The Importance of the P versus NP Question[1]

Stephen Cook
University of Toronto

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" [Sma98].

But **P** versus **NP** is also a problem of central interest in computer science. It was posed thirty years ago [Coo71, Lev73] 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, data bases, programming languages, and computer networks (see [GJ79] for 300 early examples).

If the question is resolved, what would be the consequences? Consider first a proof of **P**=**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 [RS95]. They prove that every minor closed family of graphs can be recognized in time $O(n^3)$,

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

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 **P**=**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 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 **P**=**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 **P**=**NP**, and might solve the Internet security problem.)

Now consider the consequences of a proof that **P**≠**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 SATISFIABILILITY: is it barely super polynomial 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 [Lev86, Gur91]? What about lower bounds for **NP** approximation problems [Vaz01]? 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 [Gas02] 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 which 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 [BGS75], and a proof based on Boolean circuit lower bounds cannot be "natural" [RR97]. Further, there are natural complexity class separations which 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 [Gas02]. 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 [ALM$^+$98] and derandomization [ISW99] which create deep insights 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

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SICOMP: SIAM Journal on Computing*, 1975.

[Coo71] Stephen Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[Gas02] William Gasarch. Guest column: The P=?NP poll. *SIGACT NEWS*, 33(2):34–47, June 2002.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractibility, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, 1979.

[Gur91] Yuri Gurevich. Average case completeness. *Journal Computer and System Sciences*, 14(3):346–398, June 1991.

[ISW99] Russell Impagliazzo, Ronen Shaltiel, and Avi Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the 40th FOCS*, pages 181–190, 1999.

[Lev73] L. Levin. Universal search problems (in russian)". *Problemy Peredachi Informatsii*, 9(3):265–266, 1973. English translation in Trakhtenbrot, B. A.: A survey of Russian approaches to Perebor (brute-force search) algorithms. Annals of the History of Computing, 6 (1984), pages. 384-400.

[Lev86] L. Levin. Average case complete problems. *SIAM J. Computing*, 15:285–286, 1986.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, August 1997.

[RS95] N. Robertson and P. D. Seymour. Graph minors I–XIII. *Journal of Combinatorial Theory B*, 1983–1995.

[Sma98] Steve Smale. Mathematical problems for the next century. *MATHINT: The Mathematical Intelligencer*, 20, 1998.

[Vaz01] Vijay Vazirani. *Approximation Algorithms.* Springer-Verlag, Berlin, 2001.