stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE (DEPARTMENT OF OPERATIONS RESEARCH)

BW 159/82

APRIL

J.K. LENSTRA, A.H.G. RINNOOY KAN, P. VAN EMDE BOAS

AN APPRAISAL OF COMPUTATIONAL COMPLEXITY FOR OPERATIONS RESEARCHERS

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics Subject Classification: 68C25, 90Bxx, 90Cxx

AN APPRAISAL OF COMPUTATIONAL COMPLEXITY FOR OPERATIONS RESEARCHERS

J.K. LENSTRA
Mathematisch Centrum, Amsterdam

A.H.G. RINNOOY KAN
Erasmus University, Rotterdam

P. VAN EMDE BOAS
University of Amsterdam

ABSTRACT

We review recent developments in the theory and practice of computational complexity, in order to highlight some of the basic concepts and ideas that have come out of this area. The discussion centers around the progress on twelve important open problems listed in 1979 by M.R. Garey and D.S. Johnson, the introduction of probabilistic elements in the analysis and design of algorithms, the problem classes around P and NP, and the $P \neq NP$ conjecture.

KEY WORDS & PHRASES: *computational complexity, polynomial algorithm, NP-hardness, probabilistic analysis, probabilistic algorithm.*
NOTE: This review has been written for publication in the *European Journal of Operational Research* at the invitation of the Editors.

1. INTRODUCTION

Computational complexity theory as a practical tool for the investigation of combinatorial optimization problems came into being about ten years ago, with the publication of two classical papers by S.A. Cook [11] and R.M. Karp [28]. They together laid the foundation for a technique that can be used to establish the *NP-completeness* of certain combinatorial problems. Such problems are unlikely to be *well solvable*, i.e., solvable by an amount of computational effort which is bounded by a polynomial function of problem size.

The rest is history. There is hardly any need here to recall the wealth of results that were obtained by successful applications of this technique. It has led to a surprisingly sharp borderline between *easy* problems (which are solvable in polynomial time) and *hard* problems (of which some restricted version is NP-complete), where minor changes in some problem parameter may transfer a problem from one class to the other. It has provided increasingly convincing evidence that the theoretical labels *easy* and *hard* are justified by computational practice, thereby supporting our intuitions about the inherent intractability of many notorious combinatorial optimization problems. And finally, it has spawned an impressive amount of research, ranging from refinements and extensions of the original complexity measures to theoretical studies of the performance of approximation algorithms.

We certainly do not intend to give a complete survey of the state of the art in this area. That task by itself would be virtually impossible, in view of the thousands of results that would have to be referenced as well as the ongoing stream of new publications. To the extent that it can be done at all, it has been carried out in an admirable fashion by M.R. Garey and D.S. Johnson in their textbook [21] - and they themselves prefer a quarterly update column [26] to an all-encompassing second edition.

Rather, after a brief review of the basic concepts in Section 2, we hope to point out some of the most important results and questions that have emerged from ten years of research. In the course of doing so, we shall concentrate of those issues that are relevant to an operations research audience. Although computational complexity theory has contributed significantly to bring out the joint interests of (practical) operations researchers and (theoretical) computer scientists in algorithmic problem complexity, there

are certain ramifications of the theory that, currently at least, are only of interest to the latter group.

The material has been grouped around four themes. In Section 3, we demonstrate the rapidity of the advance in this research area by focusing on the twelve *open problems* that were left as a challenge in Garey and Johnson's book; this will serve to illustrate many recent ideas and techniques. In Section 4, we discuss the use of *probability theory* in the analysis and design of algorithms. In Section 5, we briefly examine the *problem classes* around P and NP. Finally, in Section 6, we return to the fundamental $P \neq NP$ *conjecture*, that has successfully withstood ten years of attack, and indicate how this tenacity might be accounted for.

2. COMPUTATIONAL COMPLEXITY THEORY

Below, we briefly summarize the basic concepts of computational complexity theory. The reader is referred to [28,29,21,46] for details.

The theory deals primarily with *decision problems*, which require a *yes/no* answer. Such a problem *type* is usually formulated as follows: given a problem *instance* (specified in terms of sets, graphs, matrices, vectors, numbers etc.), does there exist an associated *structure* which satisfies a certain property? A problem instance is said to be *feasible* if it leads to a *yes* answer, and a problem type is formally defined as the set of all its feasible instances. The *size* of an instance is the number of bits needed to encode the data, and the *running time* of an algorithm for its solution is the number of elementary operations required.

A problem type is in the class P if there exists an algorithm that, for any instance, determines in polynomial time whether the answer is *yes* or *no*, i.e., its running time is bounded by a polynomial function of problem size. A problem type is in the class NP if there exists an algorithm that, for any instance, tests the validity of a given structure in polynomial time, thereby verifying the *yes* answer.

