The Importance of the P versus NP Question

STEPHEN COOK

University of Toronto, Toronto, Ont., Canada

The **P** versus **NP** problem is to determine whether every language accepted by some nondeterministic Turing machine in polynomial time is also accepted by some deterministic Turing machine in polynomial time. Unquestionably this problem has caught the interest of the mathematical community. For example, it is the first of seven million-dollar "Millennium Prize Problems" listed by the Clay Mathematics Institute [www.claymath.org]. The Riemann Hypothesis and Poincaré Conjecture, both mathematical classics, are farther down the list. On the other hand, Fields Medalist Steve Smale lists **P** versus **NP** as problem number three, after Riemann and Poincaré, in "Mathematical Problems for the Next Century" [Smale 1998].

But **P** versus **NP** is also a problem of central interest in computer science. It was posed thirty years ago [Cook 1971; Levin 1973] as a problem concerned with the fundamental limits of feasible computation. Although this question is front and center in complexity theory, **NP**-completeness proofs have become pervasive in many other areas of computer science, including artificial intelligence, databases, programming languages, and computer networks (see Garey and Johnson [1979] for 300 early examples).

If the question is resolved, what would be the consequences? Consider first a proof of $\mathbf{P} = \mathbf{NP}$. It is possible that the proof is nonconstructive, in the sense that it does not yield an algorithm for any **NP**-complete problem. Or it might give an impractical algorithm, for example, running in time n^{100}. In either of these cases, the proof would probably have few practical consequences other than to disappoint complexity theorists. However, experience has shown that when natural problems are proved to be in **P**, a feasible algorithm can be found. There are potential counterexamples to this assertion; most famously, the deep results of Robertson and Seymour [1993–1995], who prove that every minor closed family of graphs can be recognized in time $O(n^3)$, but their algorithm has such huge constants it is not practical. But practical algorithms are known for some specific minor-closed families (such as planar graphs), and possibly could be found for other examples if sufficient effort is expended.

If $\mathbf{P} = \mathbf{NP}$ is proved by exhibiting a truly feasible algorithm for an **NP**-complete problem such as SATISFIABILITY (deciding whether a collection of propositional clauses has a satisfying assignment), the practical consequences would be stunning. First, most of the hundreds of problems shown to be **NP**-complete can be efficiently reduced to SATISFIABILITY, so many of the optimization problems important to industry could be solved. Second, mathematics would be transformed, because computers could find a formal proof of any theorem which has a proof of reasonable length. This is because formal proofs (say in Zermelo–Fraenkel set theory) are

---

This is mostly abstracted from the author's article "The **P** versus **NP** Problem," available at www.claymath.org/prizeproblems/pvsnp.htm.

easily recognized by efficient algorithms, and hence bounded proof existence is in **NP**. Although the formal proofs may not be intelligible to humans, the problem of finding intelligible proofs would be reduced to that of finding a good recognition algorithm for formal proofs. Similar remarks apply to the fundamental problems of artificial intelligence: planning, natural language understanding, vision, and even creative endeavors such as composing music and writing novels. In each case, success would depend on finding good algorithms for recognizing good results, and this fundamental problem itself would be aided by the SAT solver by allowing easy testing of recognition theories.

One negative consequence of a feasible proof that $\mathbf{P} = \mathbf{NP}$ is that complexity-based cryptography would become impossible. The security of the Internet, including most financial transactions, depends on assumptions that computational problems such as large integer factoring or breaking DES (the Data Encryption Standard) cannot be solved feasibly. All of these problems are efficiently reducible to SATISFIABILITY. (On the other hand, quantum cryptography would survive a proof of $\mathbf{P} = \mathbf{NP}$ and might solve the Internet security problem.)

Now consider the consequences of a proof that $\mathbf{P} \neq \mathbf{NP}$. Such a proof might just answer the most basic of a long list of important related questions that could keep complexity theorists busy far in the future. How large is the time lower bound for SATISFIABILITY: is it barely superpolynomial or is it truly exponential, or is it in between? Does it apply just for the worst case inputs, or are there convincing average case lower bounds [Levin 1986; Gurevich 1991]? What about lower bounds for **NP** approximation problems [Vazirani 2001]? Are there lower bounds for problems such as integer factorization that are reducible to **NP** problems but may not be **NP**-hard? In general, proving the security of cryptographic protocols such as RSA or DES is much harder than proving $\mathbf{P} \neq \mathbf{NP}$.

