Feasible Functions*

Yuri Gurevich
Electrical Engineering and Computer Science
University of Michigan, Ann Arbor
MI 48109-2122, USA, gurevich@umich.edu

It is common to identify the class of computable functions with the class of functions computable by the Turing machine. The identification is done on the ground of the famous Church-Turing thesis. In fact, the thesis justifies only one implication: computable function are Turing computable. The other implication is quite obviously false if computability means practical computability. I am not trying to mock the Turing machine or recursion theory. They brought about a major advance in our understanding of computations. In particular, assuming the Church-Turing thesis, logicians were able to prove that some important decision problems, like Hilbert's Tenth Problem [M], are not solvable by any algorithm. But the fact is that some "computable in principle" functions are not computable *in principle* in any practical sense.

It is common to identify feasible functions (that is, feasibly computable functions) with those computable in polynomial time. Since a great many useful decision problems are NP [GJ], this identification makes the famous question P=?NP so central in complexity theory.

On the first glance, the claim PTime $\rightarrow$ feasible seems silly. The time complexity of computing a PTime function may have a terrible lower bound, like n^{1000}. No technological progress will allow us to compute such a function. Fortunately, PTime functions of practical interest tend to have low-polynomial time complexity. It seems reasonable that, given an n-bit input, we should be able, at least in principle, spend time $7n$, $n \log n$ or n^2 to work on it. One famous proponent of the thesis feasible $\leftrightarrow$ PTime is Steve Cook, of the University of Toronto. He and I debated the issue during the 1991

*London Mathematical Society Newsletter, No. 206, June 1993, 6–7.

Annual Meeting of the Association for Symbolic Logic. It would be only fair to represent here the arguments of both sides, but the format of this short article does not allow me to do so. Cook's position is reflected in the paper [C]. I describe briefly my objections to the thesis.

First, there are different kinds of feasibility. (There are also different degrees of feasibility. The proponents of Turing computability win a point: it is absolute.) What is feasible in one application may be infeasible in another. In some applications, time isn't the most important resource, and feasibility may be a function of memory size or money or a combination of various resources. Even if time is the only limiting resource, it may be more appropriate in some applications to measure time complexity in terms of the output size rather than the input size. For example, consider the task of printing out all primes $\leq N$ on input N (in decimal notation).

Second, let us examine the feasibility reflected in the thesis feasible $\leftrightarrow$ PTime. "For natural problems, PTime computability implies feasibility, but feasibility does not necessarily imply PTime computability", says Leonid Levin, a smart kid on the block who influenced this author [L]. I agree with the second part. Indeed, some non-polynomial bounds on computation time, e.g. bounds of the form $n^{c \log\log n}$, do not seem to contradict feasibility. Notice that $n^{\frac{1}{5}\log\log n} > n^2$ if and only if $n > e^{e^{10}}$. Bounds of the form $n^{c\log\log n}$ do appear in practice. The fastest known (deterministic) primality test runs in time $O(n^{c\log\log n})$ for some c (where n is the length of, say, decimal notation for the given number N) [APR] though I do not know any natural decision or search problem whose time complexity is provably of the form $n^{c\log\log n}$.

Many useful decision and optimization problems are known to be NP hard. Those problems are not decidable in PTime. But some such optimization problems are approximately solvable in PTime in a satisfying way [GJ], and some of those decision problems are decidable very quickly on average. For example, under the assumption that all n-vertex graphs are equally probably, the expected time of the backtracking algorithm for 3-colorability is ... constant. Specifically, "the average number of nodes in the backtrack search tree for this problem is about 197, averaged over graphs of all sizes" [W]. The explanation is simple. Statistically, the overwhelming majority of graphs is not 3-colorable and there are very simple witnesses for non-3-colorability, e.g., a clique of 4. If you design your algorithm to look for a 4-clique first, you can decrease the expected time substantially. Of course, not all examples are that simple [GS]. My point is that some presumably infeasible problems are solvable quickly on average, and of course the average running time may

be more important in practice than the worst-case time. Elsewhere we argued that the average-time version of the P=?NP question is preferable to the original one [G].

Finally, the weakest point of the feasible $\leftrightarrow$ PTime thesis is the asymptotic approach. In applications, one deals with input sizes in a certain range. This is, however, a weak point of almost all current complexity theory. A good theory of non-asymptotic complexity is a biggest challenge of all in the area.

References

[APR] L. M. Adleman, C. Pomerance and R. S. Rumely, "On Distinguishing Prime Numbers from Composite Numbers", Annals of Mathematics 117 (1983), 173–206.

[C] Stephen A. Cook, "Computational Complexity of Higher Type Functions", Proceedings of 1990 Intern. Congress of Mathematicians, Kyoto, Japan, Springer–Verlag, 1991, 55–69.

[GJ] Michael R. Garey and David S. Johnson, "Computers and Intractability", W. H. Freeman and Company, New York, 1979.

[G] Yuri Gurevich, "The Challenger-Solver game: Variations on the Theme of P=?NP", Bulletin of Euro. Assoc. for Theor. Computer Science, No. 39, Oct. 1989, 112–121.

[GS] Yuri Gurevich and Saharon Shelah, "Expected Computation Time for Hamiltonian Path Problem", SIAM J. on Computing 16:3 (1987), 486–502.

[L] Leonid Levin, Private correspondence.

[M] Yuri Matijasevich, "Recursively Enumerable Sets are Diophantine", Doklady 191:2 (1970), 279–282.

[W] Herbert S. Wilf. "Algorithms and Complexity", Prentice-Hall, 1986.