For example, consider the problem of scheduling n precedence constrained tasks with individual release dates, processing times and deadlines within a given time limit. In case there is an unlimited number of processors, a

straightforward critical path calculation will determine feasibility or infeasibility in $O(n^2)$ time and hence the problem is in P. In case there is only a single processor available, no such simple algorithm is known; but any given schedule can be tested for feasibility in $O(n^2)$ time and hence the problem is in NP.

It is clear that P is a subset of NP. The members of P are said to be *well solvable* or *easy*. Among the members of NP are many notorious combinatorial problems which are not known to belong to P, and it is commonly conjectured that P is a *proper* subset of NP.

To obtain further insight into the structure of NP, we introduce the notion of *reducibility*. Problem P is *reducible* to problem P' ($P \propto P'$) if P can be considered as a special case of P', or more formally, if for any instance of P a corresponding instance of P' can be constructed in polynomial time such that solving the latter solves the former as well. A problem P' is said to be *NP-complete* if it is the most difficult problem in NP, i.e., if $P' \in NP$ and $P \propto P'$ for all $P \in NP$. If, in turn, $P' \propto P''$ for some $P'' \in NP$, then P" is NP-complete as well. Note that, if the $P \neq NP$ conjecture is true, then $P \notin P$ for each NP-complete P: the NP-complete problems are unlikely to be well solvable and the use of approximation algorithms or enumerative methods for their solution seems to be unavoidable.

In 1971, Cook [11] proved the fundamental result that the SATISFIABILITY problem is NP-complete. In 1972, Karp [28] showed that SATISFIABILITY is reducible to many other problems in NP, which are therefore NP-complete as well. Further applications of this technique created a huge tree of hundreds of NP-completeness proofs, each vertex of which can be used as the starting point for new results.

As far as *optimization* problems are concerned, one usually reformulates the problem of finding a feasible solution of, say, minimum value as the problem of deciding whether there exists a feasible solution with value at most equal to a given threshold. If this decision problem is NP-complete, then the optimization problem is said to be *NP-hard* in the sense that it is at least as difficult as any problem in NP.

3. THE OPEN PROBLEMS

The textbook by Garey and Johnson [21] has rapidly become the main reference for researchers in computational complexity theory. It presents a detailed treatment of the theory and the proof techniques. But its most useful feature is probably the list of 320 main NP-completeness and NP-hardness results (and many more side results), grouped according to twelve areas of application. A thirteenth group contains twelve problems that were open in 1979, when the book appeared. These problems and their current status are listed in Table 1. Six of them have been resolved in the mean time: three have turned out to be well solvable and three have been proved NP-complete.

We will discuss the substantial progress which has been made on the seven problems that are marked by an asterisk in Table 1; they are the most relevant ones in the present context. In addition, we will mention some interesting developments with respect to the NP-complete INTEGER PROGRAMMING problem. Our presentation is partly based on Johnson's first update column [26], and the reader is urged to consult it for further details as well as for information on the five problems that are not considered here.

problem	status
GRAPH ISOMORPHISM*	open
SUBGRAPH HOMEOMORPHISM (FOR A FIXED GRAPH H)	open
GRAPH GENUS	open
CHORDAL GRAPH COMPLETION	NP-complete [62]
CHROMATIC INDEX*	NP-complete [25]
SPANNING TREE PARITY*	well solvable [48,49]
PARTIAL ORDER DIMENSION	NP-complete [41,63]
PRECEDENCE CONSTRAINED THREE-PROCESSOR SCHEDULING*	open
LINEAR PROGRAMMING*	well solvable [33]
TOTAL UNIMODULARITY*	well solvable [58]
COMPOSITE NUMBER*	open
MINIMUM-LENGTH TRIANGULATION	open

Table 1. The open problems and their current status.

GRAPH ISOMORPHISM: Given two graphs G = (V,E) and G' = (V',E'), is there a one-to-one onto function f: V → V' such that {v,w} ∈ E if and only if {f(v),f(w)} ∈ E'?

Status: open.

This remains one of the most vexing open problems. The question can be answered in polynomial time for a large number of special cases, the most notable of which is the case in which the maximum vertex degree is bounded by a constant [50]. This result relies heavily on ideas from the theory of *permutation groups*, thus providing an excellent demonstration of the growing influence of pure mathematics on algorithmic combinatorics; see COMPOSITE NUMBER and INTEGER PROGRAMMING below for other examples. The techniques from [50] have been used to obtain an algorithm for the general case that requires $O(\exp(|V|^{2/3}))$ time [2,64] - still exponential, but much better than the crude $O(|V|!)$ bound.