Most complexity theorists, including the author, believe that $\mathbf{P} \neq \mathbf{NP}$ (see Gasarch [2002] for a recent poll). I would summarize the argument in favor of $\mathbf{P} \neq \mathbf{NP}$ by saying that we are really good at inventing efficient algorithms, but really bad at proving algorithms don't exist. There are powerful techniques that are part of the standard undergraduate computer science curriculum for devising efficient algorithms for diverse problems. Millions of smart people, including engineers and programmers, have tried hard for many years to find a provably efficient algorithm for one or more of the 1000 or so **NP**-complete problems, but without success.

Contrast this with the efforts of the small set of mathematicians who seriously work on proving $\mathbf{P} \neq \mathbf{NP}$. There are reasons why the main techniques tried for proving complexity lower bounds may not work for showing $\mathbf{P} \neq \mathbf{NP}$: a proof based on diagonalization cannot relativize [Baker et al. 1975], and a proof based on Boolean circuit lower bounds cannot be "natural" [Razborov and Rudich 1997]. Further, there are natural complexity class separations that we know exist but we cannot prove. Consider the sequence of complexity class inclusions

$$\mathbf{LOGSPACE} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}.$$

A simple diagonal argument shows that the first is a proper subset of the last, so it follows that one of the three adjacent inclusions must be proper. But no proof is known that any particular one is proper.

Assuming that $\mathbf{P} \neq \mathbf{NP}$, when and if will a proof be found? Apparently by the year 2100, if one believes the majority opinion from the poll [Gasarch 2002]. It

is difficult to say whether much progress has been made to date, since there is no convincing program toward finding a proof. There are recent beautiful results in complexity theory involving probabilistically checkable proofs [Arora et al. 1998] and derandomization [Impagliazzo et al. 1999] which create deep incites into the nature of computation, and it is nice to think that these ideas will someday contribute to a proof of $\mathbf{P} \neq \mathbf{NP}$.

REFERENCES

ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M. 1998. Proof verification and the hardness of approximation problems. *J. ACM 45*, 3 (May), 501–555.

BAKER, T., GILL, J., AND SOLOVAY, R. 1975. Relativizations of the P =? NP question. *SICOMP: SIAM J. Comput.*

COOK, S. 1971. The complexity of theorem-proving procedures. In *Conference Record of 3rd Annual ACM Symposium on Theory of Computing*. ACM New York, pp. 151–158.

GASARCH, W. 2002. Guest column: The P=?NP poll. *SIGACT NEWS 33*, 2 (June), 34–47.

GAREY, M. R., AND JOHNSON, D. S. 1979. *Computers and Intractibility, a Guide to the Theory of NP-Completeness*. Freeman, San Francisco, Calif.

GUREVICH, Y. 1991. Average case completeness. *J. Comput. Syst. Sci. 14*, 3 (June), 346–398.

IMPAGLIAZZO, R., SHALTIEL, R., AND WIGDERSON, A. 1999. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the 40th Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 181–190.

LEVIN, L. 1973. Universal search problems (in Russian). *Problemy Peredachi Informatsii 9*, 3, 265–266. (English translation in Trakhtenbrot, B. A.: A survey of Russian approaches to Perebor (brute-force search) algorithms. *Ann. Hist. Comput.* 6 (1984), 384–400.)

LEVIN, L. 1986. Average case complete problems. *SIAM J. Comput. 15*, 285–286.

RAZBOROV, A. A., AND RUDICH, S. 1997. Natural proofs. *J. Comput. Syst. Sci. 55*, 1 (Aug.), 24–35.

ROBERTSON, N., AND SEYMOUR, P. D. 1983–1995. Graph minors i–xiii. *J. Combinat. Theory B*.

SMALE, S. 1998. Mathematical problems for the next century. *MATHINT: The Mathematical Intelligencer 20*.

VAZIRANI, V. 2001. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany.