NP-completeness: A Retrospective

Christos H. Papadimitriou[*]

University of California, Berkeley, USA

Abstract. For a quarter of a century now, NP-completeness has been computer science's favorite paradigm, fad, punching bag, buzzword, alibi, and intellectual export. This paper is a fragmentary commentary on its origins, its nature, its impact, and on the attributes that have made it so pervasive and contagious.

1. A keyword search in Melvyl, the University of California's on-line library, reveals that about 6,000 papers each year have the term "NP-complete" on their title, abstract, or list of keywords. This is more than each of the terms "compiler," "database," "expert," "neural network," and "operating system." Even more surprising is the diversity of the disciplines with papers referring to "NP-completeness:" They range from statistics and artificial life to automatic control and nuclear engineering. What is the nature and extent of the impact of NP-completeness on theoretical computer science, computer science in general, computing practice, as well as other domains of the natural sciences, applied science, and mathematics? And why did NP-completeness become such a pervasive and influential concept?

2. One of the reasons of the immense impact of NP-completeness has to be the appeal and elegance of the class P, that is, of the thesis that "polynomial worst-case time" is a plausible and productive mathematical surrogate of the empirical concept of "practically solvable computational problem." But, obviously, NP-completeness also draws on the importance of NP, as it rests on the widely conjectured contradistinction between these two classes. In this regard, it is crucial that NP captures vast domains of computational, scientific, and mathematical endeavor, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly. True, there are domains, such as strategic analysis and counting, which have been within our computational ambitions, and still seem to lie outside NP; but they are the exceptions rather than the rule. NP-completeness has thus become a valuable intermediary between the abstraction of computational models and the reality of computational problems, grounding complexity theory to computational practice.

3. Also crucial for the success of NP-completeness has been its surprising ubiquity and effectiveness as a classification tool, and the scarcity of problems in

[*] christos@cs.berkeley.edu. Partially supported by the National Science Foundation. A version of this talk was given at a meeting in the Fall of 1995 celebrating the 60th birthday of Richard M. Karp, to whom this paper is also affectionately dedicated.

NP that resist classification as either polynomial-time solvable or NP-complete. (Ladner's result on intermediate degrees between P and NP-completeness [12] had been known almost as soon as NP-completeness was introduced, and thus theoretically the world could be full of mysterious intermediate problems.) In several occasions, extremely broad classes of computational problems in NP have been dichotomized with surprising accuracy into polynomially solvable and NP-complete, see [21, 22] for two early examples.

4. The founders of NP-completeness [2, 10, 13] appear to have anticipated its broad applicability and classification power. Leonid Levin [13] wrote in 1973: *"The method described here clearly provides a means for readily obtaining results of [this type] for the majority of important sequential search problems."* In Karp's paper [10] twenty one problems were proved NP-complete, showing beyond any doubt the surprisingly broad applicability of the method. Significantly, Karp seems annoyed and surprised that three other problems (linear programming, primality, and graph isomorphism) resisted at the time such classification. Primality and graph isomorphism were also mentioned by Cook [2]. Knuth was sufficiently convinced about the importance and broad applicability of the new concept to take early and deliberate action on the terminological front [11].

5. NP-completeness has had tremendous impact even in areas where, in some sense, it should not have. It is now common knowledge among computer scientists that NP-completeness is largely irrelevant to public-key cryptography, since in that area one needs sophisticated *cryptographic assumptions* that go beyond NP-completeness and worst-case polynomial-time computation [19]; furthermore, cryptographic protocols based on NP-complete problems have been ill-fated. Fortunately, the founders of modern cryptography did not know this. Diffie and Hellman base their famous pronouncement *"We stand today on the brink of a revolution in cryptography"* [3] on two facts: (1) Very fast hardware and software, and (2) novel techniques for proving problems hard (they cite Karp's paper [10]).

6. NP-completeness has also exhibited a great amount of versatility, adapting to contexts and computational aspects beyond its original scope of worst-case analysis of exact algorithms for decision and optimization problems. For example, it was used early on to show that certain optimization problems cannot be approximated satisfactorily [20], and indeed in a most ingenious and comprehensive way more recently [1]. By showing that even less ambitious goals than worst-case polynomial exact solution are unattainable, NP-completeness is thus a most useful tool for repeatedly pruning unpromising research directions and thus redirecting research to new ones (in a manner reminiscent of the struggle between Hercules and the monster Hydra [16]).