We note that knowledge about well-solvable special cases imposes constraints on the construction of an NP-completeness proof for the general case. We conjecture that the present constraints are too strong and that GRAPH ISOMORPHISM is not NP-complete.

CHROMATIC INDEX: Given a graph G = (V,E) and an integer k, can E be partitioned into at most k disjoint sets (color classes) such that no two edges in the same set have a common endpoint?

Status: NP-complete.

The reader should distinguish between this *edge* coloring problem and the more familiar *vertex* coloring problem. The *chromatic number* of a graph is the minimum number of colors to be assigned to its vertices such that no two adjacent vertices get the same color; CHROMATIC NUMBER is one of the war-horses in the NP-complete repertoire [28,21]. The *chromatic index* of a graph is the minimum value of k for which the above question has a positive answer. We know, by Vizing's Theorem [8], that it is equal to either m or m+1, where m is the maximum vertex degree in G; CHROMATIC INDEX is thus the problem of choosing between these two values. This decision problem has been proved NP-complete by I. Holyer [25], even for m = 3.

SPANNING TREE PARITY: Given a graph G = (V,E) and a partition of E into disjoint pairs of edges, is there a spanning tree of G such that, for each pair, either both edges are in the tree or neither of them is?

Status: well solvable.

This problem has turned out to be solvable in polynomial time by a very complicated algorithm due to L. Lovász [48,49]. His method in fact solves the more general *matroid parity* problem for the case that the matroid is *representable* (i.e., its independent sets correspond to the independent sets in a linear space) and such a representation is given. The latter condition can even be dropped [39]. In the case that no special structure of the matroid is known and one needs to call a subroutine (or *oracle*) to determine whether any given set is independent or not, an exponential number of calls may be required, and hence the general matroid parity problem is *not* well solvable [49].

PRECEDENCE CONSTRAINED THREE-PROCESSOR SCHEDULING: Given n unit-time jobs, arbitrary precedence constraints between them and a deadline d, can the jobs be scheduled on three identical parallel machines such that the precedence constraints are respected and each job is processed in the interval [0,d]?

Status: open.

An inordinate amount of research effort has been spent on a more general version of this problem, where the number of machines is an input variable rather than a given constant. NP-completeness has been established for many exotic types of precedence constraints, and many other equally exotic cases can be solved in polynomial time; see [26,40] for details.

In spite of all this, the three-processor problem has stayed out of reach. It is one of the foremost open problems in a class of several thousands of *scheduling* problems, which is surveyed in [40] and catalogued in [37,38]. These investigations have led to precise insights into the location of the borderline between easy and hard scheduling problems and, as a result, into the problem features that account for border hopping.

Another area that calls for a similarly detailed complexity analysis is *location* theory, where one finds a proliferation of polynomial algorithms and NP-hardness results. In a third important application area, that of *routing* and *distribution* problems, almost all problems are NP-hard [47].

The recent developments in these three areas provide good examples of the interaction between complexity theory and the design and analysis of heuristics. NP-completeness theory offers some immediate insights (such as the incompatibility of "strong" NP-completeness and the existence of a fully polynomial approximation scheme [21]) and some less immediate ones (such as the worst case performance of polynomial heuristics as implied by certain problem reductions). The rise of complexity theory has coincided very fortunately with the emergence of analytical (rather than empirical) techniques for studying the quality of fast heuristics for hard problems.

LINEAR PROGRAMMING: Given an integer m×n-matrix A, an integer m-vector b, an integer n-vector c and an integer d, is there a *rational* n-vector x such that $Ax \leq b$ and $cx \geq d$?

Status: well solvable.

The most impressive result in the mathematics of operations research over the past few years is the development of a polynomial algorithm for LINEAR PROGRAMMING, the *ellipsoid method* due to L.G. Khachiyan [33] (see [20] for an alternative presentation and [10] for a survey of recent research into the ellipsoid method). It was well known that the problem is easy to solve in practice by the *simplex method*, and it was equally well known that the simplex method can exhibit exponential running time in the worst case [34]. The ellipsoid method confirms our intuition that LINEAR PROGRAMMING should admit of a polynomial algorithm - but it does so in a disconcerting manner. More often than not, the method seems to require its worst case number of iterations; this number is proportional to the number of bits needed to store all coefficients of A, b, c and d and hence is very large indeed. Here is a theoretically *polynomial* algorithm that is practically no *good* at all, thereby undermining the justification of our basic concepts.

The major role of the ellipsoid method, however, seems to be to establish that certain problems belong to P and to clear the way for really efficient algorithms. As such, it has become an important tool in the resolution of many combinatorial optimization problems [22,32]. It is fair to say that the episode has led to a less dogmatic attitude towards polynomial solvability as well as to more appreciation for the contributions from nonlinear and non-differentiable optimization to combinatorial optimization.

