (μ, R) to (λ, U), since $\lambda'(f(x)) = \lambda'(0^{|x|}1pm(x)) = O(|x|^{-3})/pm(x) \succeq 1/m(x) \geq \mu'(x)$. Finally, (λ, U) is reducible to the tiling problem in a standard way: the tiled square corresponds to the space-time history of the Turing computation accepting $U(w, y)$, where w is chosen randomly and y is guessed non-deterministically. A tile letter represents either the tape symbol and the direction (left or right) to the head or the head state and the direction to the neighboring cell it looks at. ■

Corollary:
For every $\varepsilon < 0$, Tiling is polynomial on average iff it has a polynomial time algorithm for a $1 - n^{\varepsilon}$ fraction of instances (by the "padding" argument) and only if such are all NP random problems.

Random NP problems look like "fair games" between suppliers of questions and answers (if both are restricted to a polynomial-time probabilistic machine). Thus, their average hardness seems to be a more balanced question than "P=NP?".

Essential intuitive comments and an excellent survey of the area may be found in [1]. The author is grateful to R. Rivest for encouragement and discussion, to D. Johnson for valuable corrections and to U. Vazirani and Z. Galil for improvements of the account.

References

[1] David S. Johnson. The NP-Completeness Column: An Ongoing Guide. *Journal of Algorithms,* **5**, 284-299 (1984).