34.4-4

Show that the problem of determining whether a boolean formula is a tautology is complete for co-NP. (*Hint:* See Exercise 34.3-7.)

34.4-5

Show that the problem of determining the satisfiability of boolean formulas in disjunctive normal form is polynomial-time solvable.

34.4-6

Suppose that someone gives you a polynomial-time algorithm to decide formula satisfiability. Describe how to use this algorithm to find satisfying assignments in polynomial time.

34.4-7

Let 2-CNF-SAT be the set of satisfiable boolean formulas in CNF with exactly 2 literals per clause. Show that 2-CNF-SAT $\in$ P. Make your algorithm as efficient as possible. (*Hint:* Observe that $x \vee y$ is equivalent to $\neg x \rightarrow y$. Reduce 2-CNF-SAT to an efficiently solvable problem on a directed graph.)

34.5   NP-complete problems

NP-complete problems arise in diverse domains: boolean logic, graphs, arithmetic, network design, sets and partitions, storage and retrieval, sequencing and scheduling, mathematical programming, algebra and number theory, games and puzzles, automata and language theory, program optimization, biology, chemistry, physics, and more. In this section, we shall use the reduction methodology to provide NP-completeness proofs for a variety of problems drawn from graph theory and set partitioning.

Figure 34.13 outlines the structure of the NP-completeness proofs in this section and Section 34.4. We prove each language in the figure to be NP-complete by reduction from the language that points to it. At the root is CIRCUIT-SAT, which we proved NP-complete in Theorem 34.7.

34.5.1   The clique problem

A ***clique*** in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in E. In other words, a clique is a complete subgraph of G. The ***size*** of a clique is the number of vertices it contains. The ***clique problem*** is the optimization problem of finding a clique of maximum size in
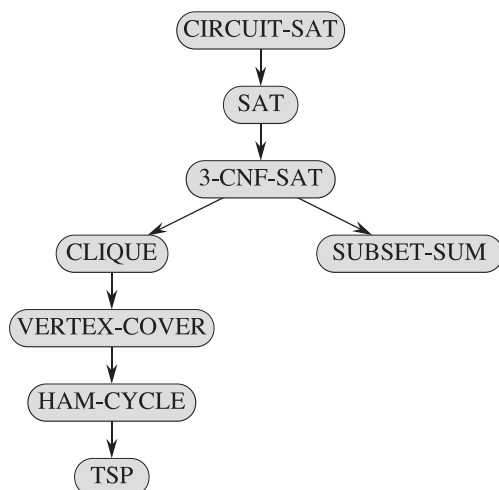
Figure 34.13 The structure of NP-completeness proofs in Sections 34.4 and 34.5. All proofs ultimately follow by reduction from the NP-completeness of CIRCUIT-SAT.

a graph. As a decision problem, we ask simply whether a clique of a given size k exists in the graph. The formal definition is

CLIQUE $= \{\langle G, k\rangle : G \text{ is a graph containing a clique of size } k\}$.

A naive algorithm for determining whether a graph $G = (V, E)$ with $|V|$ vertices has a clique of size k is to list all k-subsets of V, and check each one to see whether it forms a clique. The running time of this algorithm is $\Omega(k^2 \binom{|V|}{k})$, which is polynomial if k is a constant. In general, however, k could be near $|V|/2$, in which case the algorithm runs in superpolynomial time. Indeed, an efficient algorithm for the clique problem is unlikely to exist.

Theorem 34.11
The clique problem is NP-complete.

Proof To show that CLIQUE $\in$ NP, for a given graph $G = (V, E)$, we use the set $V' \subseteq V$ of vertices in the clique as a certificate for G. We can check whether V' is a clique in polynomial time by checking whether, for each pair $u, v \in V'$, the edge (u, v) belongs to E.

We next prove that 3-CNF-SAT $\leq_P$ CLIQUE, which shows that the clique problem is NP-hard. You might be surprised that we should be able to prove such a result, since on the surface logical formulas seem to have little to do with graphs.