TOTAL UNIMODULARITY: Given an m×n-matrix A with entries from the set {-1,0,1}, is A *not* totally unimodular, i.e., is there a square submatrix A' of A such that $\det(A') \notin \{-1,0,1\}$?

Status: well solvable.

Note that, in the above formulation, the problem is obviously in *NP*. Its membership of *P* follows from a theorem due to P.D. Seymour [58]. The interest in this problem stems from the well-known fact that, if A is totally unimodular, then LINEAR PROGRAMMING has an *integer* solution x, if there is any solution at all (or, in other words, the integrality restriction on x in INTEGER PROGRAMMING is superfluous). Further work has resulted in polynomial algorithms for linear programs on totally unimodular matrices [17,52,61], which are more efficient than the ellipsoid method.

COMPOSITE NUMBER: Given a positive integer n, are there positive integers $p,q > 1$ such that $n = p \cdot q$?

Status: open.

This problem is unlikely to be NP-complete. Highly sophisticated ideas from *number theory* have led to a $\Theta(\ell^{c \log\log \ell})$ algorithm (where $\ell = \log n$ is the problem size) [1,44], which is not only very close to polynomial but also very fast in practice.

If a number n passes the test, we know that n is composite but we do not get its prime factors. The *factorization* problem seems to be much harder than the basic decision problem, and a similarly efficient algorithm for its solution would immediately endanger the safety of many cryptographic codes.

INTEGER PROGRAMMING: Given an integer m×n-matrix A, an integer m-vector b, an integer n-vector c and an integer d, is there an *integer* n-vector x such that $Ax \leq b$ and $cx \geq d$?

Status: NP-complete.

This is one of the most widely studied problem types in combinatorial optimization. It is extremely useful in the formulation of many practical operations research problems, and several commercial computer codes are available for its solution.

The problem is also extremely difficult: many highly restricted special cases are NP-complete. However, the case in which the number n of variables

is *fixed* has turned out to be solvable in polynomial time by an algorithm due to H.W. Lenstra [43]. His method is based on ideas from the *geometry of numbers*. As a corollary, the case in which the number m of constraints is fixed is solvable in polynomial time as well. If the condition $x \geq 0$ (representing n constraints) is added to the problem statement, then the case of fixed m can be solved in "pseudopolynomial" time [54].

A recent improvement in the method from [43] yielded, as a surprising byproduct, a polynomial algorithm for the problem of *factoring univariate polynomials with rational coefficients* [42].

4. PROBABILISTIC ASPECTS

NP-Completeness theory is essentially concerned with the *worst case* analysis of problems and algorithms. Such an analysis has to account for the isolated time consuming problem instance, and hence the results can be overly pessimistic and generally give a very misleading picture of the *average case*. This point is strongly supported by an abundance of empirical evidence. Thus the ultimate analytical explanation of why algorithms behave as they do must be of a *probabilistic* nature.

A probabilistic analysis requires first of all the specification of a probability distribution over the set of all problem instances. For example, a *random graph* can be obtained by specifying a fixed probability that any vertex pair constitutes an edge or, alternatively, by distributing a fixed number of edges uniformly over all vertex pairs. Both notions have been well studied, especially in representing complex counting arguments [18]. For many other combinatorial structures, the choice of a reasonable probability model is far less obvious.

Moreover, the technical difficulties encountered in a probabilistic analysis are formidable. The main reasons for this are the very special structure of problem instances and solutions, as well as the interdependence between the various steps of an algorithm. What happens at a node of a search tree, for example, depends highly on what happened at its predecessors, and no real way has been found around the resulting mathematical obstacles [45].

Nevertheless, progress has been made on various fronts. One of these is

probabilistic running time analysis, an approach that is now more or less standard for the basic algorithms in computer science such as sorting, searching and selection. It has been shown that, in the second probability model for random graphs mentioned above, GRAPH ISOMORPHISM (see Section 3) is solvable by an algorithm that runs in linear expected time [3]. The great challenge here remains the explanation of the success of the simplex method for LINEAR PROGRAMMING (see Section 3): polynomial expected behavior has been established, but only under assumptions concerning the method rather than the underlying problem [12,53], which is not satisfactory at all. A similar challenge is to give rigid proofs of the polynomial expected running time of some tree search methods, in order to confirm informal analyses (such as for the traveling salesman algorithm in [7]) or empirical evidence (such as for the knapsack algorithm in [5]). So far, all precise results in this direction have been negative, in the sense that within a certain probability model for some NP-hard problem any tree search method of a certain type can be proved to require almost always superpolynomial time [30].

