stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 77/77 JUNI
(DEPARTMENT OF OPERATIONS RESEARCH)

J.K. LENSTRA, A.H.G. RINNOOY KAN

COMPUTATIONAL COMPLEXITY OF DISCRETE OPTIMIZATION PROBLEMS

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, 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).

AMS (MOS) subject classification scheme (1970): 05Cxx, 68A20, 90B35, 90C10

COMPUTATIONAL COMPLEXITY OF DISCRETE OPTIMIZATION PROBLEMS

J.K. LENSTRA
Mathematisch Centrum, Amsterdam, The Netherlands

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

ABSTRACT

Recent developments in the theory of computational complexity as applied to combinatorial problems have revealed the existence of a large class of so-called *NP-complete* problems, either *all* or *none* of which are solvable in polynomial time. Since many infamous combinatorial problems have been proved to be NP-complete, the latter alternative seems far more likely. In that sense, NP-completeness of a problem justifies the use of enumerative optimization methods and of approximation algorithms. In this paper we give an informal introduction to the theory of NP-completeness and derive some fundamental results, in the hope of stimulating further use of this valuable analytical tool.

KEY WORDS & PHRASES: *good algorithm, reducibility, NP-completeness, satisfiability, clique, set partition, integer programming, hamiltonian circuit, machine scheduling*
NOTE: This report is not for review; it will be submitted for publication in a journal.

1. INTRODUCTION

After a wave of initial optimism, integer programming soon proved to be much harder than linear programming. As integer programming formulations were found for more and more discrete optimization problems, it also became obvious that such formulations yielded little computational benefit. To this day, integer programming problems of more than miniature size remain computationally intractable.

For some specially structured problems, however, highly efficient algorithms have been developed. Network flow and matching provide well-known examples of problems that are *easy* in the sense that they are solvable by a *good* algorithm - a term coined by J. Edmonds [Edmonds 1965A] to indicate an algorithm whose running time is bounded by a polynomial function of problem size. Thus, in a network on v vertices a maximum flow can be determined in $O(v^3)$ time [Dinic 1970; Karzanov 1974; Even 1976] and a maximum weight matching can be found in $O(v^3)$ time [Gabow 1976; Lawler 1976].

It is commonly conjectured that no good algorithm exists for the general integer programming problem. A similar conjecture holds with respect to many other combinatorial problems that are *notorious for their computational intractability* [Johnson 1973], such as graph colouring, set covering, travelling salesman and job-shop scheduling problems. Typically, all optimization methods that have been proposed so far for these problems are of an enumerative nature. They involve some type of backtrack search in a tree whose depth is bounded by a polynomial function of problem size. In the worst case, those algorithms require superpolynomial (*e.g.*, exponential) time.

We shall denote the class of all problems solvable in polynomial time by P and the class of all problems solvable by polynomial-depth backtrack search by NP. It is obvious that $P \subset NP$.

The battle against hard combinatorial problems dragged on until S. Cook [Cook 1971] and R.M. Karp [Karp 1972] showed the way to *peace with honour* [Fisher 1976]. They exhibited the existence within NP of a large class of so-called NP-complete problems [Knuth 1974] that are equivalent in the following sense:

- none of them is known to belong to P;
- if one of them belongs to P, then all problems in NP belong to P, which would imply that $P = NP$.

NP-completeness of a problem is generally accepted as strong evidence against the existence of a good algorithm and consequently as a justification for the use of enumerative optimization methods such as branch-and-bound or of approximation algorithms. By way of examples, all the hard problems mentioned above are NP-complete.

NP-completeness has proved to be an extremely fruitful research area. The computational complexity of many types of combinatorial problems has been analyzed in detail. Under the assumption that $P \neq NP$, this analysis often reveals the existence of a sharp borderline between P and the class of NP-complete problems that is expressible in terms of natural problem parameters. Moreover, establishing NP-completeness of a problem provides important information on the quality of the algorithm that one can hope to find, which makes it easier to accept the computational burden of enumerative methods or to face the inevitability of a heuristic approach.

In this paper we shall not attempt to present an exhaustive survey of all NP-completeness results (see [Karp 1972; Karp 1975; Garey & Johnson 1978]). Instead, we shall examine some typical NP-complete problems, demonstrate some typical proof techniques and discuss some typical open problems (*cf.* [Aho *et al.* 1974; Savage 1976; Reingold *et al.* 1977]). We hope that as a result the reader will be stimulated to consider the computational complexity of his or her favourite combinatorial problem and to draw the algorithmic implications.

2. CONCEPTS OF COMPLEXITY THEORY

A formal theory of NP-completeness would require the introduction of *Turing machines* as theoretical computing devices [Aho *et al.* 1974]. Turing machines can be designed to recognize *languages*; the input to the machine consists of a *string*, which is accepted if and only if it belongs to the language. P is then defined as the class of languages recognizable in polynomial time by a *deterministic* Turing machine, a suitable model for an ordinary computer. NP is similarly defined as the class of languages recognizable in polynomial time by a *nondeterministic* Turing machine, which can be thought of as a deterministic one that can duplicate its current state in zero time when-

ever convenient. For our purposes, however, we may identify languages and strings with problem types and problem instances respectively, and retain the informal definitions of $\mathcal{P}$ and $\mathcal{NP}$ given in the introduction.

Problem P' is said to be *reducible* to problem P (notation: $P' \propto P$) if for any instance of P' an instance of P can be constructed in polynomial time such that solving the instance of P will solve the instance of P' as well. Informally, the reducibility of P' to P implies that P' can be considered as a special case of P, so that P is as least as hard as P'.

P is called *NP-hard* if $P' \propto P$ for every $P' \in \mathcal{NP}$. In that case, P is at least as hard as any problem in $\mathcal{NP}$. P is called *NP-complete* if P is NP-hard and $P \in \mathcal{NP}$. Thus, the NP-complete problems are the most difficult problems in $\mathcal{NP}$.

A good algorithm for an NP-complete problem P could be used to solve all problems in $\mathcal{NP}$ in polynomial time, since for any instance of such a problem the construction of the corresponding instance of P and its solution can both be effected in polynomial time. Note the following two important observations.

- It is very unlikely that $\mathcal{P} = \mathcal{NP}$, since $\mathcal{NP}$ contains many notorious combinatorial problems, for which in spite of a considerable research effort no good algorithms have been found so far.
- It is very unlikely that $P \in \mathcal{P}$ for any NP-complete P, since this would imply that $\mathcal{P} = \mathcal{NP}$ by the earlier argument.

The first NP-completeness result is due to Cook [Cook 1971]. He designed a "master reduction" to prove that every problem in $\mathcal{NP}$ is reducible to the SATISFIABILITY problem of determining whether a boolean expression in conjunctive normal form can assume the value *true*. Given this result, one can establish NP-completeness of some $P \in \mathcal{NP}$ by specifying a reduction $P' \propto P$ with P' already known to be NP-complete: for every $P'' \in \mathcal{NP}$, $P'' \propto P'$ and $P' \propto P$ then imply that $P'' \propto P$ as well.

