3

Proving NP-Completeness Results

If every NP-completeness proof had to be as complicated as that for SATISFIABILITY, it is doubtful that the class of known NP-complete problems would have grown as fast as it has. However, as discussed in Section 2.4, once we have proved a single problem NP-complete, the procedure for proving additional problems NP-complete is greatly simplified. Given a problem $\Pi \in \text{NP}$, all we need do is show that some already known NP-complete problem Π' can be transformed to Π. Thus, from now on, the process of devising an NP-completeness proof for a decision problem Π will consist of the following four steps:

(1) showing that Π is in NP,

(2) selecting a known NP-complete problem Π',

(3) constructing a transformation f from Π' to Π, and

(4) proving that f is a (polynomial) transformation.

In this chapter, we intend not only to acquaint readers with the end results of this process (the finished NP-completeness proofs) but also to prepare them for the task of constructing such proofs on their own. In Section 3.1 we present six problems that are commonly used as the "known NP-complete problem" in proofs of NP-completeness, and we prove that

these six are themselves NP-complete. In Section 3.2 we describe three general approaches for transforming one problem to another, and we demonstrate their use by proving a wide variety of problems NP-complete. A concluding section contains some suggested exercises.

3.1 Six Basic NP-Complete Problems

When seasoned practitioners are confronted with a problem Π to be proved NP-complete, they have the advantage of having a wealth of experience to draw upon. They may well have proved a similar problem Π' NP-complete in the past or have seen such a proof. This will suggest that they try to prove Π NP-complete by mimicking the NP-completeness proof for Π' or by transforming Π' itself to Π. In many cases this may lead rather easily to an NP-completeness proof for Π.

All too often, however, no known NP-complete problem similar to Π can be found (even using the extensive lists at the end of this book). In such cases the practitioner may have no direct intuition as to which of the hundreds of known NP-complete problems is best suited to serve as the basis for the desired proof. Nevertheless, experience can still narrow the choices down to a core of basic problems that have been useful in the past. Even though in theory *any* known NP-complete problem can serve just as well as any other for proving a new problem NP-complete, in practice certain problems do seem to be much better suited for this task. The following six problems are among those that have been used most frequently, and we suggest that these six can serve as a "basic core" of known NP-complete problems for the beginner.

3-SATISFIABILITY (3SAT)
INSTANCE: Collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses on a finite set U of variables such that $|c_i| = 3$ for $1 \leqslant i \leqslant m$.
QUESTION: Is there a truth assignment for U that satisfies all the clauses in C?

3-DIMENSIONAL MATCHING (3DM)
INSTANCE: A set $M \subseteq W \times X \times Y$, where W, X, and Y are disjoint sets having the same number q of elements.
QUESTION: Does M contain a *matching*, that is, a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' agree in any coordinate?

VERTEX COVER (VC)
INSTANCE: A graph $G = (V,E)$ and a positive integer $K \leqslant |V|$.
QUESTION: Is there a *vertex cover* of size K or less for G, that is, a subset $V' \subseteq V$ such that $|V'| \leqslant K$ and, for each edge $\{u,v\} \in E$, at least one of u and v belongs to V'?

CLIQUE
INSTANCE: A graph $G = (V,E)$ and a positive integer $J \leqslant |V|$.
QUESTION: Does G contain a *clique* of size J or more, that is, a subset $V' \subseteq V$ such that $|V'| \geqslant J$ and every two vertices in V' are joined by an edge in E?

HAMILTONIAN CIRCUIT (HC)
INSTANCE: A graph $G = (V,E)$.
QUESTION: Does G contain a Hamiltonian circuit, that is, an ordering $\langle v_1, v_2, \ldots, v_n \rangle$ of the vertices of G, where $n = |V|$, such that $\{v_n, v_1\} \in E$ and $\{v_i, v_{i+1}\} \in E$ for all i, $1 \leqslant i < n$?

PARTITION
INSTANCE: A finite set A and a "size" $s(a) \in Z^+$ for each $a \in A$.
QUESTION: Is there a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a) \; ?$$

One reason for the popularity of these six problems is that they all appeared in the original list of 21 NP-complete problems presented in [Karp, 1972]. We shall begin our illustration of the techniques for proving NP-completeness by proving that each of these six problems is NP-complete, noting, whenever appropriate, variants of these problems whose NP-completeness follows more or less directly from that of the basic problems.

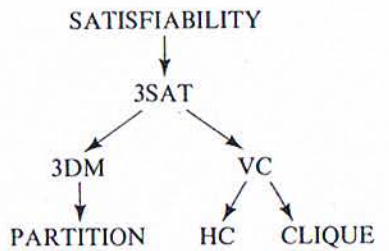

Figure 3.1 Diagram of the sequence of transformations used to prove that the six basic problems are NP-complete.

Our initial transformation will be from SATISFIABILITY, since it is the only "known" NP-complete problem we have so far. However, as we proceed through these six proofs, we will be enlarging our collection of known NP-complete problems, and all problems proved NP-complete before a problem Π will be available for use in proving that Π is NP-complete. The diagram of Figure 3.1 shows which problems we will be transforming to each of our six basic problems, where an arrow is drawn from one problem to another if the first is transformed to the second. This sequence of

transformations is not identical to that used by Karp, and, even when his sequence coincides with ours, we have sometimes modified or replaced the original transformation in order to illustrate certain general proof techniques.

