$$\mathcal{Z}' := \{\{\lambda_1, \lambda_2, y_1\}\{\overline{y_1}, \lambda_3, y_2\}, \{\overline{y_2}, \lambda_4, y_3\}, \ldots, \{\overline{y_{k-4}}, \lambda_{k-2}, y_{k-3}\}, \{\overline{y_{k-3}}, \lambda_{k-1}, \lambda_k\}\}.$$

If $Z = \{\lambda_1, \lambda_2\}$, we choose a new variable y_1 ($Y := \{y_1\}$) and set

$$\mathcal{Z}' := \{\{\lambda_1, \lambda_2, y_1\}, \{\lambda_1, \lambda_2, \overline{y_1}\}\}.$$

If $Z = \{\lambda_1\}$, we choose a set $Y = \{y_1, y_2\}$ of two new variables and set

$$\mathcal{Z}' := \{\{\lambda_1, y_1, y_2\}, \{\lambda_1, y_1, \overline{y_2}\}, \{\lambda_1, \overline{y_1}, y_2\}, \{\lambda_1, \overline{y_1}, \overline{y_2}\}\}.$$

Observe that in each case Z can be equivalently replaced by $\mathcal{Z}'$ in any instance of SATISFIABILITY. □

Theorem 15.22. (Cook [1971]) 3SAT *is NP-complete.*

Proof: As a restriction of SATISFIABILITY, 3SAT is certainly in *NP*. We now show that SATISFIABILITY polynomially transforms to 3SAT. Consider any collection $\mathcal{Z}$ of clauses $Z_1, \ldots, Z_m$. We shall construct a new collection $\mathcal{Z}'$ of clauses with three literals per clause such that $\mathcal{Z}$ is satisfiable if and only if $\mathcal{Z}'$ is satisfiable.

To do this, we replace each clause Z_i by an equivalent set of clauses, each with three literals. This is possible in linear time by Proposition 15.21. □

If we restrict each clause to consist of just two literals, the problem (called 2SAT) can be solved in linear time (Exercise 7).

15.5 Some Basic *NP*-Complete Problems

Karp discovered the wealth of consequences of Cook's work for combinatorial optimization problems. As a start, we consider the following problem:

STABLE SET	
Instance:	A graph G and an integer k.
Question:	Is there a stable set of k vertices?

Theorem 15.23. (Karp [1972]) STABLE SET *is NP-complete.*

Proof: Obviously, STABLE SET $\in$ *NP*. We show that SATISFIABILITY polynomially transforms to STABLE SET.

Let $\mathcal{Z}$ be a collection of clauses $Z_1, \ldots, Z_m$ with $Z_i = \{\lambda_{i1}, \ldots, \lambda_{ik_i}\}$ ($i = 1, \ldots, m$), where the λ_{ij} are literals over some set X of variables.

We shall construct a graph G such that G has a stable set of size m if and only if there is a truth assignment satisfying all m clauses.

For each clause Z_i, we introduce a clique of k_i vertices according to the literals in this clause. Vertices corresponding to different clauses are connected by an edge

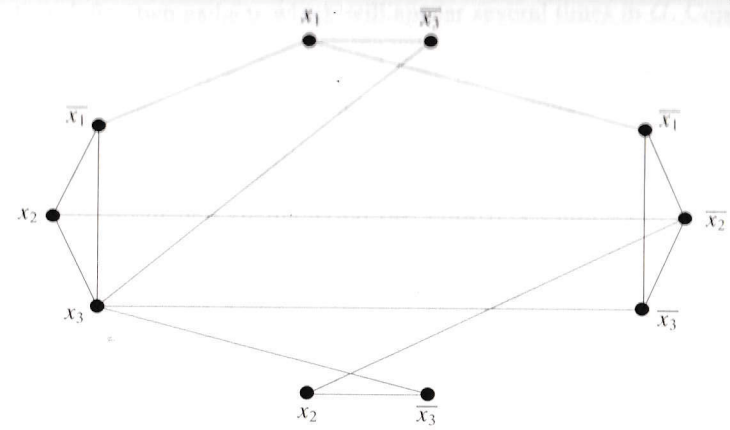

Fig. 15.1.

if and only if the literals contradict each other. Formally, let $V(G) := \{v_{ij} : 1 \leq i \leq m,\ 1 \leq j \leq k_i\}$ and

$$E(G) := \big\{\{v_{ij}, v_{kl}\} : (i = k \text{ and } j \neq l) \text{ or } (\lambda_{ij} = x \text{ and } \lambda_{kl} = \overline{x} \text{ for some } x \in X)\big\}.$$

See Figure 15.1 for an example ($m = 4$, $Z_1 = \{\overline{x_1}, x_2, x_3\}$, $Z_2 = \{x_1, \overline{x_3}\}$, $Z_3 = \{x_2, \overline{x_3}\}$ and $Z_4 = \{\overline{x_1}, \overline{x_2}, \overline{x_3}\}$).

Suppose G has a stable set of size m. Then its vertices specify pairwise compatible literals belonging to different clauses. Setting each of these literals to be *true* (and setting variables not occurring there arbitrarily) we obtain a truth assignment satisfying all m clauses.

Conversely, if some truth assignment satisfies all m clauses, then we choose a literal which is *true* out of each clause. The set of corresponding vertices then defines a stable set of size m in G. □

It is essential that k is part of the input: for each fixed k it can be decided in $O(n^k)$ time whether a given graph with n vertices has a stable set of size k (simply by testing all vertex sets with k elements). Two interesting related problems are the following:

VERTEX COVER	
Instance:	A graph G and an integer k.
Question:	Is there a vertex cover of cardinality k?

CLIQUE	
Instance:	A graph G and an integer k.
Question:	Has G a clique of cardinality k?

Corollary 15.24. (Karp [1972]) VERTEX COVER *and* CLIQUE *are NP-complete.*

Proof: By Proposition 2.2, STABLE SET polynomially transforms to both VERTEX COVER and CLIQUE. □

We now turn to the famous Hamiltonian circuit problem (already defined in Section 15.3).

Theorem 15.25. (Karp [1972]) HAMILTONIAN CIRCUIT *is NP-complete.*

Proof: Membership in *NP* is obvious. We prove that 3SAT polynomially transforms to HAMILTONIAN CIRCUIT. Given a collection $\mathcal{Z}$ of clauses $Z_1, \ldots, Z_m$ over $X = \{x_1, \ldots, x_n\}$, each clause containing three literals, we shall construct a graph G such that G is Hamiltonian iff $\mathcal{Z}$ is satisfiable.

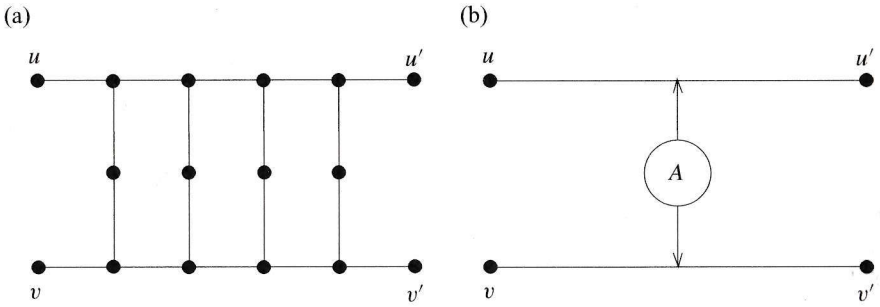

Fig. 15.2.

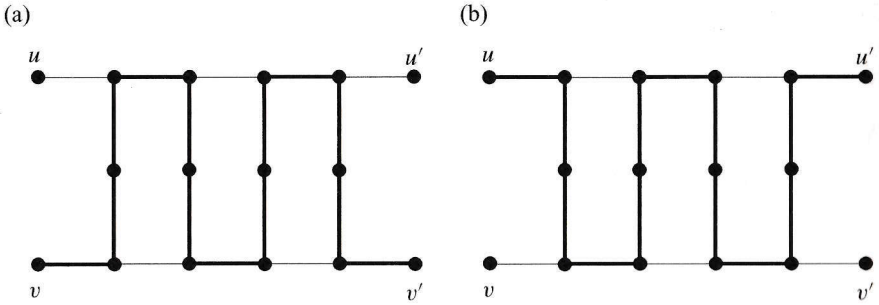

Fig. 15.3.

We first define two gadgets which will appear several times in G. Consider the graph shown in Figure 15.2(a), which we call A. We assume that it is a subgraph of G and no vertex of A except u, u', v, v' is incident to any other edge of G. Then any Hamiltonian circuit of G must traverse A in one of the ways shown in Figure 15.3(a) and (b). So we can replace A by two edges with the additional restriction that any Hamiltonian circuit of G must contain exactly one of them (Figure 15.2(b)).