In the following section we shall present several such proofs. In order to ensure compatibility between *recognition* problems which require a yes/no answer and *optimization* problems, we shall reformulate a minimization (maximization) problem by asking for the existence of a feasible solution with value at most (at least) equal to a given *threshold*. NP-completeness will always be proved with respect to recognition problems. The correspon-

ding optimization problem can often not formally be shown to belong to NP, but it might be called NP-hard in the sense that the existence of a good algorithm for its solution would imply that $P = NP$.

3. NP-COMPLETENESS RESULTS

In this section we shall establish some basic NP-completeness results according to the scheme given in Figure 1, and we shall mention similar results for related problems. Our proofs will be sketchy; for instance, it will be left to the reader to verify the membership of NP for the problems considered and the polynomial-boundedness of the reductions presented.

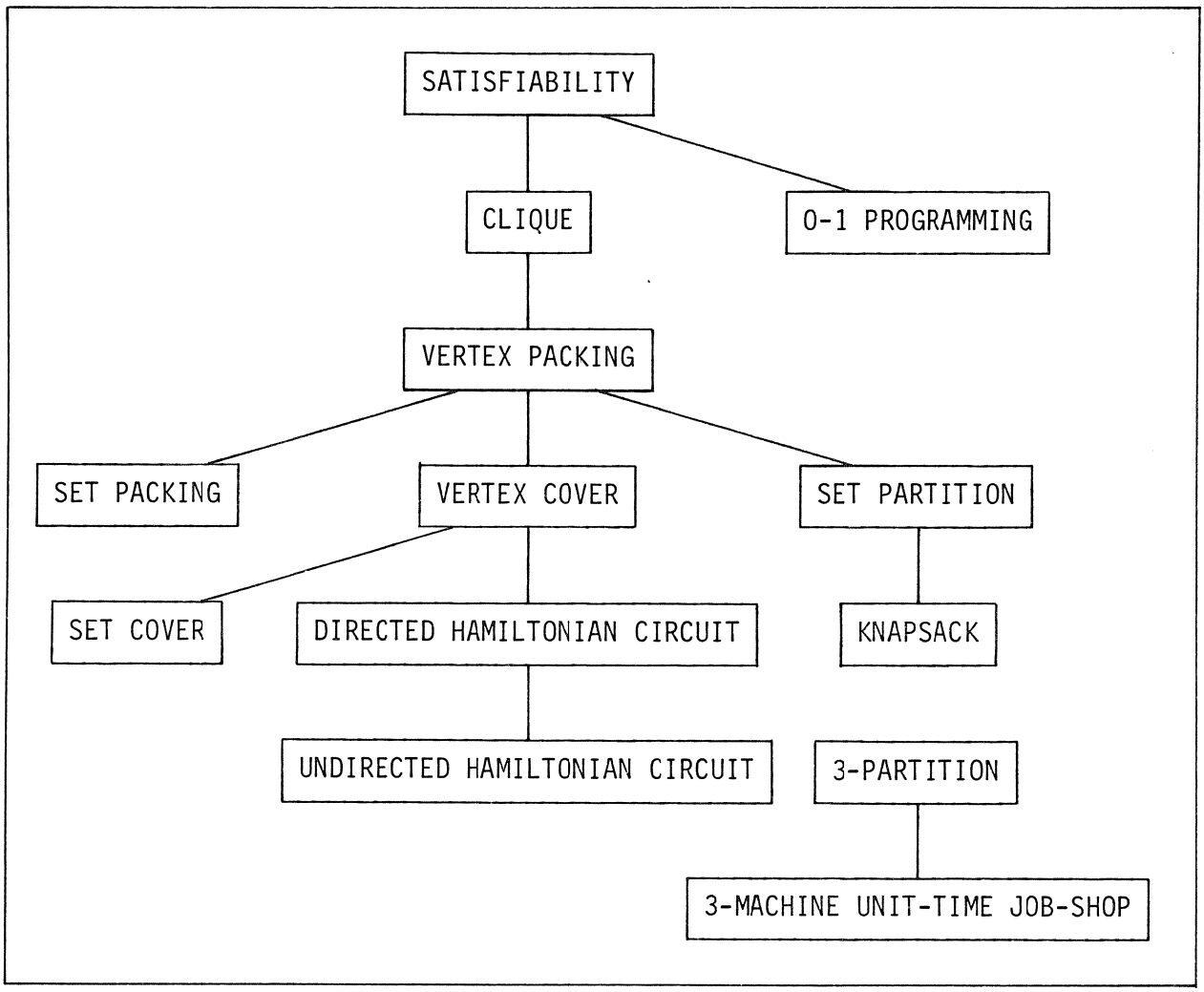

Figure 1 Scheme of reductions.

3.1. SATISFIABILITY

SATISFIABILITY: Given a *conjunctive normal form expression, i.e.* a conjunction of *clauses* $C_1,\ldots,C_s$, each of which is a disjunction of *literals* $x_1,\bar{x}_1,\ldots,x_t,\bar{x}_t$ where $x_1,\ldots,x_t$ are boolean variables and $\bar{x}_1,\ldots,\bar{x}_t$ denote their complements, is there a truth assignment to the variables such that the expression assumes the value *true*?

NP-completeness

It has already been mentioned that SATISFIABILITY was the first problem shown to be NP-complete. The proof of this key result is quite technical and beyond the scope of this paper; we refer to [Cook 1971; Aho *et al.* 1974].

We shall take the instance of SATISFIABILITY given in Figure 2 as an example to illustrate subsequent reductions. Clearly, the expression is satisfied if $x_1 = x_2 = x_3 = true$.

$$(\underline{\underline{x_1}}) \wedge (\bar{x}_1 \vee \underline{\underline{x_2}} \vee \bar{x}_3) \wedge (\underline{\underline{x_3}})$$

Figure 2 Instance of SATISFIABILITY for the example.

Related results

Even the 3-SATISFIABILITY problem, *i.e.* SATISFIABILITY with at most three literals per clause, is NP-complete [Cook 1971]. The 2-SATISFIABILITY problem, however, belongs to P. Often, the borderline between easy and hard problems is crossed when a problem parameter increases from two to three. This phenomenon will be encountered on various occasions below, and is held by some to explain the division of mankind in two and not three sexes.

3.2. CLIQUE, VERTEX PACKING & VERTEX COVER

CLIQUE: Given an undirected graph G = (V,E) and an integer k, does G contain a set of at least k pairwise adjacent vertices?

VERTEX PACKING (INDEPENDENT SET): Given an undirected graph G' = (V',E') and an integer k', does G' contain a set of at least k' pairwise nonadjacent vertices?

