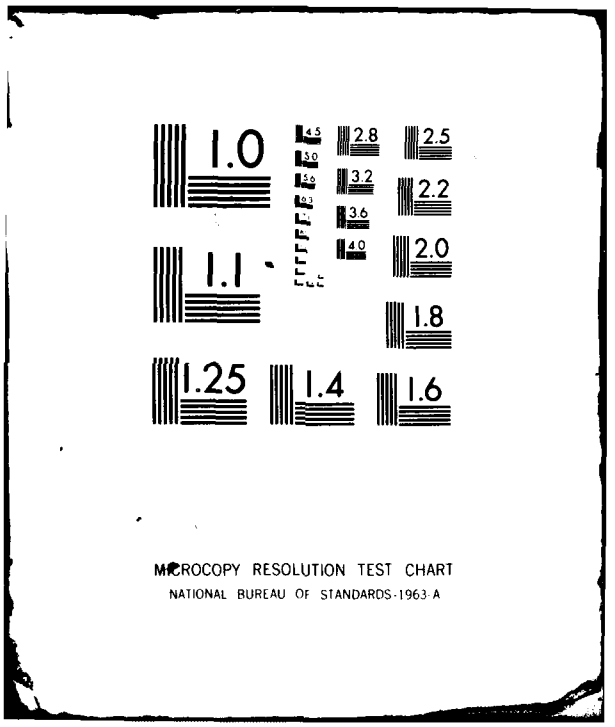
AD-A089 177 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 12/1
COMPUTATIONAL COMPLEXITY,(U)
JUL 80 J F TRAUB N00014-76-C-0370
UNCLASSIFIED NL

END
DATE FILMED
10 80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DEPARTMENT of COMPUTER SCIENCE

Columbia University

LEVEL II

(6) COMPUTATIONAL COMPLEXITY,

(10) Joseph F./Traub
Department of Computer Science
Columbia University
New York, N.Y. 10027

(11) Jul[redacted]80

(12) 13

To appear in
ENCYCLOPEDIA OF COMPUTER SCIENCE

This research was supported in part by the National Science Foundation under Grant MCS-7823676 and the Office of Naval Research under Contract N00014-76-C-0370, NR 044-422.

On November 7, 1979 the New York Times carried a front page story headlined

"SOVIET DISCOVERY ROCKS WORLD OF MATHEMATICS"

According to the story, which had been picked up from the magazine Science, "apart from its profound theoretical interest, the discovery may be applicable in weather prediction, complicated industrial processes, petroleum refining, the scheduling of workers at large factories, secret codes and many other things". The New York Times coverage generated substantial controversy concerning the merits of the discovery, a new algorithm for doing linear programming. Near the end of this article we will evaluate this algorithm.

The New York Times story concerned the "computational complexity" of the linear programming problem. Although computational complexity does not always make front page news, it is a very active and important research area. Its subject is the determination of the intrinsic difficulty of mathematically posed problems arising in many disciplines. The study of complexity has led to more efficient algorithms than those previously known or suspected. We illustrate some of the important ideas of computational complexity with the example of matrix multiplication.

COMPUTATIONAL COMPLEXITY OF MATRIX MULTIPLICATION. We consider first the multiplication of 2 x 2 matrices. Let

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix},$$

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.$$

Given A, B, we seek C = AB .

The classical algorithm computes C by

$$c_{11} = a_{11}\, b_{11} + a_{12}\, b_{21}, \text{ etc.}$$

at a cost of eight multiplications.

Until the late sixties no one seems to have been asked whether two matrices could be multiplied in less than eight scalar multiplications. Then Strassen showed that 7 scalar multiplications are sufficient.

Consider next the multiplication of N x N matrices. The classical algorithm uses N^3 arithmetic operations. (In this article we disregard constants in giving the cost of our algorithm.) By repeated partitioning of N by N matrices into 2 by 2 submatrices, two matrices can be multiplied in $N^{\log_2 7} \sim N^{2.81}$ arithmetic operations.

After a decade during which there was practically no progress on decreasing the number of arithmetic operations used in matrix multiplication, Schönhage and Pan showed $N^{2.52}$ arithmetic operations are sufficient. The number 2.52 is just the current state of our knowledge and researchers expect the exponent will be further decreased.

We must emphasize that the above results are of theoretical rather than practical value. The value of N has to be enormous before the new algorithm would be faster than the classical one. On the other hand, there are some problems for which new algorithms have had profound influence. A good example is provided by the finite Fourier transform on N points. The fast Fourier transform uses only N log N arithmetic operations compared to N^2 for the classical algorithms. Since N log N is much smaller than N^2 for even moderate values of N and since the finite Fourier tranform is often needed for a large number of points, the introduction of the fast Fourier transform has revolutionized a number of scientific fields.

