REDUCEIBILITY AMONG COMBINATORIAL PROBLEMS

Richard M. Karp
University of California at Berkeley

Abstract: A large class of computational problems involve the determination of properties of graphs, digraphs, integers, arrays of integers, finite families of finite sets, boolean formulas and elements of other countable domains. Through simple encodings from such domains into the set of words over a finite alphabet these problems can be converted into language recognition problems, and we can inquire into their computational complexity. It is reasonable to consider such a problem satisfactorily solved when an algorithm for its solution is found which terminates within a number of steps bounded by a polynomial in the length of the input. We show that a large number of classic unsolved problems of covering, matching, packing, routing, assignment and sequencing are equivalent, in the sense that either each of them possesses a polynomial-bounded algorithm or none of them does.

1. INTRODUCTION

All the general methods presently known for computing the chromatic number of a graph, deciding whether a graph has a Hamilton circuit, or solving a system of linear inequalities in which the variables are constrained to be 0 or 1, require a combinatorial search for which the worst case time requirement grows exponentially with the length of the input. In this paper we give theorems which strongly suggest, but do not imply, that these problems, as well as many others, will remain intractable permanently.

This research was partially supported by National Science Foundation Grant GP-474.
We are specifically interested in the existence of algorithms that are guaranteed to terminate in a number of steps bounded by a polynomial in the length of the input. We exhibit a class of well-known combinatorial problems, including those mentioned above, which are equivalent, in the sense that a polynomial-time algorithm for any one of them would effectively yield a polynomial-bounded algorithm for all. We also show that, if these problems do possess polynomial-bounded algorithms then all the problems in an unexpectedly wide class (roughly speaking, the class of problems solvable by polynomial-depth backtrack search) possess polynomial-bounded algorithms.

The following is a brief summary of the contents of the paper. For the sake of definiteness our technical development is carried out in terms of the recognition of languages by one-tape Turing machines, but any of a wide variety of other abstract models of computation would yield the same theory. Let \( L^* \) be the set of all finite strings of 0's and 1's. A subset of \( L^* \) is called a language. Let \( P \) be the class of languages recognizable in polynomial time by one-tape deterministic Turing machines, and let \( NP \) be the class of languages recognizable in polynomial time by one-tape nondeterministic Turing machines. Let \( \Sigma \) be the class of functions from \( L^* \) into \( L^* \) computable in polynomial time by one-tape Turing machines. Let \( L \) and \( M \) be languages. We say that \( L \cdot M \) (\( L \) is reducible to \( M \)) if there is a function \( f \) such that \( f(x) \in M \leftrightarrow x \in L \). If \( L \in P \) and \( L \cdot M \) then \( L \cdot M \in P \).

The main contribution of this paper is the demonstration that a large number of classic difficult computational problems, arising in fields such as mathematical programming, graph theory, combinatorics, computational logic and switching theory, are complete (and hence equivalent) when expressed in a natural way as language recognition problems.

This paper was stimulated by the work of Stephen Cook (1971), and rests on an important theorem which appears in his paper. The author also wishes to acknowledge the substantial contributions of Eugene Lawler and Robert Tarjan.

2. THE CLASS \( \Sigma \)

There is a large class of important computational problems which involve the determination of properties of graphs, digraphs, integers, finite families of finite sets, boolean formulas and elements of other computable functions. Characterization of problems in \( \Sigma \) are now widely accepted, if only if the running time is bounded in polynomial time in the length of the input.

We begin by giving the following algorithms, \( P \) into a countable \( \mathbb{R} \) for any finite set of elements of \( P \) and a string of elements of \( L \) of length \( x \).

A deterministic algorithm

\[ f(x) = \begin{cases} 1 & \text{if } x \in P \\ 0 & \text{otherwise} \end{cases} \]

A nondeterministic algorithm

\[ f(x) = \begin{cases} 1 & \text{if } x \in \mathbb{R} \\ 0 & \text{otherwise} \end{cases} \]

A transition in the computation of \( A \)

\[ f(x) = \begin{cases} 1 & \text{if } x \in \mathbb{R} \\ 0 & \text{otherwise} \end{cases} \]

To discuss algorithms the concept of a transit is familiar. A transit in a graph is a path that begins at a node and ends at another node, with no repeated nodes. A transit in a graph is a path that begins at a node and ends at another node, with no repeated nodes.
elements of other countable domains. It is a reasonable working hypothesis, championed originally by Jack Edmonds (1965) in connection with problems in graph theory and integer programming, and by now widely accepted, that such a problem can be regarded as tractable if and only if there is an algorithm for its solution whose running time is bounded by a polynomial in the size of the input. In this section we introduce and begin to investigate the class of problems solvable in polynomial time.

We begin by stating an extremely general definition of 'deterministic algorithms', computing a function from a countable domain \( D \) into a countable range \( B \).

For any finite alphabet \( A \), let \( A^* \) be the set of finite strings of elements of \( A \); i.e., \( x \in A^* \), let \( |x| \) denote the length of \( x \).

A deterministic algorithm \( A \) is specified by:

- A countable set \( D \) (the domain)
- A countable set \( B \) (the range)
- A finite alphabet \( \Sigma \) such that \( \Sigma^*B = \{ \} \)
- An encoding function \( E : D \to \Sigma^* \)
- A transition function \( T : \Sigma^* \times \Sigma \to \Sigma^* \).

The computation of \( A \) on input \( x \in D \) is the unique sequence \( y_1, y_2, \ldots \) such that \( y_i \in E(x) \) and \( y_{i+1} = T(y_i) \) for all \( i \). Any string occurring as an element of a computation is called an instantaneous description. If the computation of \( A \) on input \( x \) is finite and of length \( |x| \), then \( x \) is called a string recognizing \( A \). A terminating algorithm \( A \) computes the function \( f_A : D \to B \) such that \( f_A(x) \) is the last element of the computation of \( A \) on \( x \).

If \( R = \{ \text{ACCEPT, REJECT} \} \) then \( A \) is called a recognition algorithm. A recognition algorithm in which \( D = \Sigma^* \) is called a string recognition algorithm. If \( A \) is a string recognition algorithm then the language recognized by \( A \) is \( \{ x \in \Sigma^* : f_A(x) = \text{ACCEPT} \} \). If \( D = \Sigma^* \) then \( A \) is called a string mapping algorithm. A terminating algorithm \( A \) with domain \( D = \Sigma^* \) operates in polynomial time if there is a polynomial \( p(\cdot) \) such that, for every \( x \in \Sigma^* \), \( |x| \leq p(|x|) \).

To discuss algorithms in any practical context we must specialize the concept of deterministic algorithm. Various well known classes of string recognition algorithms (Markov algorithms, one-tape Turing machines, multitape and multthead Turing machines).
random access machines, etc.) are delineated by restricting the func-
tions E and p to be of certain very simple types. These definitions
are standard (Hopcroft & Ullman (1969)) and will not be repeated here.
It is by no commonplace to observe that many such classes are equiva-
 lent in their capability to recognize languages; for each such
class of algorithms, the class of languages recognized is the
class of recursive languages. This invariance under changes in
definition is part of the evidence that recursiveness is the cor-
rect technical formulation of the concept of decidability.

The class of languages recognizable by string recognition
algorithms which operate in polynomial time is also invariant
under a wide range of changes in the class of algorithms. For
example, any language recognizable in time \( p(\cdot) \) by a multithread
or multitape Turing machine is recognizable in time \( p(T) \) by a
one-tape Turing machine. Thus the class of languages recognizable
in polynomial time by one-tape Turing machines is the same as the
class recognizable by the ostensibly more powerful multithread
or multitape Turing machines. Similar remarks apply to random access
machines.

Definition 1. \( F \) is the class of languages recognizable by
one-tape Turing machines which operate in polynomial time.

Definition 2. \( \Pi \) is the class of functions from \( \mathbb{Z}^n \) into \( \mathbb{Z}^n \)
defined by one-tape Turing machines which operate in polynomial time.

The reader will not go wrong by identifying \( F \) with the class
of languages recognizable by digital computers (with unbounded
storage), which operate in polynomial time and \( \Pi \) with the
class of string mappings performed in polynomial time by such
computers.

Remark. If \( \varepsilon: \mathbb{Z}^n \rightarrow \mathbb{Z}^n \) is in \( \Pi \), then there is a polynomial
\( p(\cdot) \) such that \( \varepsilon(x) \leq p(|x|) \).

We next introduce a concept of reducibility which is of cen-
tral importance in this paper.

Definition 3. Let \( L \) and \( M \) be languages. Then \( L \equiv M \)
(\( L \) is reducible to \( M \)) if there is a function \( f \in \Pi \) such that
\( f(x) \in M \leftrightarrow x \in L \).

Lemma. If \( L \equiv M \) and \( M \in F \) then \( L \in F \).

Proof. The following is a polynomial-time bounded algorithm
to decide if \( x \in L \): compute \( f(x) \); test in polynomial time
whether \( f(x) \in M \).

We will be interested in the difficulty of recognizing subsets
of countable domains other than \( \mathbb{Z}^n \). Given such a domain \( D \),

REUDUCIBILITY AMONG COM
there is usually a sim-
ples we can represent
is comprising its basic
array, etc.; and this
into strings over a
finite alphabet as s
ing \( e: D \rightarrow \mathbb{Z}^n \), we
ominal time if \( e(T) \)
and encoding function
if \( e(\cdot) = e'(\cdot) \).

As a rule, there is
possible. For instance,
nt, by its incidence
notations, there remain
and punctuation. For
any two "reasonable" exams;
es in such a way as to
starts, i.e., \( e \) con-
cerns the represent-
that a positive integral
representation. In
ominal time and
ishes problems in
hiring an encoding in

We complete this
which are solvable by
ine a number of clo-
known to be solvable by
our notation.

Each problem in
"INPUT": A general-
the heading "PROPERTY"
accepted.

SATISFIABILITY WITH
INPUT: \( C \), \( e \)
PROPERTY: \( C \) is an
i.e., there is a set
a) \( S \) does not
b) \( S \), \( T \), \( \not\in \).

MINIMUM SPANNING TREE
INPUT: \( G, w, W \)
PROPERTY: There exist
RICHARD M. KARP

restricting the function. These definitions will not be repeated here. Each class is a set of parameters in the under changes in the interpreters in the core environment.

The recognition also invariant algorithms. For a multiset by a multiset having recognizable the same as the multiset multiset and recognizable by the canonical time.

from \( \mathbb{P} \) into \( \mathbb{P} \) is polynomial

A polynomial time algorithm with the class \( \mathbb{P} \) and each with the class is polynomial in time by such

which is of bounded linear operations by the same

Then \( L = \mathbb{N} \) if \( \in \mathbb{N} \) such that \( L \in \mathbb{N} \) bounded algorithm in polynomial time recognizing subsets in a domain \( D \).