VERTEX COVER: Given an undirected graph G = (V,E) and an integer k, does G contain a set of at most k vertices such that every edge is incident with at least one of them?

NP-completeness

SATISFIABILITY $\propto$ CLIQUE:

$V = \{(x,i) \mid x \text{ is a literal in clause } C_i\}$;

$E = \{\{(x,i),(y,j)\} \mid x \neq \bar{y},\ i \neq j\}$;

$k = s$.

Cf. Figure 3. We have created a vertex for each occurence of a literal in a clause and an edge for each pair of literals that can be assigned the value *true* independently of each other. A clique of size k corresponds to s literals (one in each clause) that satisfy the expression and *vice versa* [Cook 1971]. The NP-completeness of CLIQUE now follows from (i) its membership of NP, (ii) the polynomial-boundedness of the reduction, and (iii) the NP-completeness of SATISFIABILITY.

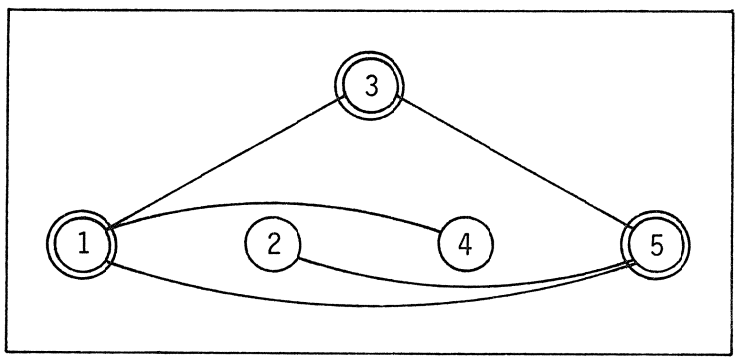

Figure 3 Instance of CLIQUE for the example.

CLIQUE $\propto$ VERTEX PACKING:

$V' = V$;

$E' = \{\{i,j\} \mid i \neq j,\ \{i,j\} \notin E\}$;

$k' = k$.

Cf. Figure 4. A set of vertices is independent in G' if and only if it is a

clique in the complementary graph G. This relation between the two problems belongs to folklore.

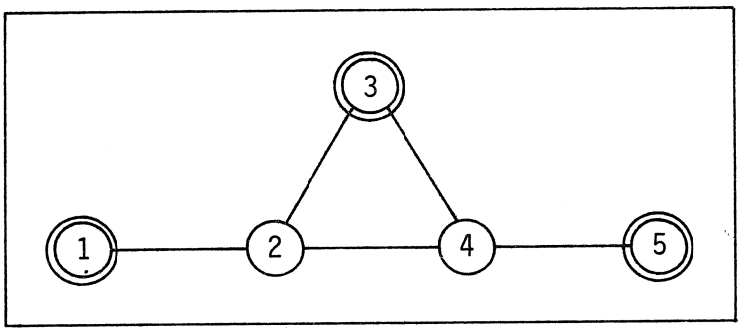

Figure 4 Instance of VERTEX PACKING for the example.

VERTEX PACKING $\propto$ VERTEX COVER:

$V = V'$;

$E = E'$;

$k = |V'|-k'$.

Cf. Figure 5. It is easily seen that a set of vertices covers all edges if and only if its complement is independent [Karp 1972].

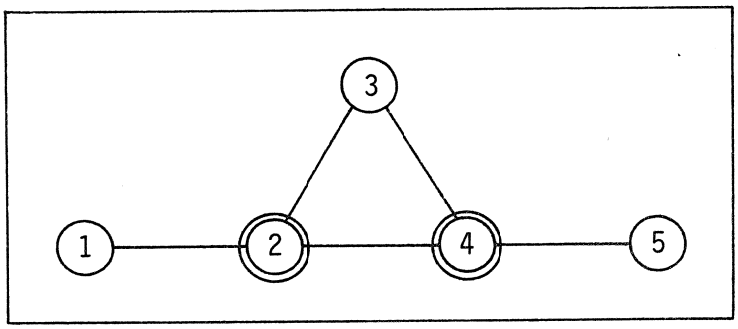

Figure 5 Instance of VERTEX COVER for the example.

Related results

Given the above results, it is not surprising (though less easy to prove) that the problems of determining whether the vertex set of a graph can be covered by at most k cliques or, after complementation, by at most k independent sets are NP-complete [Karp 1972]. These problems are known as CLIQUE COVER and GRAPH COLOURABILITY respectively. In fact, it is already an NP-complete problem to determine if a planar graph with vertex degree at most 4 is 3-colourable [Garey *et al.* 1976C], whereas 2-colourability is equivalent to bipartiteness and can be checked in polynomial time.

3.3. SET PACKING, SET COVER & SET PARTITION

SET PACKING: Given a finite set S, a finite family $\mathcal{S}$ of subsets of S and an integer ℓ, does $\mathcal{S}$ contain a subfamily $\mathcal{S}'$ of at least ℓ pairwise disjoint sets?

SET COVER: Given a finite set S, a finite family $\mathcal{S}$ of subsets of S and an integer ℓ, does $\mathcal{S}$ contain a subfamily $\mathcal{S}'$ of at most ℓ sets such that $\bigcup_{S' \in \mathcal{S}'} S' = S$?

SET PARTITION (EXACT COVER): Given a finite set S and a finite family $\mathcal{S}$ of subsets of S, does $\mathcal{S}$ contain a subfamily $\mathcal{S}'$ of pairwise disjoint sets such that $\bigcup_{S' \in \mathcal{S}'} S' = S$?

NP-completeness

VERTEX PACKING $\propto$ SET PACKING:

$S = E'$;

$\mathcal{S} = \{\{\{i,j\} \mid \{i,j\} \in E'\} \mid i \in V'\}$;

$\ell = k'$.

VERTEX COVER $\propto$ SET COVER:

delete the primes in the above reduction.

VERTEX PACKING and VERTEX COVER are easily recognized as special cases of SET PACKING and SET COVER respectively, and these reductions require no further comment.

VERTEX PACKING $\propto$ SET PARTITION:

$S = E' \cup \{1,\ldots,k'\}$;

$\mathcal{S} = \{S_{ih} \mid i \in V', h = 1,\ldots,k'\} \cup \{S_{\{i,j\}} \mid \{i,j\} \in E'\}$, where

$S_{ih} = \{\{i,j\} \mid \{i,j\} \in E'\} \cup \{h\}$,

$S_{\{i,j\}} = \{\{i,j\}\}$.