3.1.1 3-SATISFIABILITY

The 3-SATISFIABILITY problem is just a restricted version of SATISFIABILITY in which all instances have exactly three literals per clause. Its simple structure makes it one of the most widely used problems for proving other NP-completeness results.

Theorem 3.1. 3-SATISFIABILITY is NP-complete.

Proof: It is easy to see that 3SAT $\in$ NP since a nondeterministic algorithm need only guess a truth assignment for the variables and check in polynomial time whether that truth setting satisfies all the given three-literal clauses.

We transform SAT to 3SAT. Let $U=\{u_1,u_2,\ldots,u_n\}$ be a set of variables and $C=\{c_1,c_2,\ldots,c_m\}$ be a set of clauses making up an arbitrary instance of SAT. We shall construct a collection C' of three-literal clauses on a set U' of variables such that C' is satisfiable if and only if C is satisfiable.

The construction of C' will merely replace each individual clause $c_j \in C$ by an "equivalent" collection C_j' of three-literal clauses, based on the original variables U and some additional variables U_j' whose use will be limited to clauses in C_j'. These will be combined by setting

$$U' = U \cup \left(\bigcup_{j=1}^{m} U_j' \right)$$

and

$$C' = \bigcup_{j=1}^{m} C_j'$$

Thus we only need to show how C_j' and U_j' can be constructed from c_j.

Let c_j be given by $\{z_1,z_2,\ldots,z_k\}$ where the z_i's are all literals derived from the variables in U. The way in which C_j' and U_j' are formed depends on the value of k.

Case 1. $k=1$. $U_j'=\{y_j^1,y_j^2\}$

$C_j'=\{\{z_1,y_j^1,y_j^2\},\{z_1,y_j^1,\bar{y}_j^2\},\{z_1,\bar{y}_j^1,y_j^2\},\{z_1,\bar{y}_j^1,\bar{y}_j^2\}\}$

Case 2. $k=2$. $U_j'=\{y_j^1\}$, $C_j'=\{\{z_1,z_2,y_j^1\},\{z_1,z_2,\bar{y}_j^1\}\}$

Case 3. $k=3$. $U_j'=\phi$, $C_j'=\{\{c_j\}\}$

Case 4. $k>3$. $U_j'=\{y_j^i: 1\leqslant i\leqslant k-3\}$

$C_j'=\{\{z_1,z_2,y_j^1\}\} \cup \{\{\bar{y}_j^i,z_{i+2},y_j^{i+1}\}: 1\leqslant i\leqslant k-4\}$

$\cup \{\{\bar{y}_j^{k-3},z_{k-1},z_k\}\}$

To prove that this is indeed a transformation, we must show that the set C' of clauses is satisfiable if and only if C is. Suppose first that $t: U\rightarrow\{T,F\}$ is a truth assignment satisfying C. We show that t can be extended to a truth assignment $t': U'\rightarrow\{T,F\}$ satisfying C'. Since the variables in $U'-U$ are partitioned into sets U_j' and since the variables in each U_j' occur only in clauses belonging to C_j', we need only show how t can be extended to the sets U_j' one at a time, and in each case we need only verify that all the clauses in the corresponding C_j' are satisfied. We can do this as follows: If U_j' was constructed under either Case 1 or Case 2, then the clauses in C_j' are already satisfied by t, so we can extend t arbitrarily to U_j', say by setting $t'(y)=T$ for all $y\in U_j'$. If U_j' was constructed under Case 3, then U_j' is empty and the single clause in C_j' is already satisfied by t. The only remaining case is Case 4, which corresponds to a clause $\{z_1,z_2,\ldots,z_k\}$ from C with $k>3$. Since t is a satisfying truth assignment for C, there must be a least integer l such that the literal z_l is set true under t. If l is either 1 or 2, then we set $t'(y_j^i)=F$ for $1\leqslant i\leqslant k-3$. If l is either $k-1$ or k, then we set $t'(y_j^i)=T$ for $1\leqslant i\leqslant k-3$. Otherwise we set $t'(y_j^i)=T$ for $1\leqslant i\leqslant l-2$ and $t'(y_j^i)=F$ for $l-1\leqslant i\leqslant k-3$. It is easy to verify that these choices will insure that all the clauses in C_j' will be satisfied, so all the clauses in C' will be satisfied by t'. Conversely, if t' is a satisfying truth assignment for C', it is easy to verify that the restriction of t' to the variables in U must be a satisfying truth assignment for C. Thus C' is satisfiable if and only if C is.