7. Let me illustrate this versatility of NP-completeness by a technical interlude on an aspect of efficient computation that has interested me recently, namely, *output polynomial time.* Certain computational problems require an output $f(x)$ on input x that is in the worst case exponential in the input. For such problems, one would like to have algorithms that are polynomial in $|x|$ *and* $|f(x)|$. The class of problems thus solvable can be called *output polynomial time*. One can use NP-completeness to prove that certain functions are not in output-polynomial time, unless P=NP. For example, consider the function MIN which maps a regular expression to the minimum-state equivalent *deterministic* finite-state automaton. MIN can be computed by first designing a nondeterministic automaton M, then an equivalent deterministic automaton M', and next minimizing the states of M' to obtain the final output; the problem is, of course, that the intermediate result M' could be exponential in *both* the input and the output. It is rather straightforward to use "traditional" NP-completeness techniques to show the following:

Theorem 1. *Unless P=NP, MIN is not in output polynomial time.*

In fact, we cannot even compute in output-polynomial time a deterministic automaton that has *at most polynomially more states than the minimum* —unless, of course, P=NP.

8. Often the required output $f(x)$ is a set $\{y_1, \ldots, y_k\}$ of strings that are related to x via an NP mapping; for example, if G is a graph, let AMIS(G) be the set of all maxim*al* independent sets of G. AMIS is known to be in output-polynomial time (see [9] for an exposition and strengthening of this result, and an early discussion of output polynomial time). For such problems we have an elegant alternative definition of output polynomial time. A function $f : \Sigma^* \mapsto 2^{\Sigma^*}$ is in output polynomial time if the following problem is solvable in polynomial time: Given x and $y \subseteq \Sigma^*$, either decide that $y = f(x)$, or find a string in $y \oplus f(x)$. It is easy to see that, if such an algorithm exists, then its iteration starting with $S = \emptyset$ gives an output polynomial time algorithm for f; and vice-versa, if an output polynomial time algorithm exists for f, it can be used to produce an element of $y \oplus f(x)$. For example, AMIS is in output polynomial time; its generalization to hypergraphs is open, but was recently shown to be in output $n^{c \log n}$ time [6]; see [5] for an extensive discussion of the hypergraph generalization of AMIS. One can use again "traditional" NP-completeness to show that the following generalization is *not* in output polynomial time, unless P=NP: Given a monotone circuit, compute the set of all minimal (with respect to the set of true inputs) satisfying truth assignments.

9. But, sometimes, "traditional" NP-completeness techniques do not seem to suffice to bring out the intractability of a problem, because this problem belongs to a class or computational mode that appears to be "between" P and NP. In such cases NP-completeness has acted as an open-ended research paradigm, spawning variants that are appropriate for the computational context being studied; examples are classes that capture local search [8], the parity argument [14], logarithmic nondeterminism [18], the related concept of fixed-parameter tractability [4], and approximability [17].

10. Complexity classes introduced this way, as abstractions of natural computational problems of mysteriously intermediate complexity, are in some precise sense well-motivated, indeed necessary; they are *discovered*, not *invented*, as they have always existed by dint of their natural complete problems. The only way to make them go away is to *collapse them with P or NP* —as occasionally happens, recall [17] and its brilliant follow-up [1].

11. NP-completeness is of course a valuable tool for demonstrating the difficulty of computational problems. However, NP-completeness is often used "allegorically;" a problem is shown NP-complete that is not, strictly speaking, a natural computational problem, but an artificial problem created to capture a mathematical concept. NP-completeness in this context suggests that a problem, area, or approach is *mathematically nasty.*. Because, if we believe that efficient algorithms are the natural outflow of the mathematical structure of a problem (a view shared by all computer scientists, with the possible exception of researchers in "metaphor-based" algorithmic paradigms such as neural nets, in which algorithmic behavior is thought to be "emergent"), then, contrapositively, complexity must be the manifestation of mathematical poverty, lack of structure. See [7] for an early example of such a use of NP-completeness in the theory of relational databases.

12. Beyond mathematics, NP-completeness (and complexity in general) can also be applied "allegorically" in other disciplines. It can be used as a metaphor for chaos in dynamical systems, for unbounded rationality in game theory, for unfairness in economics, for integrity of electoral systems in political science, for cognitive implausibility in artificial intelligence, for genetic indeterminism in genetics, and so on (see [16] for references).