REDUCIBILITY AMONG COMBINATORIAL PROBLEMS

there is usually a natural one-one encoding \( e : D \to \mathbb{P} \). For example, we can represent a positive integer by the string of \( 0 \)'s and \( 1 \)'s comprising its binary representation, a 1-dimensional integer array as a list of integers, a matrix as a list of 1-dimensional arrays, etc.; and these are standard techniques for encoding lists into strings over a finite alphabet, and strings over an arbitrary finite alphabet as strings of \( 0 \)'s and \( 1 \)'s. Given such an encoding \( e : D \to \mathbb{P} \), we say that a set \( T \subseteq D \) is recognizable in polynomial time if \( e(T) \notin \mathbb{P} \). Also, given sets \( T \subseteq D \) and \( U \subseteq \mathbb{P} \), and encoding functions \( e : D \to \mathbb{P} \) and \( e' : \mathbb{P} \to \mathbb{P} \) we say \( T \subseteq U \) if \( e(T) = e'(U) \).

As a rule several natural encodings of a given domain are possible. For instance a graph can be represented by its adjacency matrix, by its incidence matrix, or by a list of unordered pairs of nodes, corresponding to the arcs. Given one of these representations, there remains a number of arbitrary decisions as to format and punctuation. Fortunately, it is almost always obvious that any two "reasonable" encodings \( e_0 \) and \( e_1 \) of a given problem are equivalent; i.e., \( e_0(e_1(D)) \notin \mathbb{P} \), \( e_1(e_0(D)) \notin \mathbb{P} \). An important exception concerns the representation of positive integers; we stipulate that a positive integer is encoded in a binary, rather than unary, representation. In view of the invariance of recognizability in polynomial time and reducibility under reasonable encodings, we discuss problems in terms of their original domains, without specifying an encoding into \( \mathbb{P} \).

We complete this section by listing a sampling of problems which are solvable in polynomial time. In the next section we examine a number of close relatives of these problems which are not known to be solvable in polynomial time. Appendix B establishes our notation.

Each problem is specified by giving (under the heading "INPUT") a generic element of its domain of definition and (under the heading "PROPERTY") the property which causes an input to be accepted.

SATISFIABILITY WITH AT MOST 2 LITERALS PER CLAUSE [Cook (1971)]

INPUT: \( \mathbb{C}_0, \mathbb{C}_1, \ldots, \mathbb{C}_n \), each containing at most 2 literals

PROPERTY: The conjunction of the given clauses is satisfiable; i.e., there is a set \( S \subseteq \{ x_0, x_1, \ldots, x_2, x_3, \ldots, x_n \} \) such that 

- \( S \) does not contain a complementary pair of literals and
- \( S \cap \mathbb{C}_k \neq \emptyset, k = 0, 1, \ldots, n \).

MINIMUM SPANNING TREE [Kruskal (1956)]

INPUT: \( G, W, W \)

PROPERTY: There exists a spanning tree of weight \( \leq W \).
SHORTEST PATH [Dijkstra (1959)]
INPUT: G, v, W, s, t
PROPERTY: There is a path between s and t of weight \( \leq W \).

MINIMUM CUT [Edmonds & Karp (1972)]
INPUT: G, s, t
PROPERTY: There is an s, t cut of weight \( \leq W \).

ARC COVER [Edmonds (1965)]
INPUT: G, k
PROPERTY: There is a set \( Y \subseteq A \) such that \( |Y| \leq k \) and every node is incident with an arc in \( Y \).

ARC DELETION
INPUT: G, k
PROPERTY: There is a set of \( k \) arcs whose deletion breaks all cycles.

BIPARTITE MATCHING [Hall (1948)]
INPUT: \( K \subseteq \mathbb{Z} \times \mathbb{Z} \)
PROPERTY: There are \( p \) elements of \( S \), no two of which are equal in either component.

SEQUENCING WITH DEADLINES
INPUT: \( \{T_1, \ldots, T_n\} \subseteq \mathbb{Z}^+, \{D_1, \ldots, D_n\} \subseteq \mathbb{Z}^+, k \)
PROPERTY: Starting at time \( 0 \), one can execute jobs \( 1, 2, \ldots, n \) with execution times \( T_i \) and deadlines \( D_i \), in some order such that not more than \( k \) jobs miss their deadlines.

SOLVABILITY OF LINEAR EQUATIONS
INPUT: \( \{c_{ij}\}, \{a_i\} \)
PROPERTY: There exists a vector \( \{y_j\} \) such that, for each \( i \),
\[ \sum_j c_{ij} y_j = a_i \]

3. NONDETERMINISTIC ALGORITHMS AND COOK'S THEOREM

In this section we state an important theorem due to Cook (1971) which asserts that any language in a certain wide class \( NP \) is reducible to a specific set \( S \), which corresponds to the problem of deciding whether a boolean formula is in conjunctive normal form is satisfiable.

Let \( P(2) \) denote the class of subsets of \( \mathbb{Z}^n \times \mathbb{Z}^n \) which are recognizable in polynomial time. Given \( L_2 \in P(2) \) and a polynomial \( p \), we define a language \( L \) as follows:
\[ L = \{ x | \text{there exists } y \text{ such that } \langle x, y \rangle \in L_2 \text{ and } |y| \leq p(|x|) \}. \]
RICHARD M. KARP

We refer to L as the language derived from L^{(2)} by p-bounded existential quantification.

**Definition 6.** NP is the set of languages derived from elements of L^{(2)} by polynomial-bounded existential quantification.

