P Versus *NP*

Richard M. Karp

Computational complexity theory is the branch of theoretical computer science concerned with the fundamental limits on the efficiency of automatic computation. It focuses on problems that appear to require a very large number of computation steps for their solution. The inputs and outputs to a problem are sequences of symbols drawn from a finite alphabet; there is no limit on the length of the input, and the fundamental question about a problem is the rate of growth of the number of required computation steps as a function of the length of the input.

Some problems seem to require a very rapidly growing number of steps. One such problem is the *independent set problem*: given a graph, consisting of points called vertices and lines called edges connecting pairs of vertices, a set of vertices is called *independent* if no two vertices in the set are connected by a line. Given a graph and a positive integer *n*, the problem is to decide whether the graph contains an independent set of size *n*. Every known algorithm to solve the independent set problem encounters a combinatorial explosion, in which the number of required computation steps grows exponentially as a function of the size of the graph. On the other hand, the problem of deciding whether a given set of vertices is an independent set in a given graph is solvable by inspection. There are many such dichotomies, in which it is hard to decide whether a given type of structure exists within an input object *(the existence problem)*, but it is easy to decide whether a given structure is of the required type *(the verification problem)*.

It is generally believed that existence problems are much harder to solve than the corresponding verification problems. For example, it seems hard to decide whether a jigsaw puzzle is solvable, but easy to verify that a given arrangement of the puzzle pieces is a solution. Similarly, it seems hard to solve Sudoku puzzles but easy to verify given solutions. Complexity theory provides precise definitions of *"P"*, the class of all existence problems that are easy to solve, and *"NP"*, the class of existence problems whose solutions are easy to verify. The general belief that verifying is easier than solving strongly suggests that the class *NP* properly includes the class *P*, but this claim has never been proven. The question of whether $P = NP$ is the most central open question in theoretical computer science, and one of the most notorious open questions in all of mathematics.

In a 1972 paper entitled "Reducibility Among Combinatorial Problems" I demonstrated a technique that has made it possible to prove that thousands of problems, arising in mathematics, the sciences, engineering, commerce and everyday life, are equivalent, in the sense that an efficient algorithm for any one of them would yield efficient algorithms for all the problems in *NP*, and thus establish that $P =$

NP. Conversely, if *P* is unequal to *NP*, then none of these problems are easy to solve. These problems are called *NP-complete*. The moral of the story is that *NP*-completeness is a widespread phenomenon; most combinatorial problems arising in practice are *NP*-complete, and hence, in all likelihood, hard to solve.

The technique stems from a 1971 paper in which Stephen Cook of the University of Toronto showed that a particular problem in *NP*, the *Satisfiability Problem* (denoted *Sat*) of propositional logic is *NP*-complete. To do so, he showed that any problem in *NP* is efficiently reducible to *Sat*; *i.e.*, for any problem *A* in *NP*, there is an efficient algorithm that converts any instance of *A* into an equivalent instance of *Sat*. It follows that, if *Sat* is easy to solve, then every problem in *NP* is easy to solve. Around the same time, Leonid Levin in the Soviet Union, who is now a professor at Boston University, proved a similar result.

In my 1972 paper I demonstrated the prevalence of *NP*-completeness by using a tree of efficient reductions to show that *21* canonical problems are *NP*-complete. The figure exhibits reductions among 13 of these problems. Each node of the tree is labeled with the name of problem in *NP*, and each edge indicates that the upper problem is efficiently reducible to the lower one; thus, if the lower problem is easy to solve, then so is the upper problem. Hence, if any problem in the tree were easy to solve, then *Sat* would be easy to solve and therefore, by Cook's seminal result, every problem in *NP* would be easy to solve.

The prevailing, but not quite universal, opinion among complexity theorists is that *P* is unequal to *NP*, but no proof or disproof appears to be on the horizon. Perhaps some brilliant youngster, motivated by this essay, will find the elusive approach that will crack the *P* vs. *NP* problem.

Beauty in mathematics can be found at many levels: in the symmetry and elegance of mathematical curves, surfaces and combinatorial structures; in the subtle logic of mathematical proofs; or, as in the case of NP-completeness, in the discovery that a large number of seemingly unrelated mathematical phenomena are all manifestations of a single underlying principle.