Secondly, there is the area of *probabilistic error analysis*, where the error refers to the difference between an approximate solution value and the optimum. Again, the empirical behavior of heuristics suggests that the worst case is seldom met in practice, but analytical verification remains very difficult. Most research of this type is actually based on *probabilistic value analysis*, the third and perhaps most surprising area. Many hard combinatorial optimization problems, notably those with a Euclidean structure such as routing and location problems in the plane, allow a simple probabilistic description of their optimal solution value as a function of problem parameters. The shining example here is the planar traveling salesman problem: the length of a shortest tour through n cities, uniformly distributed over a circle of area 1, is almost surely equal to $\beta\sqrt{n}$, where β is a constant that can be estimated numerically [6,23,59]; the analysis of Karp's partitioning heuristic for the problem [31,60] is based on this theorem. Similar results have been obtained for the planar K-median problem [19,24]. They find application in the analysis of hierarchical planning systems for multistage scheduling and distribution problems [14,15,51].

There is a second way in which probability theory has entered complexity

theory. This is through the notion of a *probabilistic algorithm*, i.e., an algorithm that flips a coin at certain points in order to decide how to proceed [56]. A decision problem is in the class RP if there exists a probabilistic algorithm that runs in polynomial time and, if the answer is *yes*, produces that answer with probability greater than $\frac{1}{2}$. (This probability can, in fact, be brought up to any value smaller than 1.) The most prominent member of RP is COMPOSITE NUMBER (see Section 3) [57]; the rejection of a number by Rabin's algorithm yields virtual certainty that it is a prime.

It is clear that P is a subset of RP, and the two classes might well be the same. If a problem belongs to RP, that provides circumstantial evidence against its NP-completeness. Moreover, the polynomial probabilistic algorithm for its solution might be quite practical.

5. AROUND P AND NP

Many problem classes have arisen around P and NP. The class RP defined above is one of them. Some of the others will be briefly discussed here.

Of some relevance to operations researchers is the class co-NP. A problem is in co-NP if its *complement* is in NP. For example, the HAMILTONIAN CIRCUIT problem (given a graph, does it contain a Hamiltonian circuit?) belongs to NP, and therefore its complement NO HAMILTONIAN CIRCUIT (given a graph, does it contain *no* Hamiltonian circuit?) belongs to co-NP. The latter problem is not known to be in NP: there is no obvious structure corresponding to the non-existence of a Hamiltonian circuit, let alone a polynomial algorithm for its verification. It is conjectured that the only problems in NP as well as in co-NP are precisely those in P. Hence, if a problem and its complement are both in NP, this provides a strong indication for the existence of a polynomial algorithm. Because of duality theory, LINEAR PROGRAMMING is such a problem, and the indication has been correct. COMPOSITE NUMBER is another one [55]; as we have seen before, the problem might very well turn out to belong to P.

Other examples are the class DLOGSPACE of problems that (in addition to the space used for the input) require no more than logarithmic space for their solution, and the class PSPACE of problems that require polynomial space. It

is not hard to see that DLOGSPACE $\subseteq P \subseteq NP \subseteq$ PSPACE, and it is conjectured that each of these inclusions is a proper one. LINEAR PROGRAMMING has been shown to be *log-space complete for* P in the sense that all other problems in P are transformable to it in logarithmic space [16]; hence, DLOGSPACE = P if LINEAR PROGRAMMING would belong to DLOGSPACE, but that would represent a dramatic improvement over the ellipsoid method.

Beyond the problems in PSPACE, there are the *intractable* problems, for which superpolynomial worst case time requirement is not merely conjectured but has been proved.

For a detailed discussion of these and still other problem classes, we refer to [21,Ch.7]. The basic distinction between *solvability in polynomial time* and *NP-hardness* provides sufficient terminology for everyday practice in combinatorial optimization.

6. P VERSUS NP

A puzzling aspect of the state of the art in computational complexity is the very fact that the concentrated effort of so many researchers has failed to settle the $P \neq NP$ conjecture. The equality $P = NP$ is after all very unlikely to hold in the real world of computation. If it would be true, then our impression that the empirical notions of *easy* and *hard* had found their theoretical counterpart would have been a sad mistake.

Why should this problem be as hard as it seems to be? The basic notion of complexity, leading to the simple distinction between solvability in polynomial time and NP-hardness, appears to be too complex to be described and understood by the use of formal mathematics. Several lines of attack to the problem are reviewed below. Each of them has yielded a lot of fruitful insights, but has failed to settle the conjecture, thereby confirming that its true implications have not yet been grasped.