There is an alternative characterization of NP in terms of nondeterministic Turing machines. A **nondeterministic recognition algorithm** A is specified by:

- a countable set D (the domain)
- a finite alphabet Σ such that Δ^n × (ACCEPT, REJECT) = ∅
- an encoding function e: D → Δ^n
- a transition relation T ⊆ Δ^n × (Δ^n U {ACCEPT, REJECT}) such that, for every y_i ∈ Δ^n, the set {y_i, y_i.e(x)} ∪ T has fewer than k_A elements, where k_A is a constant. A computation of A on input x ∈ D is a sequence y_1, y_2, ..., y_k of length ≤ p(lg(x)) which occurs in some computation is an instantaneous description. A finite computation ending in ACCEPT is an accepting computation. Input x is accepted if there is an accepting computation for x. If D = Δ^n then A is a **nondeterministic string recognition algorithm** and we say that A operates in polynomial time if there is a polynomial p(·) such that, whenever A accepts x, there is an accepting computation for x of length ≤ p(lg(x)).

A nondeterministic algorithm can be regarded as a process which, when confronted with a choice between (say) two alternatives, can create two copies of itself, and follow up the consequences of both courses of action. Repeated splitting may lead to an exponentially growing number of copies; the input is accepted if any sequence of choices leads to acceptance.

The nondeterministic 1-tape Turing machines, multitape Turing machines, random-access machines, etc. define classes of nondeterministic string recognition algorithms by restricting the encoding function e and transition relation T to particularly simple forms. All these classes of algorithms, restricted to operate in polynomial time, define the same class of languages. Moreover, this class is NP.

**Theorem 1.** L ∈ NP if and only if L is accepted by a nondeterministic Turing machine which operates in polynomial time.

**Proof.** Suppose L ∈ NP. Then, for some (2) p(2) and some polynomial p, L is obtained from Λ^{(2)} by p-bounded existential quantification. We can construct a nondeterministic
machine which first guesses the successive digits of a string $y$ of length $\leq p(\lg y)$ and then tests whether $\langle x, y \rangle \in L^1$. Such a machine clearly recognizes $L$ in polynomial time.

Suppose $L$ is accepted by a non-deterministic Turing machine $T$ which operates in time $p$. Assume without loss of generality that, for any instantaneous description $Z$, there are at most $m$ instantaneous descriptions that may follow $Z$ (i.e., at most two primitive transitions are applicable). Then the sequence of choices of instantaneous descriptions made by $T$ in a given computation can be encoded as a string $y$ of 0's and 1's, such that $1^g(y) \leq p(\lg y)$.

Thus we can construct a deterministic Turing machine $T'$, with $2^m \times 2^m$ as its domain of inputs, which, on input $\langle x, y \rangle$, simulates the action of $T$ on input $x$ with the sequence of choices $y$. Clearly $T'$ operates in polynomial time, and $L$ is obtained by polynomial bounded existential quantification from the set of pairs of strings accepted by $T'$.

The class $NP$ is very extensive. Loosely, a recognition problem is in $NP$ if and only if it can be solved by a backtrack search of polynomial bounded depth. A wide range of important computational problems which are not known to be in $P$ are obviously in $NP$. For example, consider the problem of determining whether the nodes of a graph $G$ can be colored with $k$ colors so that no two adjacent nodes have the same color. A non-deterministic algorithm can simply guess an assignment of colors to the nodes and then check (in polynomial time) whether all pairs of adjacent nodes have distinct colors.

In view of the wide extent of $NP$, the following theorem due to Cook is remarkable. We define the satisfiability problem as follows:

**SATISFIABILITY**

**INPUT:** Clause $C_1, C_2, \ldots, C_p$

**PROPERTY:** The conjunction of the given clauses is satisfiable; i.e., there is a set $S \subseteq \{1, 2, \ldots, n\}$ such that:

a) $S$ does not contain a complementary pair of literals

and

b) $S \cap C_k \neq \emptyset$ for $k = 1, 2, \ldots, p$.

**Theorem 2 (Cook).** If $L \in NP$ then $L = SATISFIABILITY$.

The theorem stated by Cook (1971) uses a weaker notion of reducibility than the one used here, but Cook's proof supports the present statement.

**Corollary 1.** $P = NP \leftrightarrow SATISFIABILITY \in P$.

**Reducibility Among Complete Sets.**

Proof. If $SATISFIABILITY$ is in $P$, then $L = SATISFIABILITY$ is clearly SATISFIABILITY-complete.

Remark. If $P$ and polynomial-bounded existential quantification are also closed under polynomial reductions, it follows that a polynomial hierarchy ($\Sigma_i^p$ and $\Pi_i^p$,

Theorem 2 shows a method to decide membership of an input on a polynomially solvable by a polynomially bounded existential quantification from the set of pairs of strings accepted by $T'$.

The main object of interest is the class $NP$. We can extend the concept of a problem to the class of all problems on countable domains over countable domains.

**Definition 3.**

a) $L \in NP$ and

b) $SATISFIABILITY$ is in $NP$.

**Theorem 3.** $L$ is $NP$-complete if and only if $L$ is $NP$-hard.

We can extend the concept of a problem to the class of all problems on countable domains.

**Definition 4.**

a) $L$ is $NP$-complete if $L \in NP$ and $\forall L \in NP$.

**Theorem 4.** $L$ is $NP$-complete if and only if $L$ is $NP$-hard.

Let $a(x)$ be defined as $T'$ if there is a $b$ such that $a(x) = b$ and $b$ is a solution of an equation $T$ whenever $x \in L$.

The rest of the proof is sketched by the following theorem.
EDUCIBILITY AMONG COMBINATORIAL PROBLEMS

Proof. If SATISFIABILITY ∈ P then, for each L ∈ NP, L ∈ P, since L = SATISFIABILITY. If SATISFIABILITY ∈ P, then, since clearly SATISFIABILITY ∉ NP, P ∉ NP.

Remark. If P = NP then NP is closed under complementation and polynomial-bounded existential quantification. Hence it is also closed under polynomial-bounded universal quantification. It follows that a polynomial-bounded analogue of Kleene's Arithmetic Hierarchy [Rogers (1967)] becomes trivial if P = NP.