To see that this transformation can be performed in polynomial time, it suffices to observe that the number of three-literal clauses in C' is bounded by a polynomial in mn. Hence the size of the 3SAT instance is bounded above by a polynomial function of the size of the SAT instance, and, since all details of the construction itself are straightforward, the reader should have no difficulty verifying that this is a polynomial transformation. ■

The restricted structure of 3SAT makes it much more useful than SAT for proving NP-completeness results. Any proof based on SAT (except for the one we have just given) can be converted immediately to one based on 3SAT, without even changing the transformation. In fact, the normalization to clauses having the same size often can simplify the transformations we need to construct and thus make them easier to find. Furthermore, the very smallness of these clauses permits us to use transformations that would not work for instances containing larger clauses. This suggests that it would be still more convenient if we could show that the analogous 2-SATISFIABILITY problem, in which each clause has exactly *two* literals, were NP-complete. However, 2SAT can be solved by "resolution" tech-

niques in time bounded by a polynomial in the product of the number of clauses and the number of variables in the given instance [Cook, 1971] (see also [Even, Itai, and Shamir, 1976]), and hence is in P.

3.1.2 3-DIMENSIONAL MATCHING

The 3-DIMENSIONAL MATCHING problem is a generalization of the classical "marriage problem": Given n unmarried men and n unmarried women, along with a list of all male-female pairs who would be willing to marry one another, is it possible to arrange n marriages so that polygamy is avoided and everyone receives an acceptable spouse? Analogously, in the 3-DIMENSIONAL MATCHING problem, the sets W, X, and Y correspond to *three* different sexes, and each triple in M corresponds to a *3-way marriage* that would be acceptable to all three participants. Traditionalists will be pleased to note that, whereas 3DM is NP-complete, the ordinary marriage problem can be solved in polynomial time (for example, see [Hopcroft and Karp, 1973]).

Theorem 3.2 3-DIMENSIONAL MATCHING is NP-complete.

Proof: It is easy to see that 3DM $\in$ NP, since a nondeterministic algorithm need only guess a subset of $q=|W|=|X|=|Y|$ triples from M and check in polynomial time that no two of the guessed triples agree in any coordinate.

We will transform 3SAT to 3DM. Let $U=\{u_1,u_2,\ldots,u_n\}$ be the set of variables and $C=\{c_1,c_2,\ldots,c_m\}$ be the set of clauses in an arbitrary instance of 3SAT. We must construct disjoint sets W, X, and Y, with $|W|=|X|=|Y|$, and a set $M\subseteq W\times X\times Y$ such that M contains a matching if and only if C is satisfiable.

The set M of ordered triples will be partitioned into three separate classes, grouped according to their intended function: "truth-setting and fan-out," "satisfaction testing," or "garbage collection."

Each truth-setting and fan-out component corresponds to a single variable $u\in U$, and its structure depends on the total number m of clauses in C. This structure is illustrated for the case of $m=4$ in Figure 3.2. In general, the truth-setting and fan-out component for a variable u_i involves "internal" elements $a_i[j]\in X$ and $b_i[j]\in Y$, $1\leqslant j\leqslant m$, which will not occur in any triples outside of this component, and "external" elements $u_i[j],\bar{u}_i[j]\in W$, $1\leqslant j\leqslant m$, which *will* occur in other triples. The triples making up this component can be divided into two sets:

$$T_i^t = \{(\bar{u}_i[j],a_i[j],b_i[j]): 1\leqslant j\leqslant m\}$$
$$T_i^f = \{(u_i[j],a_i[j+1],b_i[j]): 1\leqslant j<m\}\cup\{(u_i[m],a_i[1],b_i[m])\}$$

Since none of the internal elements $\{a_i[j],b_i[j]: 1\leqslant j\leqslant m\}$ will appear in any

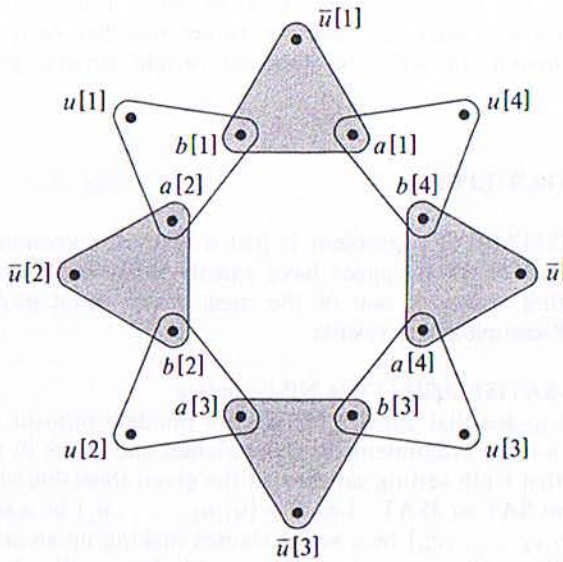

Figure 3.2 Truth setting component T_i when $m=4$ (subscripts have been deleted for simplicity). Either all the sets of T_i^t (the shaded sets) or all the sets of T_i^f (the unshaded sets) must be chosen, leaving uncovered all the $u_i[j]$ or all the $\bar{u}_i[j]$, respectively.

triples outside of $T_i=T_i^t\cup T_i^f$, it is easy to see that any matching M' will have to include exactly m triples from T_i, either all triples in T_i^t or all triples in T_i^f. Hence we can think of the component T_i as forcing a matching to make a choice between setting u_i true and setting u_i false. Thus, in general, a matching $M'\subseteq M$ specifies a truth assignment for U, with the variable u_i being set true if and only if $M'\cap T_i=T_i^t$.