Originally, the merit of Cook's result that SATISFIABILITY is NP-complete [11] seemed to be that it reduced the effort needed to settle the conjecture. To verify that $P \neq NP$, one only had to prove the nonexistence of a polynomial algorithm for SATISFIABILITY - and if someone would unexpectedly come up with

such an algorithm, then $P = NP$ would follow. The thousands of NP-completeness results obtained in the past decade should have improved the chance of one of the two events to occur. However, the situation has not changed at all, since it has been shown that all *known* NP-complete problems are essentially the same: they are *polynomially isomorphic* in the sense that they are reducible to one another by means of one-to-one surjective transformations [9].

It is not known whether this statement is true for *all* NP-complete problems. That conjecture implies that $P \neq NP$, by the following trivial argument: if $P = NP$, then all problems in P are NP-complete and hence isomorphic; but P contains problems with a finite as well as with an infinite number of feasible instances, which cannot be isomorphic.

A standard tool that has been used to prove intractability and even undecidability of problems is the *diagonalization* construction. Let $(P_1, P_2, \ldots)$ be a list of all problems in P, and let $(I_1, I_2, \ldots)$ be a list of all their instances in some standard encoding and enumeration scheme. A problem Q which is guaranteed to be *not* in P is then easily defined by $Q = \{I_i \mid I_i \notin P_i,\ i = 1,2,\ldots\}$. So all that remains to be done in order to prove $P \neq NP$ is to organize the lists in such a way that the resulting problem Q belongs to NP. No one has been able to do this so far.

There are indications that such an approach is doomed to fail. They are based on the observation that the above argument is hardly related to the real world of computation and carries through in other worlds. One way to create such a world is *relativization* of our concepts with respect to a given problem P, by the use of an *oracle* machine. An oracle machine has a special instruction to test in unit time whether any given instance is feasible for problem P. For a model of computation extended with this feature one can define classes P^P and NP^P, analogous to P and NP. For example, if P = $\emptyset$, then $P^P = P$ and $NP^P = NP$; if P = SATISFIABILITY, then $P^P \supset NP$ (since SATISFIABILITY is NP-complete).

If one could prove $P \neq NP$ by diagonalization, then this might suggest that the proof would continue to work after relativization, so that $P^P \neq NP^P$ for arbitrary P. It has been shown, however, that $P^P = NP^P$ for some problems P whereas $P^Q \neq NP^Q$ for other problems Q [4]. This does not imply that $P \neq NP$ cannot be proved by diagonalization, but it does imply that such a proof

must be based on a model of computation for the real world, so that relativization to an arbitrary problem is impossible. We refer to [35] for further developments in this direction.

Another way to proceed is to start from the assumption that $P = NP$ (or $P \neq NP$) and to see whether it leads to a contradiction. A typical result along this line is as follows: under the hypothesis that $P \neq NP$, there are problems in NP that are neither in P nor NP-complete; in fact, there would be an infinite hierarchy of distinct equivalence classes between P and the class of NP-complete problems [36]. This approach suggests some really weird problems in NP, but does not answer the $P \neq NP$ conjecture.

This type of investigation is comparable to the efforts undertaken in the eighteenth century by researchers in *Euclidean geometry*, who tried to prove the parallel postulate. Living in the twentieth century, we know that the parallel postulate is independent of the other axioms in geometry: one can construct perfectly reasonable models of geometry in which it is valid and others in which it is not, and *non-Euclidean geometry* is now a well-established branch of mathematics.

One could imagine that the $P \neq NP$ conjecture will ultimately achieve the same status in the theory of computation over the natural numbers as the parallel postulate in geometry, i.e., that it turns out to be independent of the underlying axiom system. Combinatorial computations are founded on a formal theory known as *Peano arithmetic*, and the natural question to ask is whether both $P = NP$ and $P \neq NP$ can be consistent with this theory.

A partial result in this direction [13] is that the $P = NP$ assumption is consistent with ET, a theory which is weaker than Peano arithmetic. There are far more models of ET than of Peano arithmetic. On closer analysis [27], some of these appear to behave in a very strange way which does not reflect our intuitions about the real world at all. An extreme example is that some models of ET allow computations which have bounded running time but fail to halt! Altogether, the result from [13] does not imply that $P = NP$ is a valid assumption in the real world, which is believed to be a model of Peano arithmetic.

Research into the $P \stackrel{?}{=} NP$ question is going on. We do not expect that it will

yield an immediate answer, but rich side benefits will be obtained. In a broader sense, computational complexity will continue to fascinate a wide spectrum of researchers, stretching from pure mathematicians to those engaged in practical problem solving.