13. NP-completeness is thus an important "intellectual export" of computer science to other disciplines. And it does fill a void in the interdisciplinary intellectual trade: It seems to me that the concept of lower bounds —and negative results in general— is particular to computer science, and has no well-developed counterpart in other disciplines. True, one sees isolated results in other sciences (such as Heisenberg's uncertainty principle in quantum mechanics, Arrow's impossibility theorem in economics, and Carnot's theorem in thermodynamics) which are arguably negative; however, nowhere else in science does one find such a comprehensive methodology for obtaining negative results (with the exception of complexity's own precursor mathematical logic, with its many incompleteness, undecidability, and inexpressibility results). NP-completeness is therefore valuable for another reason: It is one of the few precious features which give our science its special character, which set it apart from the other sciences (see [15] for another development of this argument).

14. In science, successful ideas are those that are pervasive and invasive, are invitingly elegant and methodical, are open to extensions and variants, and capture an objective necessity, answer a widespread but diffuse sense of dissatisfaction in the scientific community (in the case of NP-completeness, the widespread feeling among computer scientists in the 1960s that automata theory, the previous great paradigm, had run its course as a useful abstraction of computation). Thinking about the nature and history of NP-completeness could give us useful hints about computer science's next great paradigm, which, for all I know, has started being articulated somewhere else in this volume.

References

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, "Proof verification and hardness of approximation problems." *Proc. 33rd FOCS* (1992) pp. 14–23.
2. S. A. Cook "The complexity of theorem-proving procedures," *Proc. 3rd STOC,* (1971), pp. 151–158.
3. W. Diffie and M. E. Hellman "New directions in cryptography," *IEEE Trans. Inform. Theory, 22*, pp. 644–654, 1976.
4. R. G. Downey and M. R. Fellows "Fixed-parameter tractability and completeness I: Basic results," *SIAM Journal on Computing, 24,* 4, pp. 873-921, 1995.
5. T. Eiter, G. Gottlob "Identifying the minimal transversals of a hypergraph and related problems" *SIAM Journal on Computing, 24*, 6, pp. 1278-1304, 1995.
6. M. Fredman and L. Khachiyan "On the complexity of dualization of monotone disjunctive normal forms" *Journal of Algorithms, 21*, 3, pp. 618–628, 1996.
7. P. Honeyman, R. E. Ladner, M. Yannakakis, "Testing the universal instance assumption," *Information Processing Letters, 12*, pp. 14–19, 1980.
8. D. S. Johnson, C. H. Papadimitriou, M. Yannakakis "How Easy is Local Search?" *J.CSS,* 1988 (special issue for the 1985 FOCS Conference).
9. D. S. Johnson, C. H. Papadimitriou, M. Yannakakis "On Generating All Maximal Independent Sets", *Information Processing Letters* 1988.
10. R. M. Karp "Reducibility among combinatorial problems," pp. 85–103 in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (eds), 1972.
11. D. E. Knuth "A terminological proposal," *SIGACT News, 6*, 1, pp. 12–18, 1974.
12. R. E. Ladner "On the structure of polynomial time reducibility," *J.ACM, 22*, pp. 155–171, 1975.
13. L. Levin "Universal sorting problems," *Pr. Inf. Transm., 9*, p¿ 265–266, 1973.
14. C. H. Papadimitriou "On the Complexity of the Parity Argument and other Inefficient Proofs of Existence" *JCSS, 48*, 3, 498–532, 1994.
15. C. H. Papadimitriou "Database metatheory: asking the big queries," *Proc. 1995 PODS Conf.*, reprinted in *SIGACT News*, spring 1996.
16. C. H. Papadimitriou "The complexity of knowledge representation," *Proc. 1996 Computational Complexity Symposium.*
17. C. H. Papadimitriou, M. Yannakakis "Optimization, approximation, and complexity classes" *Proc. 1988 STOC*, and *J.CSS,*, 1991.
18. C. H. Papadimitriou, M. Yannakakis "On limited nondeterminism and the complexity of the Vapnic-Chervonenkis dimension," special issue of *J.CSS* 1996 (special issue for the 1993 Structures Conf.).
19. R. L. Rivest "Cryptography," pp. 717–755 in *Handbook of Theoretical Computer Science,* J. van Leeuwen (ed), The MIT Press/Elsevier, 1990.
20. S. Sahni, T. Gonzalez "P-complete approximation problems," *J.ACM, 23,* pp. 555–565, 1976.
21. T. J. Schaeffer "The complexity of satisfiability problems," *Proc. 10th STOC,* (1978), pp. 216–226.
22. M. Yannakakis "Node- and edge-deletion problems," *Proc. 10th STOC,* (1978), pp. 253–264.

This article was processed using the LaTeX macro package with LLNCS style