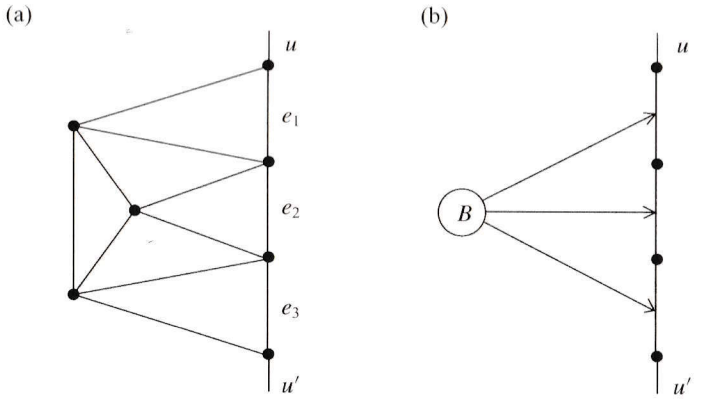

Fig. 15.4.

Now consider the graph B shown in Figure 15.4(a). We assume that it is a subgraph of G, and no vertex of B except u and u' is incident to any other edge of G. Then no Hamiltonian circuit of G traverses all of e_1, e_2, e_3. Moreover, one easily checks that for any $S \subset \{e_1, e_2, e_3\}$ there is a Hamiltonian path from u to u' in B that contains S but none of $\{e_1, e_2, e_3\} \setminus S$. We represent B by the symbol shown in Figure 15.4(b).

We are now able to construct G. For each clause, we introduce a copy of B, joined one after another. Between the first and the last copy of B, we insert two vertices for each variable, all joined one after another. We then double the edges between the two vertices of each variable x; these two edges will correspond to x and $\overline{x}$, respectively. The edges e_1, e_2, e_3 in each copy of B are now connected via a copy of A to the first, second, third literal of the corresponding clause. This construction is illustrated by Figure 15.5 with the example $\{\{x_1, \overline{x_2}, \overline{x_3}\}, \{\overline{x_1}, x_2, \overline{x_3}\}, \{\overline{x_1}, \overline{x_2}, x_3\}\}$. Note that an edge representing a literal can take part in more than one copy of A; these are then arranged in series.

Now we claim that G is Hamiltonian if and only if $\mathcal{Z}$ is satisfiable. Let C be a Hamiltonian circuit. We define a truth assignment by setting a literal *true* iff C contains the corresponding edge. By the properties of the gadgets A and B each clause contains a literal that is *true*.

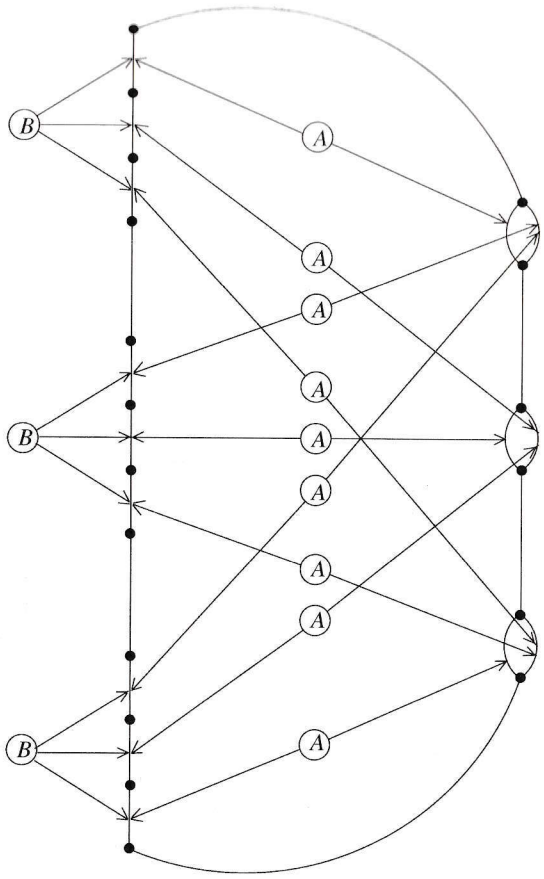

Fig. 15.5.

Conversely, any satisfying truth assignment defines a set of edges corresponding to literals that are *true*. Since each clause contains a literal that is *true* this set of edges can be completed to a tour in G. □

This proof is essentially due to Papadimitriou and Steiglitz [1982]. The problem of deciding whether a given graph contains a Hamiltonian path is also *NP*-complete (Exercise 14(a)). Moreover, one can easily transform the undirected versions to the directed Hamiltonian circuit or Hamiltonian path problem by replacing each undirected edge by a pair of oppositely directed edges. Thus the directed versions are also *NP*-complete.

