Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of the Franklin Institute 342 (2005) 629–637
Journal of The Franklin Institute
www.elsevier.com/locate/jfranklin

The 2004 Benjamin Franklin Medal in Computer and Cognitive Science presented to Richard M. Karp

Bruce Char*

Department of Computer Science, Drexel University, Philadelphia, PA 19104, USA

Abstract

The Benjamin Franklin Medal in Computer and Cognitive Science is awarded to Dr. Richard M. Karp for his contributions to the understanding of computational complexity. His work helps programmers find workable solution procedures avoiding approaches that would fail to find a solution in a reasonable amount of time. Scientific, commercial, or industrial situations where his work applies include establishing least-cost schedules for industrial production, transportation routing, circuit layout, communication network design, and predicting the spatial structure of a protein from its amino acid sequencing.

Keywords: Computational complexity; Polynomial time algorithms; Theory of computation; NP-completeness; Combinatorial problems

1. Introduction and background

1.1. The history and context for Richard Karp's work on computational complexity

With the advent of "high-speed" electronic computing starting from World War II, many problems that had previously been too laborious to solve through calculation became feasible. Some of these problems involved finding approximations to the

*Tel.: +1 215 895 2670; fax: +1 215 895 1582.
E-mail address: charbw@drexel.edu.

0016-0032/$30.00 © 2005 Published by Elsevier Ltd. on behalf of The Franklin Institute.
doi:10.1016/j.jfranklin.2005.04.009

solutions of problems in continuous mathematics, for example, those involving solution of systems of algebraic or differential equations. Others involved "combinatorial optimization". Richard Karp himself described these problems in his 1985 Turing Award lecture:

> Combinatorial search problems … can be likened to jigsaw puzzles where one has to assemble the parts of a structure in a particular way. Such problems involve searching through a finite, but extremely large, structured set of possible solutions, patterns, or arrangements, in order to find one that satisfies a stated set of conditions. Some examples of such problems are the placement and interconnection of components on an integrated circuit chip, the scheduling of the National Football League, and the routing of a fleet of school buses. Within any one of these combinatorial puzzles lurks the possibility of a combinatorial explosion.
>
> Because of the vast, furiously growing number of possibilities that have to be searched through, a massive amount of computation may be encountered unless some subtlety is used in searching through the space of possible solutions [1].

Thoughtful work on each kind of problem led to advances—more efficient algorithms for solving them. This allowed faster solution finding, or the ability to handle larger versions of the problems (scheduling more teams, or figuring out how to interconnect chips with more components). However, those who worked on solving combinatorial problems realized that some seemed to admit no particularly efficient general techniques for solving them. However, rigorously establishing and characterizing this difficulty required a theory of computation to handle this.

It was known from Alan Turing's "uncomputability" result [2] that there were some problems which lacked computer algorithms to solve them. Yet this result seemed primarily of interest to theoreticians. Combinatorial problems typically can be solved by enumerating and trying all possible ways of combining the entities involved, and seeing which one solves the problem. Since there are only a finite (albeit very large) number of possibilities, this approach always produces a solution algorithm for combinatorial problems. However, from a practical viewpoint the number of possibilities is so large, even for relatively modestly sized problems, that such algorithms would take more time than the age of the universe, even if run on an ultra-fast computer with many processors running in parallel. What was needed from the theory of computation was some way of determining if these combinatorial problems were inherently difficult (even if not impossible), or whether there was a yet-unfound technique that could make a solution of such problems tractable. Computer scientists working in the theory of computation, in addition to continuing to find new or improved algorithms to problems, started spending time trying to characterize the differences in problems that made some easier (faster) to solve than other kinds of problems in hope of making the search for improved algorithms more sophisticated.

During the 1950s and the 1960s, the theory of computing had focused on the capabilities of a few mathematically specified models of automata: finite automata (regular sets), pushdown automata (context-free languages), and Turing Machines