Cf. Figure 6. Suppose that G' contains an independent set $U' \subset V'$ of size k', say, $U' = \{v_1,\ldots,v_{k'}\}$. Then the sets $S_{v_1 1},\ldots,S_{v_{k'},k'}$ are pairwise disjoint, and the elements of S not contained in any of them belong to E'. It follows that a partition of S is given by

$$\{S_{v_1 1},\ldots,S_{v_{k'},k'}\} \cup \{S_{\{i,j\}} \mid \{i,j\} \in E', i \notin U', j \notin U'\}.$$

Conversely, suppose that there exists a partition S' of S. Then S' contains k' pairwise disjoint sets $S_{v_1,1},\ldots,S_{v_{k'},k'}$, and the vertices $v_1,\ldots,v_{k'}$ clearly constitute an independent set of size k' in G'.

This reduction simplifies the NP-completeness proof given in [Karp 1972].

S	S_{11}	S_{21}	S_{31}	S_{41}	S_{51}	S_{12}	S_{22}	S_{32}	S_{42}	S_{52}	S_{13}	S_{23}	S_{33}	S_{43}	S_{53}	$S_{\{1,2\}}$	$S_{\{2,3\}}$	$S_{\{2,4\}}$	$S_{\{3,4\}}$	$S_{\{4,5\}}$
{1,2}	◉	•				•	•				•	•				•				
{2,3}		•	•				•	◉				•	•				•			
{2,4}		•		•			•		•			•		•				◉		
{3,4}			•	•				◉	•				•	•					•	
{4,5}				•	•				•	•				•	◉					•
1	◉	•	•	•	•															
2						•	•	◉	•	•										
3											•	•	•	•	◉					

Figure 6 Instance of SET PARTITION for the example.

Related results

Even the EXACT 3-COVER problem, where all subsets in S are constrained to be of size 3, is NP-complete, since it is an obvious generalization of the 3-DIMENSIONAL MATCHING problem, proved NP-complete in [Karp 1972]. An EXACT 2-COVER corresponds to a perfect matching in a graph, which can be found in polynomial time. Generally, the existence of good matching algorithms proves that EDGE PACKING and EDGE COVER problems are members of P.

3.4. DIRECTED & UNDIRECTED HAMILTONIAN CIRCUIT

DIRECTED HAMILTONIAN CIRCUIT: Given a directed graph H = (W,A), does H contain a directed cycle passing through each vertex exactly once?

UNDIRECTED HAMILTONIAN CIRCUIT: Given an undirected graph G = (V,E), does G contain a cycle passing through each vertex exactly once?

NP-completeness

VERTEX COVER $\propto$ DIRECTED HAMILTONIAN CIRCUIT:

$$W = \{(i,j),\{i,j\},(j,i) \mid \{i,j\} \in E\} \cup \{1,\ldots,k\};$$

$$A = \{((i,j),\{i,j\}),(\{i,j\},(i,j)),((j,i),\{i,j\}),(\{i,j\},(j,i)) \mid \{i,j\} \in E\}$$
$$\cup \{((h,i),(i,j)) \mid \{h,i\},\{i,j\} \in E,\ h \neq j\}$$
$$\cup \{((i,j),h),(h,(i,j)),((j,i),h),(h,(j,i)) \mid \{i,j\} \in E,\ h = 1,\ldots,k\}.$$

Cf. Figure 7. For each edge $\{i,j\}$ in G we have created a configuration in H consisting of three vertices $(i,j),\{i,j\},(j,i)$ and four arcs, as shown in the figure. These configurations are linked by arcs from (h,i) to (i,j) for $h \neq j$. Further, we have added k vertices $1,\ldots,k$ and all arcs between them and the vertices of type (i,j).

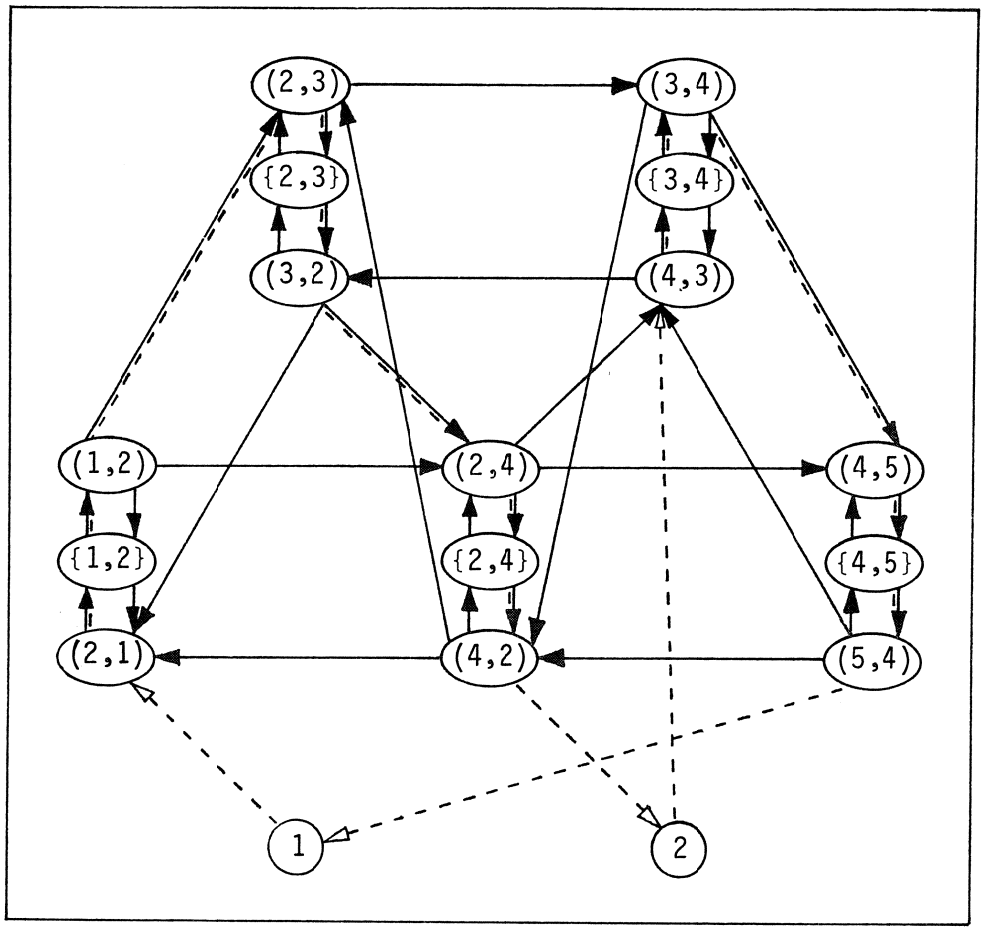

Figure 7 Instance of DIRECTED HAMILTONIAN CIRCUIT for the example. Not all arcs incident with vertices $1,\ldots,k$ have been drawn.