There is another fundamental *NP*-complete problem:

3-DIMENSIONAL MATCHING (3DM)	
Instance:	Disjoint sets U, V, W of equal cardinality and $T \subseteq U \times V \times W$.
Question:	Is there a subset M of T with $\|M\| = \|U\|$ such that for distinct $(u, v, w), (u', v', w') \in M$ one has $u \neq u'$, $v \neq v'$ and $w \neq w'$?

Theorem 15.26. (Karp [1972]) 3DM *is NP-complete.*

Proof: Membership in *NP* is obvious. We shall polynomially transform SATISFIABILITY to 3DM. Given a collection $\mathcal{Z}$ of clauses $Z_1, \ldots, Z_m$ over $X = \{x_1, \ldots, x_n\}$, we construct an instance (U, V, W, T) of 3DM which is a yes-instance if and only if $\mathcal{Z}$ is satisfiable.

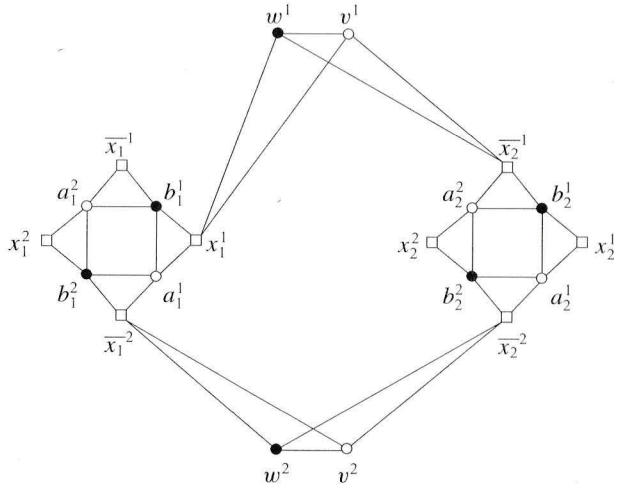

Fig. 15.6.

We define:

$$U := \{x_i^j, \overline{x}_i^j : i = 1, \ldots, n;\ j = 1, \ldots, m\}$$

$$V := \{a_i^j : i = 1, \ldots, n;\ j = 1, \ldots, m\} \cup \{v^j : j = 1, \ldots, m\} \cup \{c_k^j : k = 1, \ldots, n-1;\ j = 1, \ldots, m\}$$

$$W := \{b_i^j : i = 1, \ldots, n;\ j = 1, \ldots, m\} \cup \{w^j : j = 1, \ldots, m\} \cup \{d_k^j : k = 1, \ldots, n-1;\ j = 1, \ldots, m\}$$

$$T_1 := \{(x_i^j, a_i^j, b_i^j), (\overline{x}_i^j, a_i^{j+1}, b_i^j) : i = 1, \ldots, n;\ j = 1, \ldots, m\}, \text{ where } a_i^{m+1} := a_i^1$$

$$T_2 := \{(x_i^j, v^j, w^j) : i = 1, \ldots, n;\ j = 1, \ldots, m;\ x_i \in Z_j\}$$

$$\cup \{(\overline{x_i}^j, v^j, w^j) : i = 1, \ldots, n;\ j = 1, \ldots, m;\ \overline{x_i} \in Z_j\}$$

$$T_3 := \{(x_i^j, c_k^j, d_k^j), (\overline{x_i}^j, c_k^j, d_k^j) : i = 1, \ldots, n;\ j = 1, \ldots, m;\ k = 1, \ldots, n-1\}$$

$$T := T_1 \cup T_2 \cup T_3.$$

For an illustration of this construction, see Figure 15.6. Here $m = 2$, $Z_1 = \{x_1, \overline{x_2}\}$, $Z_2 = \{\overline{x_1}, \overline{x_2}\}$. Each triangle corresponds to an element of $T_1 \cup T_2$. The elements c_k^j, d_k^j and the triples in T_3 are not shown.

Suppose (U, V, W, T) is a yes-instance, so let $M \subseteq T$ be a solution. Since the a_i^j's and b_i^j appear only in elements T_1, for each i we have either $M \cap T_1 \supseteq \{(x_i^j, a_i^j, b_i^j) : j = 1, \ldots, m\}$ or $M \cap T_1 \supseteq \{(\overline{x_i}^j, a_i^{j+1}, b_i^j) : j = 1, \ldots, m\}$. In the first case we set x_i to *false*, in the second case to *true*.