Theorem 2 shows that, if there were a polynomial-time algorithm to decide membership in SATISFIABILITY then every problem solvable by a polynomial-depth backtrace search would also be solvable by a polynomial-time algorithm. This is strong circumstantial evidence that SATISFIABILITY ∉ P.

4. COMPLETE PROBLEMS

The main object of this paper is to establish, that a large number of important computational problems can play the role of SATISFIABILITY in Cook's theorem. Such problems will be called complete.

Definition 5. The language L : = (polynomial) complete if
a) L ∈ NP
   and
b) SATISFIABILITY = L.

Theorem 3. Either all complete languages are in P, or none of them are. The former alternative holds if and only if P = NP.

We can extend the concept of completeness to problems defined over countable domains other than \( \mathbb{N} \).

Definition 6. Let D be a countable domain, e a "standard" one-one encoding e: D → \( \mathbb{N} \) and T a subset of D. Then T is complete if and only if \( e(T) \) is complete.

Lemma 2. Let D and D' be countable domains, with one-one encoding functions e and e'. Let T ⊆ D and T' ⊆ D'. Then T = T' if there is a function f: D → D' such that
a) f(x) ∈ T' if and only if x ∈ T
and
b) there is a function f ∈ W such that f(x) = e'(f(e(x))) for all x ∈ T.
Main Theorem. All the problems on the following list are complete.

1. SATISFIABILITY
   INPUT: A boolean formula $F$ of size $n$
   PROPERTY: Is $F$ satisfiable?

2. 0-1 INTEGER PROGRAMMING
   INPUT: integer matrix $A$ and integer vector $b$
   PROPERTY: There exists a 0-1 vector $x$ such that $Ax = b$.

3. CLIQUE
   INPUT: graph $G$, positive integer $k$
   PROPERTY: $G$ has a set of $k$ mutually adjacent nodes.

4. SET PACKING
   INPUT: family of sets $[S_i]$, positive integer $k$
   PROPERTY: $[S_i]$ contains $k$ mutually disjoint sets.

5. NODE COVER
   INPUT: graph $G'$, positive integer $k$
   PROPERTY: There is a set $R \subseteq V'$ such that $|R| \leq k$ and every arc is incident with some node in $R$.

6. SET COVERING
   INPUT: finite family of finite sets $[S_i]$, positive integer $k$
   PROPERTY: There is a subfamily $[S_{i_j}] \subseteq [S_i]$ containing $\leq k$ sets such that $U_{i_j} = U$.

7. FEEDBACK NODE SET
   INPUT: digraph $H$, positive integer $k$
   PROPERTY: There is a set $R \subseteq V$ such that every (directed) cycle of $H$ contains a node in $R$.

8. FEEDBACK ARC SET
   INPUT: digraph $H$, positive integer $k$
   PROPERTY: There is a set $R \subseteq E$ such that every (directed) cycle of $H$ contains an arc in $R$.

9. DIRECTED HAMILTON CIRCUIT
   INPUT: digraph $H$
   PROPERTY: $H$ has a directed cycle which includes each node exactly once.

10. UNDIRECTED HAMILTON CIRCUIT
    INPUT: graph $G$
    PROPERTY: $G$ has a cycle which includes each node exactly once.

11. SATISFIABILITY
    INPUT: Clause literals from a boolean formula $F$
    PROPERTY: Is $F$ satisfiable?

12. CHROMATIC NUMBER
    INPUT: graph $G$
    PROPERTY: The chromatic number of $G$ is $\chi$.

13. CLIQUE COVER
    INPUT: graph $G$
    PROPERTY: $|\mathcal{C}|$.

14. EXACT COVER
    INPUT: family of sets $[S_i]$, positive integer $k$
    PROPERTY: The sets $S_i$ are $k$-exact.

15. HITTING SET
    INPUT: family of sets $[S_i]$, positive integer $k$
    PROPERTY: $|\mathcal{H}|$.

16. STEINER TREE
    INPUT: graph $G$, positive integer $k$
    PROPERTY: $G$ of nodes in $V$.

17. 3-DIMENSIONAL MATCHING
    INPUT: set $V$, positive integer $k$
    PROPERTY: The no two elements in $V$.

18. KNAPSACK
    INPUT: $(a_i, b_i)$, positive integer $k$
    PROPERTY: $\sum$.

19. JOB SEQUENCING
    INPUT: "exec "dead"
    PROPERTY: The to be positive.

20. VERTEX COVER
    INPUT: graph $G$
    PROPERTY: $\gamma$.

$(\frac{1}{2})$ if $T_i = 1$ 
$j = 1$
11. Satisfiability with at most 3 literals per clause

INPUT: Clauses \( C_1, C_2, \ldots, C_m \), each consisting of at most 3 literals from the set \( \{y_1, y_2, \ldots, y_n\} \cup \{\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n\} \)

PROPERTY: The set \( \{y_1, y_2, \ldots, y_n\} \) is satisfiable.

12. Chromatic number

INPUT: Graph \( G \), positive integer \( k \)

PROPERTY: There is a function \( \psi : V \to \{1, 2, \ldots, k\} \) such that, if \( u \) and \( v \) are adjacent, then \( \psi(u) \neq \psi(v) \).

13. Clique cover

INPUT: Graph \( G \), positive integer \( k \)

PROPERTY: \( G \) is the union of \( k \) or fewer cliques.