REFERENCES

1. L.M. Adleman, C. Pomerance and R.S. Rumely, On distinguishing prime numbers from composite numbers, Ann. of Math., to appear.
2. L. Babai, Moderately exponential bound for graph isomorphism, in: F. Gécseg, ed., Fundamentals of Computation Theory, Lecture Notes in Computer Science 117 (Springer, Berlin, 1981) 34-50.
3. L. Babai and L. Kucera, Graph canonization in linear average time, Proc. 20th Annual Symp. Foundations of Computer Science (1979) 39-46.
4. T. Baker, J. Gill and R. Solovay, Relativizations of the P =? NP question, SIAM J. Comput. 4 (1975) 431-442.
5. E. Balas and E. Zemel, An algorithm for large zero-one knapsack problems, Oper. Res. 28 (1980) 1130-1154.
6. J. Beardwood, J.H. Halton and J.M. Hammersley, The shortest path through many points, Proc. Cambridge Phil. Soc. 55 (1959) 299-327.
7. M. Bellmore and J.C. Malone, Pathology of traveling-salesman subtour-elimination algorithms, Oper. Res. 19 (1971) 278-307,1766.
8. C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1973).
9. L. Berman and J. Hartmanis, On isomorphisms and density of NP and other complete sets, SIAM J. Comput. 6 (1977) 305-322.
10. R.G. Bland, D. Goldfarb and M.J. Todd, The ellipsoid method: a survey, Oper. Res. 29 (1981) 1039-1091.
11. S.A. Cook, The complexity of theorem-proving procedures, Proc. 3rd Annual ACM Symp. Theory of Computing (1971) 151-158.
12. G.B. Dantzig, Expected number of steps of the simplex method for a linear program with a convexity constraint, Report SOL 80-3R, Systems Optimization Laboratory, Stanford University (1980).

13. R.A. DeMillo and R.J. Lipton, The consistency of 'P = NP' and related problems with fragments of number theory, Proc. 12th Annual ACM Symp. Theory of Computing (1980) 45-57.

14. M.A.H. Dempster, M.L. Fisher, L. Jansen, B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, Analytical evaluation of hierarchical planning systems, Oper. Res. 29 (1981) 707-716.

15. M.A.H. Dempster, M.L. Fisher, L. Jansen, B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, Analysis of heuristics for stochastic programming: results for hierarchical scheduling problems, Report BW 142, Mathematisch Centrum, Amsterdam (1981).

16. D. Dobkin, R. Lipton and S. Reiss, Linear programming is log-space hard for P, Inform. Process. Lett. 8 (1979) 96-97.

17. J. Edmonds, Seymour's theorem and good algorithms for totally unimodular matrices, in preparation.

18. P. Erdös and J. Spencer, Probabilistic Methods in Combinatorics (Academic Press, New York, 1974).

19. M.L. Fisher and D.S. Hochbaum, Probabilistic analysis of the planar K-median problem, Math. Oper. Res. 5 (1980) 27-34.

20. P. Gács and L. Lovász, Khachian's algorithm for linear programming, Math. Programming Stud. 14 (1981) 61-68.

21. M.R. Garey and D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness (Freeman, San Fransisco, 1979).

22. M. Grötschel, L. Lovász and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (1981) 169-197.

23. J.H. Halton and R. Terada, A fast algorithm for the Euclidean traveling salesman problem, optimal with probability one, SIAM J. Comput. 11 (1982) 28-46.

24. D. Hochbaum and J.M. Steele, Steinhaus' geometric location problem for random samples in the plane, Adv. in Appl. Probab. 14 (1981) 56-67.

25. I. Holyer, The NP-completeness of edge coloring, SIAM J. Comput. 10 (1981) 718-720.

26. D.S. Johnson, The NP-completeness column: an ongoing guide, J. Algorithms 4 (1981) 393-405.

27. D. Joseph and P. Young, A survey of some recent results on computational complexity in weak theories of arithmetic, in: J. Gruska and M. Chytil, eds., Mathematical Foundations of Computer Science 1981, Lecture Notes in Computer Science 118 (Springer, Berlin, 1981) 46-60.

28. R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum, New York, 1972) 85-103.

29. R.M. Karp, On the computational complexity of combinatorial problems, Networks 5 (1975) 45-68.

30. R.M. Karp, The probabilistic analysis of some combinatorial search algorithms, in: J.F. Traub, ed., Algorithms and Complexity: New Directions and Recent Results (Academic Press, New York, 1976) 1-19.

31. R.M. Karp, Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane, Math. Oper. Res. 2 (1977) 209-224.

32. R.M. Karp and C.H. Papadimitriou, On linear characterizations of combinatorial optimization problems, Proc. 21st Annual Symp. Foundations of Computer Science (1980) 1-9.

33. L.G. Khachiyan, A polynomial algorithm in linear programming, Soviet Math. Dokl. 20 (1979) 191-194.

34. V. Klee and G.J. Minty, How good is the simplex algorithm?, in: O. Shisha, ed., Inequalities III (Academic Press, New York, 1972) 159-175.