Furthermore, for each clause Z_j we have $(\lambda^j, v^j, w^j) \in M$ for some literal $\lambda \in Z_j$. Since λ^j does not appear in any element of $M \cap T_1$ this literal is *true*; hence we have a satisfying truth assignment.

Conversely, a satisfying truth assignment suggests a set $M_1 \subseteq T_1$ of cardinality nm and a set $M_2 \subseteq T_2$ of cardinality m such that for distinct (u, v, w), $(u', v', w') \in M_1 \cup M_2$ we have $u \neq u'$, $v \neq v'$ and $w \neq w'$. It is easy to complete $M_1 \cup M_2$ by $(n-1)m$ elements of T_3 to a solution of the 3DM instance. □

A problem which looks simple but is not known to be solvable in polynomial time is the following:

SUBSET-SUM	
Instance:	Natural numbers $c_1, \ldots, c_n, K$.
Question:	Is there a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in S} c_j = K$?

Corollary 15.27. (Karp [1972]) SUBSET-SUM *is NP-complete.*

Proof: It is obvious that SUBSET-SUM is in *NP*. We prove that 3DM polynomially transforms to SUBSET-SUM. So let (U, V, W, T) be an instance of 3DM. W.l.o.g. let $U \cup V \cup W = \{u_1, \ldots, u_{3m}\}$. We write $S := \{\{a, b, c\} : (a, b, c) \in T\}$ and $S = \{s_1, \ldots, s_n\}$.
Define

$$c_j := \sum_{u_i \in s_j} (n+1)^{i-1} \qquad (j = 1, \ldots, n)$$

and

$$K := \sum_{i=1}^{3m} (n+1)^{i-1}.$$

Written in $(n+1)$-ary form, the number c_j can be regarded as the incidence vector of s_j $(j = 1, \ldots, n)$, and K consists of 1's only. Therefore each solution to the 3DM instance corresponds to a subset R of S such that $\sum_{s_j \in R} c_j = K$, and vice versa. Moreover, $\text{size}(c_j) \leq \text{size}(K) = O(m \log n)$, so the above is indeed a polynomial transformation. □

An important special case is the following problem:

PARTITION	
Instance:	Natural numbers $c_1, \ldots, c_n$.
Question:	Is there a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$?

Corollary 15.28. (Karp [1972]) PARTITION *is NP-complete.*

Proof: We show that SUBSET-SUM polynomially transforms to PARTITION. So let $c_1, \ldots, c_n, K$ be an instance of SUBSET-SUM. We add an element $c_{n+1} := \left|\sum_{i=1}^n c_i - 2K\right|$ (unless this number is zero) and have an instance $c_1, \ldots, c_{n+1}$ of PARTITION.

Case 1: $2K \leq \sum_{i=1}^n c_i$. Then for any $I \subseteq \{1, \ldots, n\}$ we have

$$\sum_{i \in I} c_i = K \quad \text{if and only if} \quad \sum_{i \in I \cup \{n+1\}} c_i = \sum_{i \in \{1, \ldots, n\} \setminus I} c_i.$$

Case 2: $2K > \sum_{i=1}^n c_i$. Then for any $I \subseteq \{1, \ldots, n\}$ we have

$$\sum_{i \in I} c_i = K \quad \text{if and only if} \quad \sum_{i \in I} c_i = \sum_{i \in \{1, \ldots, n+1\} \setminus I} c_i.$$

In both cases we have constructed a yes-instance of PARTITION if and only if the original instance of SUBSET-SUM is a yes-instance. □

We finally note:

Theorem 15.29. INTEGER LINEAR INEQUALITIES *is NP-complete.*

Proof: We already mentioned the membership in *NP* in Proposition 15.12. Any of the above problems can easily be formulated as an instance of INTEGER LINEAR INEQUALITIES. For example a PARTITION instance $c_1, \ldots, c_n$ is a yes-instance if and only if $\{x \in \mathbb{Z}^n : 0 \leq x \leq \mathbb{1},\ 2c^T x = c^T \mathbb{1}\}$ is nonempty. □

15.6 The Class *coNP*

The definition of *NP* is not symmetric with respect to yes-instances and no-instances. For example, it is an open question whether the following problem belongs to *NP*: given a graph G, is it true that G is not Hamiltonian? We introduce the following definitions:

Definition 15.30. *For a decision problem $\mathcal{P} = (X, Y)$ we define its* **complement** *to be the decision problem $(X, X \setminus Y)$. The class coNP consists of all problems whose complements are in NP. A decision problem $\mathcal{P} \in$ coNP is called* **coNP-complete** *if all other problems in coNP polynomially transform to $\mathcal{P}$.*