The reduction algorithm begins with an instance of 3-CNF-SAT. Let $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ be a boolean formula in 3-CNF with k clauses. For $r =$
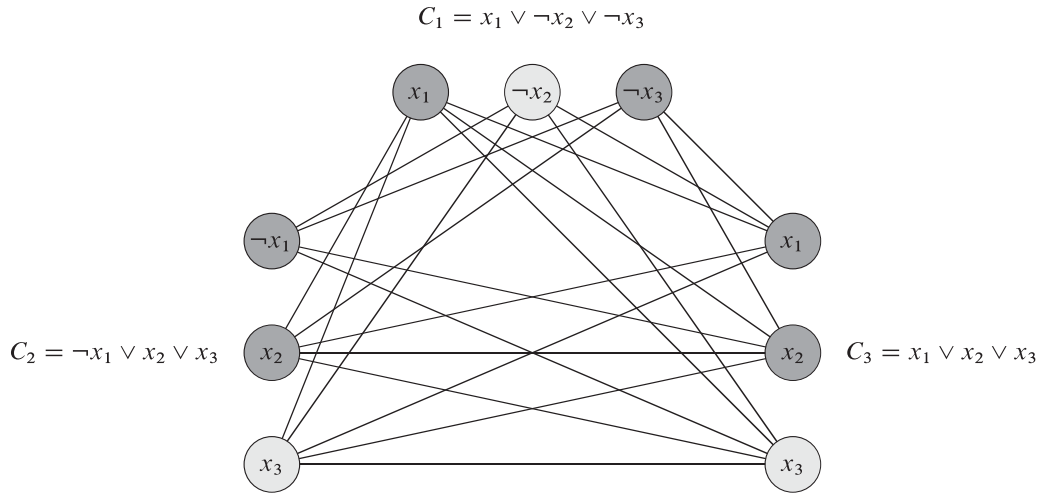
$$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$$

Figure 34.14 The graph G derived from the 3-CNF formula $\phi = C_1 \wedge C_2 \wedge C_3$, where $C_1 = (x_1 \vee \neg x_2 \vee \neg x_3)$, $C_2 = (\neg x_1 \vee x_2 \vee x_3)$, and $C_3 = (x_1 \vee x_2 \vee x_3)$, in reducing 3-CNF-SAT to CLIQUE. A satisfying assignment of the formula has $x_2 = 0$, $x_3 = 1$, and x_1 either 0 or 1. This assignment satisfies C_1 with $\neg x_2$, and it satisfies C_2 and C_3 with x_3, corresponding to the clique with lightly shaded vertices.

$1, 2, \ldots, k$, each clause C_r has exactly three distinct literals l_1^r, l_2^r, and l_3^r. We shall construct a graph G such that ϕ is satisfiable if and only if G has a clique of size k.

We construct the graph $G = (V, E)$ as follows. For each clause $C_r = (l_1^r \vee l_2^r \vee l_3^r)$ in ϕ, we place a triple of vertices v_1^r, v_2^r, and v_3^r into V. We put an edge between two vertices v_i^r and v_j^s if both of the following hold:

- v_i^r and v_j^s are in different triples, that is, $r \neq s$, and
- their corresponding literals are **consistent**, that is, l_i^r is not the negation of l_j^s.

We can easily build this graph from ϕ in polynomial time. As an example of this construction, if we have

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \,,$$

then G is the graph shown in Figure 34.14.

We must show that this transformation of ϕ into G is a reduction. First, suppose that ϕ has a satisfying assignment. Then each clause C_r contains at least one literal l_i^r that is assigned 1, and each such literal corresponds to a vertex v_i^r. Picking one such "true" literal from each clause yields a set V' of k vertices. We claim that V' is a clique. For any two vertices $v_i^r, v_j^s \in V'$, where $r \neq s$, both corresponding literals l_i^r and l_j^s map to 1 by the given satisfying assignment, and thus the literals

cannot be complements. Thus, by the construction of G, the edge (v_i^r, v_j^s) belongs to E.