14. Exact cover

INPUT: Family \( \{S_1, S_2, \ldots, S_m\} \) of subsets of \( \{a_1, a_2, \ldots, a_n\} \)

PROPERTY: There is a subfamily \( \{S_{i_1}, S_{i_2}, \ldots, S_{i_k}\} \) such that the sets \( S_{i_1}, S_{i_2}, \ldots, S_{i_k} \) are disjoint and \( \bigcup_{i=1}^k S_{i} = \{a_1, a_2, \ldots, a_n\} \).

15. Matching set

INPUT: Family \( \{U_1, U_2, \ldots, U_r\} \) of subsets of \( \{a_1, a_2, \ldots, a_n\} \)

PROPERTY: There is a set \( B \) such that, for each \( i \), \( |U_i \cap B| = 1 \).

16. Steiner tree

INPUT: Graph \( G \), \( R \subseteq V \), weighting function \( w : A = V \times Z \), positive integer \( k \)

PROPERTY: \( G \) has a subtree of weight \( \leq k \) containing the set of nodes in \( R \).

17. 3-dimensional matching

INPUT: Set \( U \subseteq \mathbb{T} \times \mathbb{T} \times \mathbb{T} \), where \( \mathbb{T} \) is a finite set

PROPERTY: There is a set \( U \subseteq \mathbb{T} \) such that \( |U| = |\mathbb{T}| \) and no two elements of \( U \) agree in any coordinate.

18. Knapsack

INPUT: \( \{a_1, a_2, \ldots, a_n\} \in Z^n \)

PROPERTY: There is a \( \{a_1, a_2, \ldots, a_n\} \in Z^n \) that satisfies a 0-1 solution.

19. Job sequence

INPUT: "Deadline vector" \( (D_1, D_2, \ldots, D_n) \in Z^n \)

PROPERTY: There is a permutation \( \pi \) of \( \{1, 2, \ldots, n\} \) such that

\[
(\sum_{j=1}^{n} D_{\pi(j)} + e_j) > 0 \quad \text{for} \quad e_j = \begin{cases} 1 & \text{if} \quad \pi_j \neq \pi_{j+1} \\ 0 & \text{else} \quad 0 \end{cases} \geq k.
\]
Figure 1 - Complete Problems

SATISFIABILITY

0-1 INTEGER PROGRAMMING

SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE

CHROMATIC NUMBER

EXACT COVER

CLIQUE COVER

DIRECTED HAMILTON CIRCUIT

UNDIRECTED HAMILTON CIRCUIT

FEEDBACK NODE SET

FEEDBACK ARC SET

SET PACKING

SET COVERING

3-DIMENSIONAL MATCHING

KNAPSACK

HITTING SET

STIERNER TREE

SEQUENCING

PARTITION

MAX CUT
20. PARTITION
   INPUT: \( (c_1, c_2, \ldots, c_n) \in \mathbb{Z}^n \)
   PROPERTY: There is a set \( I \subseteq \{1,2,\ldots,n\} \) such that
   \( \sum_{i \in I} c_i = \sum_{i \notin I} c_i \).

21. MAX CUT
   INPUT: graph \( G \), weighting function \( w: A \times 2 \), positive integer \( W \)
   PROPERTY: There is a set \( S \subseteq N \) such that
   \( \sum_{(u,v) \in A} w(u,v) \geq W \).
   \( \forall u \in S \)
   \( \forall v \notin S \)

It is clear that these problems (or, more precisely, their encodings into \( 1^* \)), are all in \( NP \). We proceed to give a series of explicit reductions, showing that \textsc{Satisfiability} is reducible to each of the problems listed. Figure 1 shows the structure of the set of reductions. Each line in the figure indicates a reduction of the upper problem to the lower one.

To exhibit a reduction of a set \( T \subseteq D \) to a set \( T' \subseteq D' \), we specify a function \( F: D \rightarrow D' \) which satisfies the conditions of \textsc{Lemma 3}. In each case, the reader should have little difficulty in verifying that \( F \) does satisfy these conditions.

\textsc{Satisfiability} = \textsc{0-1 Integer Programming}:
\[
c_{i,j} = \begin{cases} 1 & \text{if } x_i \in C_j \text{ for } i = 1,2,\ldots,p \\ -1 & \text{if } x_i \notin C_j \text{ for } i = 1,2,\ldots,n \\ 0 & \text{otherwise} \end{cases}
\]

\( b_i = 1 - \text{(the number of complemented variables in } C_j \text{)} \), for \( i = 1,2,\ldots,p \).

\textsc{Satisfiability} = \textsc{clique}:
\( N = \{<c,i> \mid c \text{ is a literal and occurs in } C_j \} \)
\( A = \{(\leq,1, \leq,2, \leq,3) \mid i \neq j \text{ and } c \notin \tilde{c} \} \)
\( k = p \), the number of clauses.

\textsc{clique} = \textsc{set packing}:
Assume \( N = \{1,2,\ldots,n\} \). The elements of the sets
\( S_1, S_2, \ldots, S_n \) are those two-element sets of nodes \( \{i,j\} \) not in \( A \).
\( S_1 = \{<i,j> \mid \{i,j\} \notin A, i = 1,2,\ldots,n \} \)
\( k = k' \).
REDUCIBILITY AMONG COM
SATISFIABILITY WITH
CHROMATIC NUMBER

\[ \text{Assume without loss of generality that } N = \{v_1, v_2, \ldots, v_n\} \]
\[ U = \{S_1, S_2, \ldots, S_r\} \]
\[ A = \{\{v_{i_1}, v_{i_2}\} \mid 1 \leq i_1 < i_2 \leq r\} \]
\[ k = r+1 \]

CHROMATIC NUMBER = \( k \)

\[ G' \text{ is the complete graph on } k \]
SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE

Assume without loss of generality that \( n \geq 4 \).

\[ N = \{u_1, u_2, \ldots, u_n\} \cup \{v_1, v_2, \ldots, v_n\} \cup \{w_1, w_2, \ldots, w_n\} \]

\[ A = \{(u_i, v_j) | i \neq j\} \cup \{(v_i, w_j) | i \neq j\} \cup \{(w_i, u_j) | i \neq j\} \]

\[ k = r + 1 \]

CHROMATIC NUMBER = CLIQUE COVER

\[ G' \text{ is the complement of } G \]

The set of elements is

\[ N \cup A \cup \{e_{ij} | 1 \leq i \leq k\} \]

The sets \( S_j \) are the following:

- For each \( f \), \( 1 \leq f \leq k \), and each \( u \in N \), \( (u, e_{ij}) \) is incident with \( u \) such that \( u \) is incident with \( e_{ij} \).