Using the matrix multiplication example we introduce some basic terminology.

The minimal number of arithmetic operations is called the computational complexity (or problem complexity) of the matrix multiplication problem. We often write complexity for brevity.

The complexity of matrix multiplication is unknown. An upper bound is $N^{2.52}$. A lower bound is N^2. Since this lower bound is linear in the number of inputs and outputs we say it is trivial lower bound. No non-trivial lower bound is known.

Algorithm complexity is the cost of a particular algorithm. This should be contrasted with problem complexity which is the minimal cost over all possible algorithms. People who do not work in complexity theory sometimes confuse these two terms.

Fast Algorithm is a qualitative term meaning faster than a classical algorithm or faster than previously known algorithms. An optimal algorithm is one whose complexity equals the problem complexity.

Table 1 summarizes the present state of our knowledge concerning matrix multiplication.

SUMMARY OF MATRIX MULTIPLICATION

upper bound	$N^{2.52}$
lower bound	N^2
complexity	unknown
optimal algorithm	unknown

TABLE 1

COMPUTATIONAL COMPLEXITY IN GENERAL. We define a model of computation stating which "operations" or "steps" are permissible and how much they cost. In the model we can ask the same questions as in the matrix multiplication example. For instance we seek:

problem complexity
upper bounds
lower bounds
fast algorithms
optimal algorithms

Typically, an upper bound is the cost of an algorithm for solving the problem. A lower bound must be established thru a theorem that states there does not exist an algorithm whose cost is less than the lower bound. Not surprisingly, lower bounds are far harder to establish than upper bounds. (An exception is provided by analytic computational complexity where a very powerful adversary principle provides lower bounds. See discussion below.)

SOME MODELS OF COMPUTATION. Numerous models of computation have been studied. In our matrix multiplication example we counted arithmetic operations. In the study of combinatorial problems we count comparisons. Very significant results have been obtained for space and time complexity in a Turing Machine model. Another important model is a random access machine (RAM). Other models are appropriate for studying parallel, asynchronous, or VLSI computation.

Often we assign a "size" N to a problem. If the number of operations or steps required to solve a problem is an exponential function of N we say the problem has exponential time complexity. If the problem requires a number of operations which is a polynomial function of N we say the problem has polynomial time complexity.

An elegant theory for an abstract complexity model is based on two axioms. One result in this abstract setting is the famous "Speed-up Theorem" which implies there cannot be a fastest algorithm for a computable function.

TYPICAL APPLICATIONS OF COMPUTATIONAL COMPLEXITY. The complexity of numerous problems has been studied. To illustrate the variety of problems we exhibit a dozen drawn from various areas.

1. Compute the finite Fourier transform at N points.

2. Determine if an N digit integer is prime; if not determine the factors.

3. Solve an elliptic partial differential equation to within ε

4. Compute the Kendall rank correlation at N points.

5. Generate a funtion to error less than ε from values of the function at N points.

6. Multiply two polynomials of degree N.

7. Prove all theorems of length less than N in a certain axiom system.

8. Solve the traveling salesman problem on N cities.

9. Solve to within ε a large sparse linear system of order N whose matrix is positive, definite, and has condition number bounded by M.

10. Find the closest neighbor of P points in K dimensions.

11. Compute the first N terms of the Qth composite of a power series.

12. Compute the first N digits of π (for, say, N = 20,000,000).

REDUCIBILITY AMONG PROBLEMS. There are many problems for which the best algorithm known costs exponential time. Such problems occur in operations research, computer design, data manipulation, graph theory and mathematical logic. Do there exist faster algorithms which solve these problems in polynomial time? We don't know. What we do know is that there is a large class of problems which are equivalent in that if one of them can be solved in polynomial time, they all can.

For technical reasons this class of problems is said to be NP-complete . Because no one has succeeded in devising a polynomial time algorithm for any of these problems, many researchers believe that NP-complete problems are exponentially hard. There is no proof of this and settling this question is one of the most important open problems in computational complexity.

ANALYTIC COMPUTATIONAL COMPLEXITY. Many problems can be only approximately solved. Examples are optimal estimation, search for an extremum, calculation of fixed points, and solution of partial differential equations. Indeed, most problems occuring in mathematics, science, engineering, and economics can be only approximately solved. On the other hand, to lower the cost we may choose to approximately solve problems which could be exactly solved. Important examples are provided by the approximate solution of NP-complete problems and the iterative solution of large sparse linear systems. Analytic Computational Complexity is the study of optimal algorithms for problems which are solved approximately.