Each satisfaction testing component in M corresponds to a single clause $c_j\in C$. It involves only two "internal" elements, $s_1[j]\in X$ and $s_2[j]\in Y$, and external elements from $\{u_i[j],\bar{u}_i[j]: 1\leqslant i\leqslant n\}$, determined by which literals occur in clause c_j. The set of triples making up this component is defined as follows:

$$C_j = \{(u_i[j],s_1[j],s_2[j]): u_i\in c_j\}\cup\{(\bar{u}_i[j],s_1[j],s_2[j]): \bar{u}_i\in c_j\}$$

Thus any matching $M'\subseteq M$ will have to contain exactly one triple from C_j. This can only be done, however, if some $u_i[j]$ (or $\bar{u}_i[j]$) for a literal $u_i\in c_j$ ($\bar{u}_i\in c_j$) does not occur in the triples in $T_i\cap M'$, which will be the case if and only if the truth setting determined by M' satisfies clause c_j.

The construction is completed by means of one large "garbage collection" component G, involving internal elements $g_1[k] \in X$ and $g_2[k] \in Y$, $1 \leqslant k \leqslant m(n-1)$, and external elements of the form $u_i[j]$ and $\bar{u}_i[j]$ from W. It consists of the following set of triples:

$$G = \{(u_i[j], g_1[k], g_2[k]), (\bar{u}_i[j], g_1[k], g_2[k]):$$
$$1 \leqslant k \leqslant m(n-1), 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$$

Thus each pair $g_1[k], g_2[k]$ must be matched with a unique $u_i[j]$ or $\bar{u}_i[j]$ that does not occur in any triples of $M'-G$. There are exactly $m(n-1)$ such "uncovered" external elements, and the structure of G insures that they can always be covered by choosing $M' \cap G$ appropriately. Thus G merely guarantees that, whenever a subset of $M-G$ satisfies all the constraints imposed by the truth-setting and fan-out components, then that subset can be extended to a matching for M.

To summarize, we set

$$W = \{u_i[j], \bar{u}_i[j]: 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$$

$$X = A \cup S_1 \cup G_1$$

where

$$A = \{a_i[j]: 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$$
$$S_1 = \{s_1[j]: 1 \leqslant j \leqslant m\}$$
$$G_1 = \{g_1[j]: 1 \leqslant j \leqslant m(n-1)\}$$

$$Y = B \cup S_2 \cup G_2$$

where

$$B = \{b_i[j]: 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$$
$$S_2 = \{s_2[j]: 1 \leqslant j \leqslant m\}$$
$$G_2 = \{g_2[j]: 1 \leqslant j \leqslant m(n-1)\}$$

and

$$M = \left(\bigcup_{i=1}^{n} T_i\right) \cup \left(\bigcup_{j=1}^{m} C_j\right) \cup G$$

Notice that every triple in M is an element of $W \times X \times Y$ as required. Furthermore, since M contains only

$$2mn + 3m + 2m^2n(n-1)$$

triples and since its definition in terms of the given 3SAT instance is quite direct, it is easy to see that M can be constructed in polynomial time.

From the comments made during the description of M, it follows immediately that M cannot contain a matching unless C is satisfiable. We now must show that the existence of a satisfying truth assignment for C implies that M contains a matching.

Let $t: U \rightarrow \{T,F\}$ be any satisfying truth assignment for C. We construct a matching $M' \subseteq M$ as follows: For each clause $c_j \in C$, let $z_j \in \{u_i, \bar{u}_i: 1 \leqslant i \leqslant n\} \cap c_j$ be a literal that is set true by t (one must exist since t satisfies c_j). We then set

$$M' = \left(\bigcup_{t(u_i)=T} T_i^t\right) \cup \left(\bigcup_{t(u_i)=F} T_i^f\right) \cup \left(\bigcup_{j=1}^{m} \{(z_j[j], s_1[j], s_2[j])\}\right) \cup G'$$

where G' is an appropriately chosen subcollection of G that includes all the $g_1[k], g_2[k]$, and remaining $u_i[j]$ and $\bar{u}_i[j]$. It is easy to verify that such a G' can always be chosen and that the resulting set M' is a matching. ■

In proving NP-completeness results, the following slightly simpler and more general version of 3DM can often be used in its place:

EXACT COVER BY 3-SETS (X3C)
INSTANCE: A finite set X with $|X| = 3q$ and a collection C of 3-element subsets of X.
QUESTION: Does C contain an *exact cover* for X, that is, a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

Note that every instance of 3DM can be viewed as an instance of X3C, simply by regarding it as an unordered subset of $W \cup X \cup Y$, and the matchings for that 3DM instance will be in one-to-one correspondence with the exact covers for the X3C instance. Thus 3DM is just a restricted version of X3C, and the NP-completeness of X3C follows by a trivial transformation from 3DM.

3.1.3 VERTEX COVER and CLIQUE

Despite the fact that VERTEX COVER and CLIQUE are independently useful for proving NP-completeness results, they are really just different ways of looking at the same problem. To see this, it is convenient to consider them in conjunction with a third problem, called INDEPENDENT SET.