Suppose that G contains a vertex cover $U \subset V$ of size k, say, $U = \{v_1,\ldots,v_k\}$. The edge set E can then be written as

$$E = \{\{v_h,w_{h1}\},\ldots,\{v_h,w_{h\ell_h}\} \mid h = 1,\ldots,k\}$$

and it is easily checked that a hamiltonian circuit in H is given by

$$(1,(v_1,w_{11}),\{v_1,w_{11}\},(w_{11},v_1),\ldots,(v_1,w_{1\ell_1}),\{v_1,w_{1\ell_1}\},(w_{1\ell_1},v_1),$$
$$\ldots$$
$$k,(v_k,w_{k1}),\{v_k,w_{k1}\},(w_{k1},v_k),\ldots,(v_k,w_{k\ell_k}),\{v_k,w_{k\ell_k}\},(w_{k\ell_k},v_k),$$
$$1).$$

Conversely, suppose that H contains a hamiltonian circuit. By deletion of all arcs incident with vertices $1,\ldots,k$, the circuit is decomposed into k paths. A path starting at (i,j) for $\{i,j\} \in E$ has to go on to visit $\{i,j\}$ and (j,i); then it ends or goes on to visit $(i,j'),\{i,j'\},(j',i)$ for some $\{i,j'\} \in E$, etc. Thus, this path corresponds to a specific vertex $i \in V$, covering edges $\{i,j\},\{i,j'\}$, etc. Since the circuit passes through each $\{i,j\}$ exactly once, each edge $\{i,j\} \in E$ is covered by one of k specific vertices, which therefore constitute a vertex cover of size k in G.

The above reduction is a modification of the original construction due to E.L. Lawler [Karp 1972], based on ideas of M. Fürer [Schuster 1976] and P. van Emde Boas.

DIRECTED HAMILTONIAN CIRCUIT $\propto$ UNDIRECTED HAMILTONIAN CIRCUIT:

$$V = \{(i,in),(i,mid),(i,out) \mid i \in W\};$$
$$E = \{\{(i,in),(i,mid)\},\{(i,mid),(i,out)\} \mid i \in W\}$$
$$\cup \{\{(i,out),(j,in)\} \mid (i,j) \in A\}.$$

The one-one correspondence between undirected hamiltonian circuits in G and directed hamiltonian circuits in H is evident. This reduction is due to R.E. Tarjan [Karp 1972].

Related results

The above results have been strengthened in various ways. For instance, the UNDIRECTED HAMILTONIAN CIRCUIT problem remains NP-complete if G has vertex degree at most 3 [Garey *et al.* 1976C] or if G is bipartite

[Krishnamoorthy 1975]. The latter result is a simple extension of the last reduction given above and we recommend it as an exercise.

NP-completeness of the (general) TRAVELLING SALESMAN problem is another obvious consequence. Intricate NP-completeness proofs for the EUCLIDEAN TRAVELLING SALESMAN problem can be found in [Papadimitriou 1975; Garey *et al.* 1976A]. It is well known that TRAVELLING SALESMAN is a special case of the problem of finding a maximum weight independent set in the intersection of three matroids. Thus, the 3-MATROID INTERSECTION problem is NP-complete, whereas 2-MATROID INTERSECTION problems, such as finding an optimal linear assignment or spanning arborescence, belong to P [Lawler 1976].

The TRAVELLING SALESMAN problem serves as a prototype for a whole class of routing problems where, given a mixed graph consisting of a set V of vertices, a set E of (undirected) edges and a set A of (directed) arcs, a salesman has to find a minimum-weight tour passing through subsets $V' \subset V$, $E' \subset E$ and $A' \subset A$. If $V' = \emptyset$, $E' = E$ and $A' = A$, we have the CHINESE POSTMAN problem, which can be solved in polynomial time in the undirected or directed case ($A = \emptyset$ or $E = \emptyset$) [Edmonds 1965B; Edmonds & Johnson 1973], but is NP-complete in the mixed case [Papadimitriou 1976]. For the case that only $V' = \emptyset$, NP-completeness has been established for the UNDIRECTED and DIRECTED RURAL POSTMAN problems ($A = \emptyset$ and $E = \emptyset$ respectively) [Lenstra & Rinnooy Kan 1976] and for the STACKER-CRANE problem ($E' = \emptyset$, $A' = A$) [Frederickson *et al.* 1976].

3.5. 0-1 PROGRAMMING, KNAPSACK & 3-PARTITION

0-1 PROGRAMMING: Given an integer matrix A and an integer vector b, does there exist a 0-1 vector x such that $Ax \geq b$?

KNAPSACK: Given positive integers $a_1,\ldots,a_t,b$, does there exist a subset $S \subset \{1,\ldots,t\}$ such that $\sum_{j\in S} a_j = b$?

3-PARTITION: Given positive integers $a_1,\ldots,a_{3t},b$, do there exist t pairwise disjoint 3-element subsets $S_i \subset \{1,..,3t\}$ such that $\sum_{j\in S_i} a_j = b$ $(i = 1,\ldots,t)$?

NP-completeness

SATISFIABILITY $\propto$ 0-1 PROGRAMMING:

$$a_{ij} = \left\{ \begin{array}{l} 1 \text{ if } x_j \text{ is a literal in clause } C_i, \\ -1 \text{ if } \overline{x}_j \text{ is a literal in clause } C_i, \\ 0 \text{ otherwise} \end{array} \right\} \quad (i = 1,\ldots,s,\ j = 1,\ldots,t);$$

$$b_i = 1 - |\{j \mid \overline{x}_j \text{ is a literal in clause } C_i\}| \quad (i = 1,\ldots,s).$$

Cf. Figure 8 and [Karp 1972].

$$\begin{aligned} x_1 &\geq 1 \\ -x_1 + x_2 - x_3 &\geq -1 \\ x_3 &\geq 1 \end{aligned}$$

Figure 8 Instance of 0-1 PROGRAMMING for the example.

SET PARTITION $\propto$ KNAPSACK:

Given $S = \{e_1,\ldots,e_s\}$ and $\mathcal{S} = \{S_1,\ldots,S_t\}$, we define

$$\varepsilon_{ij} = \left\{ \begin{array}{l} 1 \text{ if } e_i \in S_j \\ 0 \text{ if } e_i \notin S_j \end{array} \right\} \quad (i = 1,\ldots,s,\ j = 1,\ldots,t),$$

$$u = t+1,$$

and specify the reduction by

$$a_j = \sum_{i=1}^{s} \varepsilon_{ij} u^{i-1} \quad (j = 1,\ldots,t);$$

$$b = (u^s - 1)/t.$$

Cf. Figure 9. The one-one correspondence between solutions to KNAPSACK and SET PARTITION is easily verified [Karp 1972].

Given this result, the reader should have little difficulty in establishing NP-completeness for the PARTITION problem, *i.e.* KNAPSACK with $\sum_{j=1}^{t} a_j = 2b$.

3-PARTITION has been proved NP-complete through a complicated sequence of reductions, which can be found in [Garey & Johnson 1975A].