Conversely, suppose that G has a clique V' of size k. No edges in G connect vertices in the same triple, and so V' contains exactly one vertex per triple. We can assign 1 to each literal l_i^r such that $v_i^r \in V'$ without fear of assigning 1 to both a literal and its complement, since G contains no edges between inconsistent literals. Each clause is satisfied, and so ϕ is satisfied. (Any variables that do not correspond to a vertex in the clique may be set arbitrarily.)   ∎

In the example of Figure 34.14, a satisfying assignment of ϕ has $x_2 = 0$ and $x_3 = 1$. A corresponding clique of size $k = 3$ consists of the vertices corresponding to $\neg x_2$ from the first clause, x_3 from the second clause, and x_3 from the third clause. Because the clique contains no vertices corresponding to either x_1 or $\neg x_1$, we can set x_1 to either 0 or 1 in this satisfying assignment.

Observe that in the proof of Theorem 34.11, we reduced an arbitrary instance of 3-CNF-SAT to an instance of CLIQUE with a particular structure. You might think that we have shown only that CLIQUE is NP-hard in graphs in which the vertices are restricted to occur in triples and in which there are no edges between vertices in the same triple. Indeed, we have shown that CLIQUE is NP-hard only in this restricted case, but this proof suffices to show that CLIQUE is NP-hard in general graphs. Why? If we had a polynomial-time algorithm that solved CLIQUE on general graphs, it would also solve CLIQUE on restricted graphs.

The opposite approach—reducing instances of 3-CNF-SAT with a special structure to general instances of CLIQUE—would not have sufficed, however. Why not? Perhaps the instances of 3-CNF-SAT that we chose to reduce from were "easy," and so we would not have reduced an NP-hard problem to CLIQUE.

Observe also that the reduction used the instance of 3-CNF-SAT, but not the solution. We would have erred if the polynomial-time reduction had relied on knowing whether the formula ϕ is satisfiable, since we do not know how to decide whether ϕ is satisfiable in polynomial time.

34.5.2 The vertex-cover problem

A **vertex cover** of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$ (or both). That is, each vertex "covers" its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E. The **size** of a vertex cover is the number of vertices in it. For example, the graph in Figure 34.15(b) has a vertex cover $\{w, z\}$ of size 2.

The **vertex-cover problem** is to find a vertex cover of minimum size in a given graph. Restating this optimization problem as a decision problem, we wish to
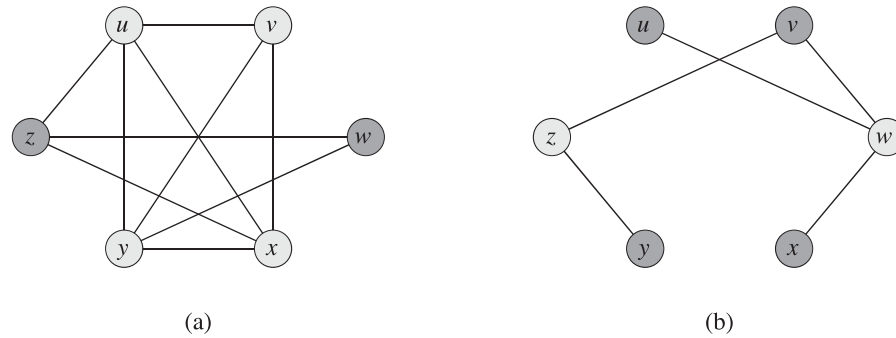
(a)   (b)

Figure 34.15   Reducing CLIQUE to VERTEX-COVER. **(a)** An undirected graph $G = (V, E)$ with clique $V' = \{u, v, x, y\}$. **(b)** The graph $\overline{G}$ produced by the reduction algorithm that has vertex cover $V - V' = \{w, z\}$.

determine whether a graph has a vertex cover of a given size k. As a language, we define

$$\text{VERTEX-COVER} = \{\langle G, k \rangle : \text{graph } G \text{ has a vertex cover of size } k\} \ .$$

The following theorem shows that this problem is NP-complete.