An *independent set* in a graph $G = (V,E)$ is a subset $V' \subseteq V$ such that, for all $u,v \in V'$, the edge $\{u,v\}$ is *not* in E. The INDEPENDENT SET problem asks, for a given graph $G = (V,E)$ and a positive integer $J \leqslant |V|$, whether G contains an independent set V' having $|V'| \geqslant J$. The following relationships between independent sets, cliques, and vertex covers are easy to verify.

Lemma 3.1 For any graph $G=(V,E)$ and subset $V'\subseteq V$, the following statements are equivalent:

(a) V' is a vertex cover for G.
(b) $V-V'$ is an independent set for G.
(c) $V-V'$ is a clique in the *complement* G^c of G, where $G^c=(V,E^c)$ with $E^c=\{\{u,v\}: u,v\in V \text{ and } \{u,v\}\notin E\}$.

Thus we see that, in a rather strong sense, these three problems might be regarded simply as "different versions" of one another. Furthermore, the relationships displayed in the lemma make it a trivial matter to transform any one of the problems to either of the others.

For example, to transform VERTEX COVER to CLIQUE, let $G=(V,E)$ and $K\leqslant|V|$ constitute any instance of VC. The corresponding instance of CLIQUE is provided simply by the graph G^c and the integer $J=|V|-K$.

This implies that the NP-completeness of all three problems will follow as an immediate consequence of proving that any one of them is NP-complete. We choose to prove this for VERTEX COVER.

Theorem 3.3 VERTEX COVER is NP-complete.

Proof: It is easy to see that VC $\in$ NP since a nondeterministic algorithm need only guess a subset of vertices and check in polynomial time whether that subset contains at least one endpoint of every edge and has the appropriate size.

We transform 3SAT to VERTEX COVER. Let $U=\{u_1,u_2,\ldots,u_n\}$ and $C=\{c_1,c_2,\ldots,c_m\}$ be any instance of 3SAT. We must construct a graph $G=(V,E)$ and a positive integer $K\leqslant|V|$ such that G has a vertex cover of size K or less if and only if C is satisfiable.

As in the previous proof, the construction will be made up of several components. In this case, however, we will have only truth-setting components and satisfaction testing components, augmented by some additional edges for communicating between the various components.

For each variable $u_i\in U$, there is a truth-setting component $T_i=(V_i,E_i)$, with $V_i=\{u_i,\bar{u}_i\}$ and $E_i=\{\{u_i,\bar{u}_i\}\}$, that is, two vertices joined by a single edge. Note that any vertex cover will have to contain at least one of u_i and $\bar{u}_i$ in order to cover the single edge in E_i.

For each clause $c_j\in C$, there is a satisfaction testing component $S_j=(V_j',E_j')$, consisting of three vertices and three edges joining them to form a triangle:

$$V_j'=\{a_1[j],a_2[j],a_3[j]\}$$
$$E_j'=\{\{a_1[j],a_2[j]\},\{a_1[j],a_3[j]\},\{a_2[j],a_3[j]\}\}$$

Note that any vertex cover will have to contain at least two vertices from V_j' in order to cover the edges in E_j'.

The only part of the construction that depends on which literals occur in which clauses is the collection of communication edges. These are best viewed from the vantage point of the satisfaction testing components. For each clause $c_j\in C$, let the three literals in c_j be denoted by x_j, y_j, and z_j. Then the communication edges emanating from S_j are given by:

$$E_j''=\{\{a_1[j],x_j\},\{a_2[j],y_j\},\{a_3[j],z_j\}\}$$

The construction of our instance of VC is completed by setting $K=n+2m$ and $G=(V,E)$, where

$$V=(\bigcup_{i=1}^{n}V_i)\cup(\bigcup_{j=1}^{m}V_j')$$

and

$$E=(\bigcup_{i=1}^{n}E_i)\cup(\bigcup_{j=1}^{m}E_j')\cup(\bigcup_{j=1}^{m}E_j'')$$

Figure 3.3 shows an example of the graph obtained when $U=\{u_1,u_2,u_3,u_4\}$ and $C=\{\{u_1,\bar{u}_3,\bar{u}_4\},\{\bar{u}_1,u_2,\bar{u}_4\}\}$.

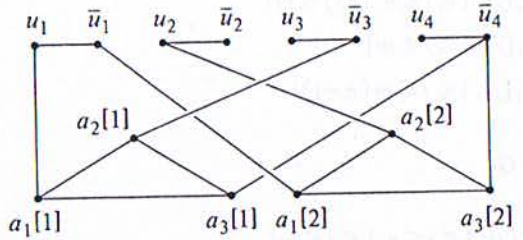

Figure 3.3 VERTEX COVER instance resulting from 3SAT instance in which $U=\{u_1,u_2,u_3,u_4\}$, $C=\{\{u_1,\bar{u}_3,\bar{u}_4\},\{\bar{u}_1,u_2,\bar{u}_4\}\}$. Here $K=n+2m=8$.

It is easy to see how the construction can be accomplished in polynomial time. All that remains to be shown is that C is satisfiable if and only if G has a vertex cover of size K or less.