ε_{ij} ↓→	(1)	2	3	4	5	6	7	(8)	9	10	11	12	13	14	(15)	16	17	(18)	19	20	a_j	b
1	(1)	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0	0	0	0	$\varepsilon_{1j}\cdot u^0+$	u^0+
2	0	1	1	0	0	0	1	(1)	0	0	0	1	1	0	0	0	1	0	0	0	$\varepsilon_{2j}\cdot u^1+$	u^1+
3	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	(1)	0	0	$\varepsilon_{3j}\cdot u^2+$	u^2+
4	0	0	1	1	0	0	0	(1)	1	0	0	0	1	1	0	0	0	0	1	0	$\varepsilon_{4j}\cdot u^3+$	u^3+
5	0	0	0	1	1	0	0	0	1	1	0	0	0	1	(1)	0	0	0	0	1	$\varepsilon_{5j}\cdot u^4+$	u^4+
6	(1)	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\varepsilon_{6j}\cdot u^5+$	u^5+
7	0	0	0	0	0	1	1	(1)	1	1	0	0	0	0	0	0	0	0	0	0	$\varepsilon_{7j}\cdot u^6+$	u^6+
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	(1)	0	0	0	0	0	$\varepsilon_{8j}\cdot u^7$	u^7

Figure 9 Instance of KNAPSACK for the example, where s = 8, t = 20, u = 21.

Binary vs. unary encoding

KNAPSACK was the first example of an NP-complete problem involving numerical data. The size of a problem instance is O(t log b) in the standard *binary* encoding and O(tb) if a *unary* encoding is allowed. Readers will have noticed that the reduction SET PARTITION $\propto$ KNAPSACK is polynomial-bounded only with respect to a binary encoding. Indeed, KNAPSACK can be solved by dynamic programming in O(tb) time [Bellmore & Dreyfus 1962], which might be called a *pseudopolynomial* algorithm in the sense that it is polynomial-bounded only with respect to a unary encoding. Thus, the *binary NP-completeness* of KNAPSACK and its *unary membership of* P are perfectly compatible results, although it tends to make us think of KNAPSACK as less hard than other NP-complete problems.

3-PARTITION was the first example of a problem involving numerical data that remains NP-complete even if we measure the problem size by using the actual numbers involved instead of their logarithms. This *strong* or *unary NP-completeness* of 3-PARTITION indicates that already the existence of a pseudopolynomial algorithm for its solution would imply that $P = NP$ [Garey & Johnson 1976C].

Quite often, a binary NP-completeness proof involving KNAPSACK or PARTITION can be converted to a unary NP-completeness proof involving 3-

PARTITION in a straightforward manner. Occasionally, however, the polynomial-boundedness of a reduction depends essentially on the allowability of a unary encoding for 3-PARTITION. An example of such a reduction is given in the next section.

3.6. 3-MACHINE UNIT-TIME JOB-SHOP

3-MACHINE UNIT-TIME JOB-SHOP: Given 3-machines M_1, M_2, M_3 each of which can process at most one job at a time, n jobs $J_1, \ldots, J_n$ where J_j $(j = 1, \ldots n)$ consists of a chain of unit-time operations, the h-th of which has to be processed on machine μ_{jh} with $\mu_{jh} \neq \mu_{j,h-1}$ for $h > 1$, and an integer k, does there exist a schedule with length at most k?

NP-completeness

3-PARTITION $\propto$ 3-MACHINE UNIT-TIME JOB-SHOP:

$n = 3t+2;$

$\mu_j = (M_1, M_3, [M_1, M_2]^{a_j}, M_3) \quad (j = 1, \ldots, 3t);$

$\mu_{n-1} = ([M_2, M_3, M_2, M_1, M_2, M_1, [M_2, M_3]^b, M_1, M_2, M_1]^t);$

$\mu_n = ([M_3, M_2, M_3, M_2, M_1, M_2, [M_3, M_1]^b, M_2, M_1, M_2]^t);$

$k = (2b+9)t;$

where $[s]^h = s, [s]^{h-1}$ for $h > 1$ and $[s]^1 = s$.

Note that both J_{n-1} and J_n consist of a chain of operations of length equal to the threshold k. We may assume the h-th operations of these chains to be completed at time h, since otherwise the schedule length would exceed k. This leaves a pattern of idle machines for the other jobs that can be described as

$$([[M_1]^3, [M_3]^3, [M_1, M_2]^b, [M_3]^3]^t)$$

(*cf.* Figure 10). We will show that this pattern can be filled properly if and only if 3-PARTITION has a solution.

Suppose that 3-PARTITION has a solution $(S_1, \ldots, S_t)$. In this case, processing J_j with $j \in S_i$ entirely within the interval $[(2b+9)(i-1), (2b+9)i]$ $(j = 1, \ldots, 3t,\ i = 1, \ldots, t)$ yields a schedule with length k.

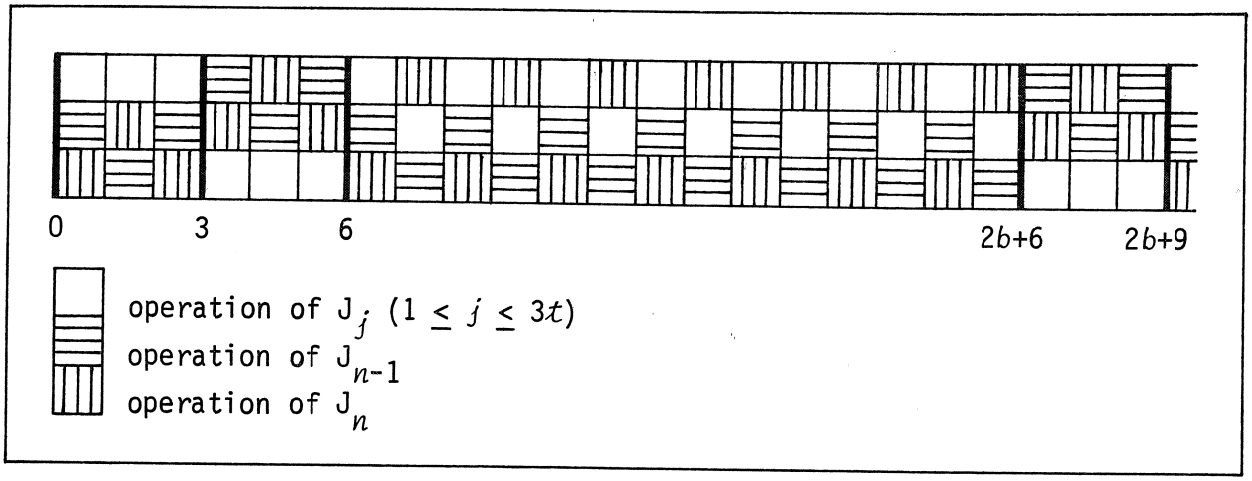

Figure 10 First part of 3-MACHINE UNIT-TIME JOB-SHOP schedule corresponding to an instance of 3-PARTITION with b = 7.