Theorem 34.12
The vertex-cover problem is NP-complete.

Proof   We first show that VERTEX-COVER $\in$ NP. Suppose we are given a graph $G = (V, E)$ and an integer k. The certificate we choose is the vertex cover $V' \subseteq V$ itself. The verification algorithm affirms that $|V'| = k$, and then it checks, for each edge $(u, v) \in E$, that $u \in V'$ or $v \in V'$. We can easily verify the certificate in polynomial time.

We prove that the vertex-cover problem is NP-hard by showing that CLIQUE $\leq_P$ VERTEX-COVER. This reduction relies on the notion of the "complement" of a graph. Given an undirected graph $G = (V, E)$, we define the ***complement*** of G as $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(u, v) : u, v \in V, u \neq v, \text{ and } (u, v) \notin E\}$. In other words, $\overline{G}$ is the graph containing exactly those edges that are not in G. Figure 34.15 shows a graph and its complement and illustrates the reduction from CLIQUE to VERTEX-COVER.

The reduction algorithm takes as input an instance $\langle G, k \rangle$ of the clique problem. It computes the complement $\overline{G}$, which we can easily do in polynomial time. The output of the reduction algorithm is the instance $\langle \overline{G}, |V| - k \rangle$ of the vertex-cover problem. To complete the proof, we show that this transformation is indeed a

reduction: the graph G has a clique of size k if and only if the graph $\overline{G}$ has a vertex cover of size $|V| - k$.

Suppose that G has a clique $V' \subseteq V$ with $|V'| = k$. We claim that $V - V'$ is a vertex cover in $\overline{G}$. Let (u, v) be any edge in $\overline{E}$. Then, $(u, v) \notin E$, which implies that at least one of u or v does not belong to V', since every pair of vertices in V' is connected by an edge of E. Equivalently, at least one of u or v is in $V - V'$, which means that edge (u, v) is covered by $V - V'$. Since (u, v) was chosen arbitrarily from $\overline{E}$, every edge of $\overline{E}$ is covered by a vertex in $V - V'$. Hence, the set $V - V'$, which has size $|V| - k$, forms a vertex cover for $\overline{G}$.

Conversely, suppose that $\overline{G}$ has a vertex cover $V' \subseteq V$, where $|V'| = |V| - k$. Then, for all $u, v \in V$, if $(u, v) \in \overline{E}$, then $u \in V'$ or $v \in V'$ or both. The contrapositive of this implication is that for all $u, v \in V$, if $u \notin V'$ and $v \notin V'$, then $(u, v) \in E$. In other words, $V - V'$ is a clique, and it has size $|V| - |V'| = k$. ∎

Since VERTEX-COVER is NP-complete, we don't expect to find a polynomial-time algorithm for finding a minimum-size vertex cover. Section 35.1 presents a polynomial-time "approximation algorithm," however, which produces "approximate" solutions for the vertex-cover problem. The size of a vertex cover produced by the algorithm is at most twice the minimum size of a vertex cover.

Thus, we shouldn't give up hope just because a problem is NP-complete. We may be able to design a polynomial-time approximation algorithm that obtains near-optimal solutions, even though finding an optimal solution is NP-complete. Chapter 35 gives several approximation algorithms for NP-complete problems.

34.5.3 The hamiltonian-cycle problem

We now return to the hamiltonian-cycle problem defined in Section 34.2.

Theorem 34.13
The hamiltonian cycle problem is NP-complete.

Proof We first show that HAM-CYCLE belongs to NP. Given a graph $G = (V, E)$, our certificate is the sequence of $|V|$ vertices that makes up the hamiltonian cycle. The verification algorithm checks that this sequence contains each vertex in V exactly once and that with the first vertex repeated at the end, it forms a cycle in G. That is, it checks that there is an edge between each pair of consecutive vertices and between the first and last vertices. We can verify the certificate in polynomial time.

We now prove that VERTEX-COVER $\leq_P$ HAM-CYCLE, which shows that HAM-CYCLE is NP-complete. Given an undirected graph $G = (V, E)$ and an