- For each \( v \in A \) and each pair \( f_1, f_2 \) such that \( 1 \leq f_1 \leq k \), \( 1 \leq f_2 \leq k \) and \( f_1 \neq f_2 \), \( (u, v, e_{ij}, f_1, f_2) \) is incident with \( u \) such that \( v \) is incident with \( e_{ij} \).

EXACT COVER = HITTING SET

The hitting set problem has sets \( U_i \) and elements \( e_{ij} \), such that \( e_{ij} \in U_i \) if \( e_{ij} \in S_j \).

EXACT COVER = STEINER TREE

The Steiner tree problem has a set \( \{S_j\} \) and elements \( e_{ij} \), such that \( e_{ij} \in U_i \) if \( e_{ij} \in S_j \).

EXACT COVER = 3-DIMENSIONAL MATCHING

Without loss of generality assume \( |S_j| \geq 2 \) for each \( j \).

Let \( T = \{e_{ij} | u_i \neq v_j\} \). Let \( \alpha \) be an arbitrary one-one function

RICHARD M. KARP

REDUCIBILITY AMONG COMBINATORIAL PROBLEMS
from \( u_i \) into \( T \). Let \( n \cdot r = T \) be a permutation such that, for each fixed \( j \), \( \langle i, j \rangle u_i \equiv S_j \) is a cycle of \( T \).

\[ U = \langle c, q, w, u_i, \langle i, j \rangle, \langle i, j \rangle \rangle \subseteq T \]

\[ U \subseteq \langle \delta, \sigma, \tau \rangle \]  

for all \( i, \delta \neq a(u_i) \).

**EXACT COVER = KNAPSACK**

Let \( d = |S_j| + 1 \). Let \( \epsilon_{j1} = \frac{1}{d} \) if \( u_i \in S_j \), \( \epsilon_{j0} = \frac{1}{d} \) if \( u_i \notin S_j \). Let

\[ r = |S_j|, \quad a_j = \sum \epsilon_{jj}d^{-1} \] and \( b = \frac{d!}{d-1} \).

**KNAPSACK = SEQUENCING**

\[ p = r, \quad T_L = F_L = a_1, \quad D_L = b. \]

**KNAPSACK = PARTITION**

\[ n = r + 2 \]

\[ c_i = a_i, \quad i = 1, 2, \ldots, r \]

\[ c_{r+1} = b + 1 \]

\[ c_{r+2} = (\frac{r}{d} a_j) + 1 - b \]

\[ i = 1 \]

**PARTITION = MAX CUT**

\[ B = \{1, 2, \ldots, n\} \]

\[ A = \{\{i, j\} | i \in N, j \in Y, i \neq j \} \]

\[ w(\{i, j\}) = c_i - c_j \]

\[ w = \frac{r^2}{d} - I \]

Some of the reductions exhibited here did not originate with the present writer. Cook (1970) showed that SATISFIABILITY = SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE. The reduction SATISFIABILITY = CLIQUE is implicit in Cook (1970), and was also known to Raymond Reiter. The reduction

**NODE COVER = FEEDBACK NODE SET**

was found by the Algorithms Seminar at the Cornell University Computer Science Department. The reduction

**NODE COVER = FEEDBACK ARC SET**

was found by Lawler and the writer, and Lawler discovered the reduction

**EXACT COVER = 3-DIMENSIONAL MATCHING**
REDUCIBILITY AMONG COMBINATORIAL PROBLEMS

The writer discovered that the exact cover problem was reducible to the directed traveling-salesman problem on a digraph in which the arcs have weight zero or one. Using refinements of the technique used in this construction, Tarjan showed that

\[ \text{EXACT COVER} = \text{DIRECTED HAMILTON CIRCUIT} \]

and, independently, Lawler showed that

\[ \text{DOE COVER} \subseteq \text{DIRECTED HAMILTON CIRCUIT} \]

The reduction

\[ \text{DIRECTED HAMILTON CIRCUIT} = \text{UNDIRECTED HAMILTON CIRCUIT} \]

was pointed out by Tarjan.

Below we list three problems in automata theory and language theory to which every complete problem is reducible. These problems are not known to be complete, since their membership in \( \text{NP} \) is presently in doubt. The reader unacquainted with automata and language theory can find the necessary definitions in Hopcroft and Ullman (1969).

EQUIVALENCE OF REGULAR EXPRESSIONS

INPUT: A pair of regular expressions over the alphabet \( \{0,1\} \)
PROPERTY: The two expressions define the same language.

EQUIVALENCE OF NONDETERMINISTIC FURTHER AUTOMATA

INPUT: A pair of nondeterministic finite automata with input alphabet \( \{0,1\} \)
PROPERTY: The two automata define the same language.

CONTEXT-SENSITIVE RECOGNITION

INPUT: A context-sensitive grammar \( \Gamma \) and a string \( x \)
PROPERTY: \( x \) is in the language generated by \( \Gamma \).