Conversely, suppose that there exists a schedule with length k. We will prove that in such a schedule exactly three jobs are started in $[0,2b+9]$ and that they are completed in this interval as well; clearly, these jobs indicate a 3-element subset S_1 with $\sum_{j \in S_1} a_j = b$. One easily proves by induction that S_i is similarly defined by the jobs started and completed in $[(2b+9)(i-1),(2b+9)i]$ $(1 < i \leq t)$.

If J starts in $[0,2b+9]$, its subchain of operations completed in that interval is of one of four types:

type 1: (M_1);

type 2: $(M_1,M_3,[M_1,M_2]^h)$ $\quad (0 \leq h \leq a_j)$;

type 3: $(M_1,M_3,[M_1,M_2]^h,M_1)$ $\quad (0 \leq h < a_j)$;

type 4: $(M_1,M_3,[M_1,M_2]^{a_j},M_3)$.

Let x_i denote the number of subchains of type i and y_i the number of operations on M_2 in subchains of type i. We have to prove that $x_1 = x_2 = x_3 = 0$, $x_4 = 3$. Observing that a schedule of length k contains no idle unit-time periods, we have

(1) $x_1 + x_2 + y_2 + 2x_3 + y_3 + x_4 + y_4 = b+3$;

(2) $y_2 + y_3 + y_4 = b$;

(3) $x_2 + x_3 + 2x_4 = 6$.

Subtracting (1) from the sum of (2) and (3), we obtain $-x_1-x_3+x_4 = 3$ and therefore $x_4 \geq 3$. Also, (3) implies that $x_4 \leq 3$. It follows that $x_4 = 3$, and $x_1 = x_2 = x_3 = 0$.

Related results

The complexity of the 2-MACHINE UNIT-TIME JOB-SHOP problem is unknown; to introduce a competitive element we shall be happy to award a chocolate windmill to the first person establishing membership of P or NP-completeness for this problem. If the processing times of the operations are allowed to be equal to 1 or 2, the 2-machine problem can be proved NP-complete by a reduction similar to (but simpler than) the above one; this improves upon related results given in [Garey *et al.* 1976B; Lenstra *et al.* 1977]. If each job has at most two operations, the 2-machine problem belongs to P even for arbitrary processing times [Jackson 1956].

These results form but a small fraction of the extensive complexity analysis carried out for scheduling problems. We refer to [Ullman 1975; Garey & Johnson 1975A; Coffman 1976; Garey *et al.* 1976B; Lenstra *et al.* 1977; Lenstra & Rinnooy Kan 1977] for further details and to [Graham *et al.* 1977] for a concise survey of the field.

4. CONCLUDING REMARKS

We hope that the preceding section has conveyed some of the flavour and elegance of NP-completeness results. In only a few years an impressive amount of results has been obtained. Nevertheless, there are still plenty of *open problems*, for which neither a polynomial algorithm nor an NP-completeness proof is available. We shall mention three famous ones, on whose complexity status little or no progress has been made so far.

(a) GRAPH ISOMORPHISM

This is the problem of determining whether there exists a one-one mapping between the vertex sets of two graphs which preserves the adjacency relation. The essential nature of the problem does not change if we restrict our attention to graphs of certain types such as bipartite, regular or series-parallel ones; all these problems are polynomially equivalent to the general case [Booth 1976]. The status of the problem is totally unknown and we do not dare to guess the final outcome.

(b) 3-MACHINE UNIT-TIME PARALLEL-SHOP

This problem involves the scheduling of unit-time jobs on three identical parallel machines subject to precedence constraints between the jobs, so as to minimize the length of the schedule. For a variable number of machines, the problem is NP-complete [Ullman 1975; Lenstra & Rinnooy Kan 1977]; the special case of tree-type precedence constraints can be solved in polynomial time [Hu 1961]. The 2-machine problem belongs to P [Coffman & Graham 1972], even if for each job a time-interval is specified in which it has to be processed [Garey & Johnson 1975B]. The 3-machine problem has remained open in spite of vigorous attacks. In this case we would be willing to extrapolate on the magic quality of three-ness and conjecture NP-completeness.

(c) LINEAR PROGRAMMING

This is perhaps the most vexing open problem. The simplex method performs very well in practice and usually requires time linear in the number of constraints. On certain weird polytopes, however, it takes exponential time [Klee & Minty 1972]. Fortunately, in this case there is circumstantial evidence againts NP-completeness. Thanks to duality theory, determining the existence or nonexistence of a feasible solution are equally hard problems, and NP-completeness of LINEAR PROGRAMMING would therefore imply NP-completeness for the complements of all other NP-complete problems as well. However, it is not even known whether the complement of any NP-complete problem belongs to NP. There is no obvious way, for instance, to check the nonexistence of a hamiltonian circuit in nondeterministic polynomial time. In addition to the above rather technical argument, it seems highly unlikely that all NP-complete problems would allow a polynomial-bounded linear programming formulation.

Interpretation of NP-completeness results as more or less definite proofs of computational intractability has stimulated the design and analysis of fast *approximation algorithms*.

With respect to the *worst-case analysis* of such algorithms, a wide variety of outcomes is possible. We give the following examples.

(1) For the optimization version of the KNAPSACK problem, a solution with-

in an arbitrary percentage ε from the optimum can be found in time polynomial in t and ε [Ibarra & Kim 1975].

(2) For the EUCLIDEAN TRAVELLING SALESMAN problem, a solution within 50% from the optimum can be found in polynomial time [Christofides 1976].

(3) For the GRAPH COLOURABILITY problem, a solution within 100% from the optimum cannot be found in polynomial time unless $P = NP$ [Garey & Johnson 1976A].

(4) For the general TRAVELLING SALESMAN problem, a solution within any fixed percentage from the optimum cannot be found in polynomial time unless $P = NP$ [Sahni & Gonzalez 1976].

We refer to [Garey & Johnson 1976B] for a survey of this area. Impressive advances have been made and more can be expected in the near future.

The above approach to *performance guarantees* may be accused of being overly pessimistic - *cf.* the simplex method with its exponential worst-case behaviour! The *probabilistic analysis* of average or "almost everywhere" behaviour, however, requires the specification of a probability distribution over the set of all problem instances. For some problems, a natural distribution function is available and some intriguing results have been derived [Karp 1976], although technically this approach seems to be very demanding.

The worst-case analysis of approximation algorithms shows that there are significant differences in complexity within the class of NP-complete problems. These problems might be classifiable according to the best possible polynomial-time performance guarantee that one can get. Another refinement of the complexity measure may be based on the way in which numerical problem data are encoded, *i.e.* on the distinction between binary and unary encoding mentioned in Section 3.5. Several other ways of measuring problem size could be devised and each of them could be subjected to a complexity analysis, producing new information on the best type of algorithm that is likely to exist.