A central notion in Analytic Complexity is that of "information". The optimality properties of a class of algorithms depend only on the information used by the algorithms. This permits an enormous simplification in the theory of optimal algorithms.

Often, but not always, only partial information is available. Partial information means the problem is not uniquely specified by the information. Partial information is important because

1. it is typical of "real world" problems
2. algorithms often utilize only partial information

We list some of the questions studied in analytic complexity.

1. Given certain information and a positive number ε, is the information "strong" enough to solve the problem to within ε ?
2. Is "adaptive information" more powerful than "non-adaptive information"?
3. Given a problem, what is the "optimal information" for its solution?
4. Given a problem, what is the optimal algorithm for its solution?
5. Given a problem, what is its complexity?(In analytic complexity this is often known to within very tight bounds.)
6. Given two problems, which is intrinsically more difficult?

EVALUATION OF THE SOVIET ALGORITHM FOR LINEAR PROGRAMMING. We evaluate the new algorithm mentioned at the beginning of this article. The following summarizes the views of most researchers .

1. The algorithm is of theoretical interest since it proves that linear programming can be performed in polynomial time.

2. The algorithm is not currently competitive with the simplex method for linear programming problems of the size which occur in practice. There is a possibility that an improved algorithm would be competitive.

3. Statements in the New York Times and the magazine Science relating linear programming to the traveling salesman problem are incorrect.

4. The new algorithm uses an iterative approach that might be useful in solving combinatorial problems which are harder than linear programming.

5. The algorithm has been erroneously ascribed solely to Khachian. Suitable names are "Soviet" or "ellipsoidal" algorithm.

CONCLUSIONS. Computational complexity deals with thefundamental issues of determining the intrinsic difficulty of mathematically posed problems. Through the study of complexity it has been established that certain problems are intrinsically hard. On the other hand, for some problem areas new algorithms have been introduced which are far superior to any previously known. Problems occuring in a rich diversity of disciplines will be subjected to complexity analysis.

REFERENCES

1968, 1969, 1973. Knuth, D. The Art of Computer Programming, Vol. I, II, and III. Reading, Mass.: Addison-Wesley Publ.

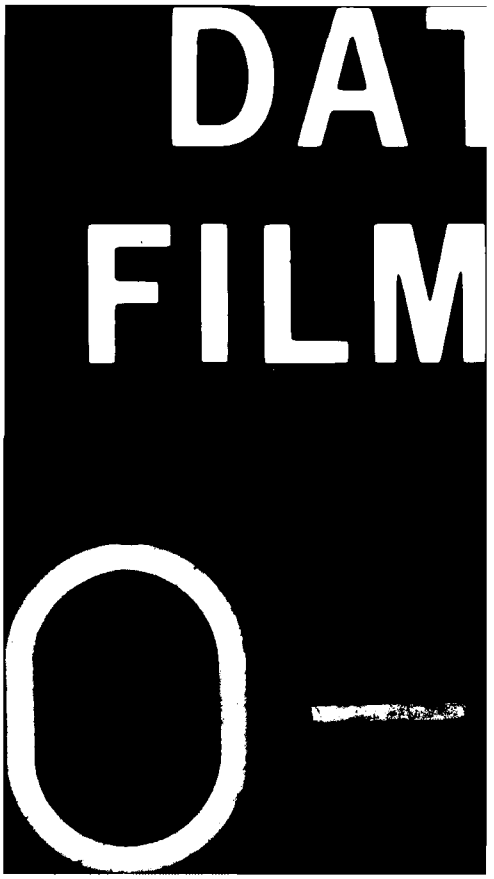
1974. Aho, A.V., J.E. Hopcroft, and J.D. Ullman. The Design and Analysis of Computer Algorithms. Reading, Mass.: Addison-Wesley Publishing Co.

1975. Borodin, A. and I. Munro. The Computational Complexity of Algebraic and Numeric Problems. New York, N.Y.: American Elsevier.

1979. Garey, M.R. and D.S. Johnson. Computers and Intractability. San Francisco, Calif.: W.H. Freeman.

1980. Arden, B. (editor). Chapter on Theory of Computation in What Can Be Automated? The Computer Science and Engineering Research Study. Cambridge, Mass: MIT Press.

1980. Traub, J.F. and H. Wozniakowski. A General Theory of Optimal Algorithms. New York, N.Y.: Academic Press.


DAT
FILM