First, suppose that $V'\subseteq V$ is a vertex cover for G with $|V'|\leqslant K$. By our previous remarks, V' must contain at least one vertex from each T_i and at least two vertices from each S_j. Since this gives a total of at least $n+2m=K$ vertices, V' must in fact contain *exactly* one vertex from each T_i and *exactly* two vertices from each S_j. Thus we can use the way in which V' intersects each truth-setting component to obtain a truth assignment $t: U\rightarrow\{T,F\}$. We merely set $t(u_i)=T$ if $u_i\in V'$ and $t(u_i)=F$ if

$\bar{u}_i \in V'$. To see that this truth assignment satisfies each of the clauses $c_j \in C$, consider the three edges in E_j''. Only two of those edges can be covered by vertices from $V_j' \cap V'$, so one of them must be covered by a vertex from some V_i that belongs to V'. But that implies that the corresponding literal, either u_i or $\bar{u}_i$, from clause c_j is true under the truth assignment t, and hence clause c_j is satisfied by t. Because this holds for every $c_j \in C$, it follows that t is a satisfying truth assignment for C.

Conversely, suppose that $t: U \rightarrow \{T,F\}$ is a satisfying truth assignment for C. The corresponding vertex cover V' includes one vertex from each T_i and two vertices from each S_j. The vertex from T_i in V' is u_i if $t(u_i) = T$ and is $\bar{u}_i$ if $t(u_i) = F$. This ensures that at least one of the three edges from each set E_j'' is covered, because t satisfies each clause c_j. Therefore we need only include in V' the endpoints from S_j of the other two edges in E_j'' (which may or may not also be covered by vertices from truth-setting components), and this gives the desired vertex cover. ■

SEMINARIO

3.1.4 HAMILTONIAN CIRCUIT

In Chapter 2, we saw that the HAMILTONIAN CIRCUIT problem can be transformed to the TRAVELING SALESMAN decision problem, so the NP-completeness of the latter problem will follow immediately once HC has been proved NP-complete. At the end of the proof we note several variants of HC whose NP-completeness also follows more or less directly from that of HC.

For convenience in what follows, whenever $\langle v_1, v_2, \ldots, v_n \rangle$ is a Hamiltonian circuit, we shall refer to $\{v_i, v_{i+1}\}$, $1 \leqslant i < n$, and $\{v_n, v_1\}$ as the edges "in" that circuit. Our transformation is a combination of two transformations from [Karp, 1972], also described in [Liu and Geldmacher, 1978].

Theorem 3.4 HAMILTONIAN CIRCUIT is NP-complete

Proof: It is easy to see that HC $\in$ NP, because a nondeterministic algorithm need only guess an ordering of the vertices and check in polynomial time that all the required edges belong to the edge set of the given graph.

We transform VERTEX COVER to HC. Let an arbitrary instance of VC be given by the graph $G = (V,E)$ and the positive integer $K \leqslant |V|$. We must construct a graph $G' = (V',E')$ such that G' has a Hamiltonian circuit if and only if G has a vertex cover of size K or less.

Once more our construction can be viewed in terms of components connected together by communication links. First, the graph G' has K "selector" vertices $a_1, a_2, \ldots, a_K$, which will be used to select K vertices from the vertex set V for G. Second, for each edge in E, G' contains a "cover-testing" component that will be used to ensure that at least one endpoint of that edge is among the selected K vertices. The component for

$e = \{u,v\} \in E$ is illustrated in Figure 3.4. It has 12 vertices,

$$V_e' = \{(u,e,i),(v,e,i): 1 \leqslant i \leqslant 6\}$$

and 14 edges,

$$E_e' = \{\{(u,e,i),(u,e,i+1)\},\{(v,e,i),(v,e,i+1)\}: 1 \leqslant i \leqslant 5\}$$
$$\cup \{\{(u,e,3),(v,e,1)\},\{(v,e,3),(u,e,1)\}\}$$
$$\cup \{\{(u,e,6),(v,e,4)\},\{(v,e,6),(u,e,4)\}\}$$

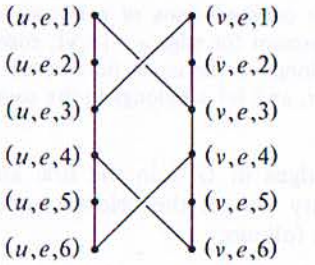

Figure 3.4 Cover-testing component for edge $e = \{u,v\}$ used in transforming VERTEX COVER to HAMILTONIAN CIRCUIT.

In the completed construction, the only vertices from this cover-testing component that will be involved in any additional edges are $(u,e,1)$, $(v,e,1)$, $(u,e,6)$, and $(v,e,6)$. This will imply, as the reader may readily verify, that any Hamiltonian circuit of G' will have to meet the edges in E_e' in exactly one of the three configurations shown in Figure 3.5. Thus, for example, if the circuit "enters" this component at $(u,e,1)$, it will have to "exit" at $(u,e,6)$ and visit either all 12 vertices in the component or just the 6 vertices (u,e,i), $1 \leqslant i \leqslant 6$.