First we show that

SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE

\( \subseteq \) EQUIVALENCE OF REGULAR EXPRESSIONS

The reduction is made in two stages. In the first stage we construct a pair of regular expressions over an alphabet \( \Delta = \{0,1,2, \ldots, n_0, n_1, \ldots, n_k\} \). We then convert these regular expressions to regular expressions over \( \{0,1\} \).

The first regular expression is \( \Sigma^\Delta \) (more exactly, \( \Delta \) is written out as \( (a_{10}, a_{11}, \ldots, a_{12}, \ldots) \), and \( \Sigma^\Delta \) represents a copy of the expression for \( \Delta \) concatenated together). The second regular expression is

\[ \Sigma^\Delta \cup \bigcup_{i=1}^{k} (\Sigma^{a_{1i}} \Sigma^{a_{2i}} \Sigma^{a_{3i}} \cup \Sigma^{a_{4i}} \Sigma^{a_{5i}} \Sigma^{a_{6i}}) \cup \bigcup_{i=1}^{l} b(i) \]
where
\[ S(0) = \begin{cases} \Lambda \cup \alpha & \text{if } \emptyset = \emptyset_1 \\
\Lambda \cup \alpha \cup \alpha \cup \alpha \cup \alpha & \text{if } \emptyset = \emptyset_1 \cup \emptyset_2 \\
\Lambda \cup \Lambda \cup \Lambda \cup \Lambda \cup \alpha \cup \Lambda \cup \alpha \cup \alpha \cup \alpha \cup \alpha & \text{if } \emptyset = \emptyset_2 \\
\Lambda \cup \Lambda \cup \Lambda \cup \Lambda \cup \Lambda \cup \alpha \cup \alpha \cup \alpha \cup \alpha \cup \alpha & \text{if } \emptyset = \emptyset_1 \cup \emptyset_2 \cup \emptyset_3 \end{cases} \]

Now let \( \emptyset \) be the least positive integer \( \geq \log |\Lambda| \), and let \( \emptyset \) be a 1-1 function from \( \emptyset \) into \( \{0,1\}^\emptyset \). Replace each regular expression by a regular expression over \( \{0,1\} \), by making the substitution \( a \rightarrow \emptyset(a) \) for each occurrence of each element of \( \emptyset \).

**EQUIVALENCE OF REGULAR EXPRESSIONS = EQUIVALENCE OF NONDETERMINISTIC FINITE AUTOMATA**

There are standard polynomial-time algorithms [Salomaa (1969)] to convert a regular expression to an equivalent nondeterministic automaton. Finally, we show that, for any \( L \notin NP \),

\[ L \equiv \text{CONTEXT-SENSITIVE RECOGNITION} \]

Suppose \( L \) is recognized in time \( p(\cdot) \) by a nondeterministic Turing machine. Then the following language \( L \) over the alphabet \( \{0,1,\emptyset\} \) is accepted by a nondeterministic linear bounded automaton which simulates the Turing machine:

\[ L = \{ \emptyset p^l(\emptyset x) \mid x \notin L \} \]

Hence \( L \) is context-sensitive and has a context-sensitive grammar \( G \). Thus \( x \notin L \) iff

\[ \emptyset \emptyset^p(\emptyset^l(\emptyset x)) \emptyset^p(\emptyset^l(\emptyset x)) \]

is an acceptable input to CONTEXT-SENSITIVE RECOGNITION.

We conclude by listing the following important problems in NP which are not known to be complete.

**GRAPH ISOMORPHISIM**
**PROPERTY:** Graphs \( G \) and \( G' \) are isomorphic to \( G' \).

**NONPRIMES**
**PROPERTY:** Positive integer \( k \) is not prime.

**LINEAR INEQUALITIES**
**INPUT:** Integer matrix \( C \), integer vector \( d \)
**PROPERTY:** \( Cx \geq d \) has a rational solution.
Reducibility Among Combinatorial Problems

Appendix I

Notation and Terminology Used in Problem Specification

Propositional Calculus

\( x_1, x_2, \ldots, x_n \) propositional variables
\( \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n \) complements of propositional variables
\( \psi_1, \psi_2, \ldots, \psi_p \) clauses
\( C_1 \subseteq \{x_1, \overline{x}_2, \ldots, x_n, \overline{x}_1, x_2, \ldots, x_n\} \)

A clause contains no complementary pair of literals.

Scalars, Vectors, Matrices

- \( Z \) the positive integers
- \( Z^p \) the set of \( p \)-tuples of positive integers
- \( Z^n \) the set \( \{0, 1, \ldots, n-1\} \)
- \( K \) elements of \( Z \)
- \( \langle x, y \rangle \) the ordered pair \( \langle x, y \rangle \)
- \( C_{ij} \) \( (x, y) \) vectors with nonnegative integer components
- \( (C_{ij}) \) \( C \) matrices with integer components

Graphs and Digraphs

- \( G = (N, A) \) \( G' = (N', A') \) finite graphs
- \( N, N' \) sets of nodes
- \( A, A' \) sets of arcs
- \( s, t, u, v \) nodes
- \( E \) arcs
- \( (x, y) \) \( (x, y) \) is a set cut.
- \( w : \mathbb{R} \rightarrow \mathbb{R} \) weight functions

The weight of a subgraph is the sum of the weights of its arcs.

- \( H = (V, E) \) digraph
- \( V \) set of nodes
- \( E \) set of arcs
- \( e, e' \) arcs

Sets

- \( \emptyset \) the empty set
- \( |S| \) the number of elements in the finite set \( S \)
- \( \{x_1, \ldots, x_n\} \) finite families of finite sets