35. D. Kozen and M. Machtey, On relative diagonals, Report RC 8184, IBM Thomas J. Watson Research Center, Yorktown Heights (1980).

36. R.E. Ladner, On the structure of polynomial time reducibility, J. Assoc. Comput. Mach. 22 (1975) 155-171.

37. B.J. Lageweg, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Computer aided complexity classification of deterministic scheduling problems, Report BW 138, Mathematisch Centrum, Amsterdam (1981).

38. B.J. Lageweg, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Computer

aided complexity classification of combinatorial problems, Comm. ACM, to appear.

39. E.L. Lawler, Optimization without representation, in preparation.

40. E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Recent developments in deterministic sequencing and scheduling: a survey, in: M.A.H. Dempster, J.K. Lenstra and A.H.G. Rinnooy Kan, eds., Deterministic and Stochastic Scheduling (Reidel, Dordrecht, 1982) 35-73.

41. E.L. Lawler and O. Vornberger, The partial order dimension problem is NP-complete, unpublished manuscript.

42. A.K. Lenstra, H.W. Lenstra, Jr. and L. Lovász, Factoring polynomials with rational coefficients, Report IW 195, Mathematisch Centrum, Amsterdam (1982).

43. H.W. Lenstra, Jr., Integer programming with a fixed number of variables, Report 81-03, Department of Mathematics, University of Amsterdam (1981).

44. H.W. Lenstra, Jr., Primality testing algorithms (after Adleman, Rumely and Williams), Séminaire Bourbaki 33 (1980/1981) No. 576, Lecture Notes in Mathematics 901 (Springer, Berlin, 1981) 243-257.

45. J.K. Lenstra and A.H.G. Rinnooy Kan, On the expected performance of branch-and-bound algorithms, Oper. Res. 26 (1978) 347-349.

46. J.K. Lenstra and A.H.G. Rinnooy Kan, Computational complexity of discrete optimization problems, Ann. Discrete Math. 4 (1979) 121-140.

47. J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of vehicle routing and scheduling problems, Networks 11 (1981) 221-227.

48. L. Lovász, Matroid matching and some applications, J. Combin. Theory Ser. B 28 (1980) 208-236.

49. L. Lovász, The matroid matching problem, in: L. Lovász and V. Sós, eds., Algebraic Methods in Graph Theory, Vol. II (North-Holland, Amsterdam, 1981) 495-517.

50. E.M. Luks, Isomorphism of bounded valence can be tested in polynomial time, Proc. 21st Annual Symp. Foundations of Computer Science (1980) 42-49.

51. A. Marchetti Spaccamela, A.H.G. Rinnooy Kan and L. Stougie, Hierarchical

vehicle routing, in preparation.

52. J.F. Maurras, K. Truemper and M. Akgül, Polynomial algorithms for a class of linear programs, Math. Programming 21 (1981) 121-136.

53. A. Orden, Computational investigation and analysis of probabilistic parameters of convergence of a simplex algorithm, in: A. Prékopa, ed., Progress in Operations Research, Vol. II (North-Holland, Amsterdam, 1976) 705-715.

54. C.H. Papadimitriou, On the complexity of integer programming, J. Assoc. Comput. Mach. 28 (1981) 765-768.

55. V. Pratt, Every prime has a succinct certificate, SIAM J. Comput. 4 (1975) 214-220.

56. M.O. Rabin, Probabilistic algorithms, in: J.F. Traub, ed., Algorithms and Complexity: New Directions and Recent Results (Academic Press, New York, 1976) 21-39.

57. M.O. Rabin, Probabilistic algorithms for testing primality, J. Number Theory 12 (1980) 128-138.

58. P.D. Seymour, Decomposition of regular matroids, J. Combin. Theory Ser. B 28 (1980) 305-359.

59. J.M. Steele, Subadditive Euclidean functionals and nonlinear growth in geometric probability, Ann. Probab. 9 (1981) 365-376.

60. J.M. Steele, Complete convergence of short paths and Karp's algorithm for the TSP, Math. Oper. Res. 6 (1981) 374-378.

61. M. Yannakakis, On a class of totally unimodular matrices, Proc. 21st Annual Symp. Foundations of Computer Science (1980) 10-16.

62. M. Yannakakis, Computing the minimum fill-in is NP-complete, SIAM J. Algebraic Discrete Methods 2 (1981) 77-79.

63. M. Yannakakis, The complexity of the partial order dimension problem, unpublished manuscript.

64. V.N. Zemlyachenko, in preparation.

ONTVANGEN 1 0 MEI 1982