Additional edges in our overall construction will serve to join pairs of cover-testing components or to join a cover-testing component to a selector vertex. For each vertex $v \in V$, let the edges incident on v be ordered (arbitrarily) as $e_{v[1]}, e_{v[2]}, \ldots, e_{v[deg(v)]}$, where $deg(v)$ denotes the *degree* of v in G, that is, the number of edges incident on v. All the cover-testing components corresponding to these edges (having v as endpoint) are joined together by the following connecting edges:

$$E_v' = \{\{(v,e_{v[i]},6),(v,e_{v[i+1]},1)\}: 1 \leqslant i < deg(v)\}$$

As shown in Figure 3.6, this creates a single path in G' that includes exactly those vertices (x,y,z) having $x = v$.

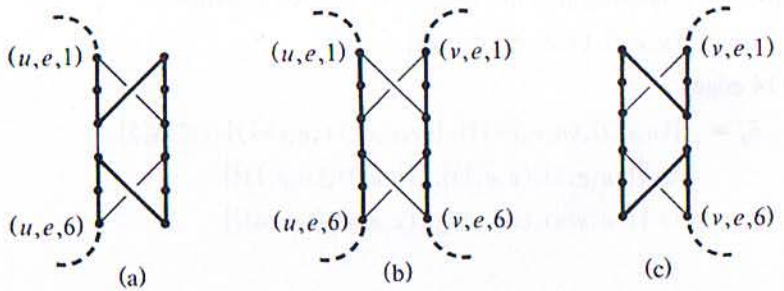

Figure 3.5 The three possible configurations of a Hamiltonian circuit within the cover-testing component for edge $e=\{u,v\}$, corresponding to the cases in which (a) u belongs to the cover but v does not, (b) both u and v belong to the cover, and (c) v belongs to the cover but u does not.

The final connecting edges in G' join the first and last vertices from each of these paths to every one of the selector vertices $a_1, a_2, \ldots, a_K$. These edges are specified as follows:

$$E'' = \{\{a_i,(v,e_{v[1]},1)\},\{a_i,(v,e_{v[deg(v)]},6)\}: 1 \leqslant i \leqslant K,\ v \in V\}$$

The completed graph $G' = (V',E')$ has

$$V' = \{a_i: 1 \leqslant i \leqslant K\} \cup \left(\bigcup_{e \in E} V'_e\right)$$

and

$$E' = \left(\bigcup_{e \in E} E'_e\right) \cup \left(\bigcup_{v \in V} E'_v\right) \cup E''$$

It is not hard to see that G' can be constructed from G and K in polynomial time.

We claim that G' has a Hamiltonian circuit if and only if G has a vertex cover of size K or less. Suppose $<v_1, v_2, \ldots, v_n>$, where $n = |V'|$, is a Hamiltonian circuit for G'. Consider any portion of this circuit that begins at a vertex in the set $\{a_1, a_2, \ldots, a_K\}$, ends at a vertex in $\{a_1, a_2, \ldots, a_K\}$, and that encounters no such vertex internally. Because of the previously mentioned restrictions on the way in which a Hamiltonian circuit can pass through a cover-testing component, this portion of the circuit must pass through a set of cover-testing components corresponding to exactly those edges from E that are incident on some one particular vertex $v \in V$. Each of the cover-testing components is traversed in one of the modes (a), (b), or (c) of Figure 3.5, and no vertex from any other cover-testing component is encountered. Thus the K vertices from $\{a_1, a_2, \ldots, a_K\}$ divide the Hamiltonian circuit into K paths, each path

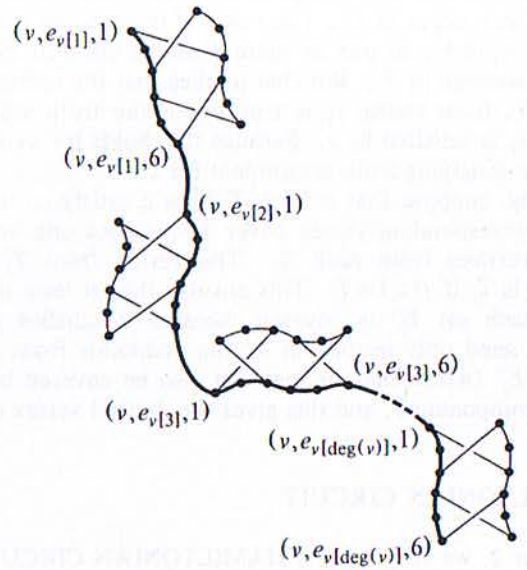

Figure 3.6 Path joining all the cover-testing components for edges from E having vertex v as an endpoint.

corresponding to a distinct vertex $v \in V$. Since the Hamiltonian circuit must include all vertices from every one of the cover-testing components, and since vertices from the cover-testing component for edge $e \in E$ can be traversed only by a path corresponding to an endpoint of e, every edge in E must have at least one endpoint among those K selected vertices. Therefore, this set of K vertices forms the desired vertex cover for G.