The concluding remarks above were intended to confirm to the reader that the field of computational complexity is still very much alive. In the first place, however, the theory of NP-completeness has yielded highly useful tools for the analysis of combinatorial problems that deserve to find acceptance in a wide circle of researchers and practitioners.

REFERENCES

A.V. AHO, J.E. HOPCROFT, J.D. ULLMAN (1974) *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass.

R.E. BELLMAN, S.E. DREYFUS (1962) *Applied Dynamic Programming*. Princeton University Press, Princeton, N.J.

K.S. BOOTH (1976) Problems polynomially equivalent to graph isomorphism. In: J.F. TRAUB (ed.) (1976) *Algorithms and Complexity: New Directions and Recent Results*. Academic Press, New York, 435.

N. CHRISTOFIDES (1976) Worst-case analysis of a new heuristic for the travelling salesman problem. *Math. Programming*, to appear.

E.G. COFFMAN (ed.) (1976) *Computer and Job-Shop Scheduling Theory*. Wiley, New York.

E.G. COFFMAN, Jr., R.L. GRAHAM (1972) Optimal scheduling for two-processor systems. *Acta Informat.* 1, 200-213.

S.A. COOK (1971) The complexity of theorem-proving procedures. *Proc. 3rd Annual ACM Symp. Theory Comput.*, 151-158.

E.A. DINIC (1970) Algorithms for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Dokl.* 11, 1277-1280.

J. EDMONDS (1965A) Paths, trees, and flowers. *Canad. J. Math.* 17, 449-467.

J. EDMONDS (1965B) The Chinese postman's problem. *Operations Res.* 13 Suppl. 1,B73.

J. EDMONDS, E.L. JOHNSON (1973) Matching, Euler tours and the Chinese postman. *Math. Programming* 5, 88-124.

S. EVEN (1976) The max flow algorithm of Dinic and Karzanov: an exposition. Department of Computer Science, Technion, Haifa.

M.L. FISHER (1976) Private communication.

G.N. FREDERICKSON, M.S. HECHT, C.E. KIM (1976) Approximation algorithms for some routing problems. *Proc. 17th Ann. Symp. Foundations of Computer Science*, 216-227.

H.N. GABOW (1976) An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *J. Assoc. Comput. Mach.* 23, 221-234.

M.R. GAREY, R.L. GRAHAM, D.S. JOHNSON (1976A) Some NP-complete geometric problems. *Proc. 8th Annual ACM Symp. Theory Comput.*, 10-22.

M.R. GAREY, D.S. JOHNSON (1975A) Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.* 4, 397-411.

M.R. GAREY, D.S. JOHNSON (1975B) Two-processor scheduling with start-times and deadlines. *SIAM J. Comput.*, to appear.

M.R. GAREY, D.S. JOHNSON (1976A) The complexity of near-optimal graph coloring. *J. Assoc. Comput. Mach.* 23, 43-49.

M.R. GAREY, D.S. JOHNSON (1976B) Approximation algorithms for combinatorial problems: an annotated bibliography. In: J.F. TRAUB (ed.) (1976) *Algorithms and Complexity: New Directions and Recent Results.* Academic Press, New York, 41-52.

M.R. GAREY, D.S. JOHNSON (1976C) "Strong" NP-completeness results: motivation, examples and implications. To appear.

M.R. GAREY, D.S. JOHNSON (1978) *Computers and Intractability: a Guide to the Theory of NP-Completeness.* To appear.

M.R. GAREY, D.S. JOHNSON, R. SETHI (1976B) The complexity of flowshop and jobshop scheduling. *Math. Operations Res.* 1, 117-129.

M.R. GAREY, D.S. JOHNSON, L. STOCKMEYER (1976C) Some simplified *NP*-complete graph problems. *Theoret. Comput. Sci.* 1, 237-267.

R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1977) Optimization and approximation in deterministic sequencing and scheduling: a survey. Presented at Discrete Optimization 1977, Vancouver, August 8-12.

T.C. HU (1961) Parallel sequencing and assembly line problems. *Operations Res.* 9, 841-848.

O.H. IBARRA, C.E. KIM (1975) Fast approximation algorithms for the knapsack and sum of subset problems. *J. Assoc. Comput. Mach.* 22, 463-468.

J.R. JACKSON (1956) An extension of Johnson's results on job lot scheduling. *Naval Res. Logist. Quart.* 3, 201-203.

D.S. JOHNSON (1973) Near-optimal bin packing algorithms. Report MAC TR-109, Massachusetts Institute of Technology, Cambridge, Mass.

R.M. KARP (1972) Reducibility among combinatorial problems. In: R.E. MILLER, J.W. THATCHER (eds.) (1972) *Complexity of Computer Computations.* Plenum Press, New York, 85-103.

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

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

A.V. KARZANOV (1974) Determining the maximal flow in a network by the method of preflows. *Soviet Math. Dokl.* 15, 434-437.

V. KLEE, G.J. MINTY (1972) How good is the simplex algorithm? In: O. SHISHA (ed.) (1972) *Inequalities III*. Academic Press, New York, 159-175.

D.E. KNUTH (1974) A terminological proposal. *SIGACT News* 6.1, 12-18.

M.S. KRISHNAMOORTHY (1975) An NP-hard problem in bipartite graphs. *SIGACT News* 7.1, 26.

E.L. LAWLER (1976) *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York.

J.K. LENSTRA, A.H.G. RINNOOY KAN (1976) On general routing problems. *Networks* 6, 273-280.

J.K. LENSTRA, A.H.G. RINNOOY KAN (1977) Complexity of scheduling under precedence constraints. *Operations Res.*, to appear.

J.K. LENSTRA, A.H.G. RINNOOY KAN, P. BRUCKER (1977) Complexity of machine scheduling problems. *Ann. Discrete Math.* 1, to appear.

C.H. PAPADIMITRIOU (1975) The euclidean traveling salesman problem is NP-complete. *Theoret. Comput. Sci.*, to appear.

C.H. PAPADIMITRIOU (1976) On the complexity of edge traversing. *J. Assoc. Comput. Mach.* 23, 544-554.

E.M. REINGOLD, J. NIEVERGELT, N. DEO (1977) *Combinatorial Computing*. To appear.

S. SAHNI, T. GONZALEZ (1976) P-complete approximation problems. *J. Assoc. Comput. Mach.* 23, 555-565.

J.E. SAVAGE (1976) *The Complexity of Computing*. Wiley, New York.

P. SCHUSTER (1976) Probleme, die zum Erfüllungsproblem der Aussagenlogik polynomial äquivalent sind. In: E. SPECKER, V. STRASSEN (eds.) (1976) *Komplexität von Entscheidungsproblemen: ein Seminar*. Lecture Notes in Computer Science 43, Springer, Berlin, 36-48.

J.D. ULLMAN (1975) *NP*-complete scheduling problems. *J. Comput. System Sci.* 10, 384-393.