Conversely, suppose $V^* \subseteq V$ is a vertex cover for G with $|V^*| \leqslant K$. We can assume that $|V^*| = K$ since additional vertices from V can always be added and we will still have a vertex cover. Let the elements of V^* be labeled as $v_1, v_2, \ldots, v_K$. The following edges are chosen to be "in" the Hamiltonian circuit for G'. From the cover-testing component representing each edge $e=\{u,v\} \in E$, choose the edges specified in Figure 3.5(a), (b), or (c) depending on whether $\{u,v\} \cap V^*$ equals, respectively, $\{u\}$, $\{u,v\}$, or $\{v\}$. One of these three possibilities must hold since V^* is a vertex cover for G. Next, choose all the edges in E'_{v_i} for $1 \leqslant i \leqslant K$. Finally, choose the edges

$$\{a_i,(v_i, e_{v_i[1]},1)\}, 1 \leqslant i \leqslant K$$

$$\{a_{i+1},(v_i, e_{v_i[deg(v_i)]}, 6)\}, 1 \leqslant i < K$$

and

$$\{a_1,(v_K, e_{v_K[deg(v_K)]}, 6)\}$$

We leave to the reader the task of verifying that this set of edges actually corresponds to a Hamiltonian circuit for G'. ■

Several variants of HAMILTONIAN CIRCUIT are also of interest. The HAMILTONIAN PATH problem is the same as HC except that we drop the requirement that the first and last vertices in the sequence be joined by an edge. HAMILTONIAN PATH BETWEEN TWO POINTS is the same as HAMILTONIAN PATH, except that two vertices u and v are specified as part of each instance, and we are asked whether G contains a Hamiltonian path beginning with u and ending with v. Both of these problems can be proved NP-complete using the following simple modification of the transformation just used for HC. We simply modify the graph G' obtained at the end of the construction as follows: add three new vertices, a_0, a_{K+1}, and a_{K+2}, add the two edges $\{a_0,a_1\}$ and $\{a_{K+1},a_{K+2}\}$, and replace each edge of the form $\{a_1,(v, e_{v[deg(v)]}, 6)\}$ by $\{a_{K+1},(v, e_{v[deg(v)]}, 6)\}$. The two specified vertices for the latter variation of HC are a_0 and a_{K+2}.

All three Hamiltonian problems mentioned so far also remain NP-complete if we replace the undirected graph G by a directed graph and replace the undirected Hamiltonian circuit or path by a directed Hamiltonian circuit or path. Recall that a directed graph $G=(V,A)$ consists of a vertex set V and a set of *ordered* pairs of vertices called *arcs*. A Hamiltonian path in a directed graph $G=(V,A)$ is an ordering of V as $<v_1,v_2,\ldots,v_n>$, where $n=|V|$, such that $(v_i,v_{i+1}) \in A$ for $1 \leqslant i < n$. A Hamiltonian circuit has the additional requirement that $(v_n,v_1) \in A$. Each of the three undirected Hamiltonian problems can be transformed to its directed counterpart simply by replacing each edge $\{u,v\}$ in the given undirected graph by the two arcs (u,v) and (v,u). In essence, the undirected versions are merely special cases of their directed counterparts.

SEMINARIO

3.1.5 PARTITION

In this section we consider the last of our six basic NP-complete problems, the PARTITION problem. It is particularly useful for proving NP-completeness results for problems involving numerical parameters, such as lengths, weights, costs, capacities, etc.

Theorem 3.5 PARTITION is NP-complete

Proof: It is easy to see that PARTITION $\in$ NP, since a nondeterministic algorithm need only guess a subset A' of A and check in polynomial time

that the sum of the sizes of the elements in A' is the same as that for the elements in $A-A'$.

We transform 3DM to PARTITION. Let the sets W,X,Y, with $|W|=|X|=|Y|=q$, and $M \subseteq W \times X \times Y$ be an arbitrary instance of 3DM. Let the elements of these sets be denoted by

$$W = \{w_1, w_2, \ldots, w_q\}$$
$$X = \{x_1, x_2, \ldots, x_q\}$$
$$Y = \{y_1, y_2, \ldots, y_q\}$$

and

$$M = \{m_1, m_2, \ldots, m_k\}$$

where $k=|M|$. We must construct a set A, and a size $s(a) \in Z^+$ for each $a \in A$, such that A contains a subset A' satisfying

$$\sum_{a \in A'} s(a) = \sum_{a \in A-A'} s(a)$$

if and only if M contains a matching.

The set A will contain a total of $k+2$ elements and will be constructed in two steps. The first k elements of A are $\{a_i: 1 \leqslant i \leqslant k\}$, where the element a_i is associated with the triple $m_i \in M$. The size $s(a_i)$ of a_i will be specified by giving its binary representation, in terms of a string of 0's and 1's divided into $3q$ "zones" of $p = \lceil \log_2(k+1) \rceil$ bits each. Each of these zones is labeled by an element of $W \cup X \cup Y$, as shown in Figure 3.7.

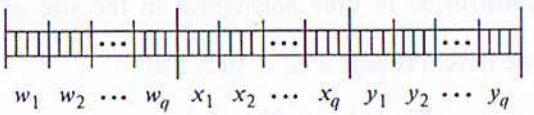

Figure 3.7 Labeling of the $3q$ "zones," each containing $p = \lceil \log_2(k+1) \rceil$ bits of the binary representation for $s(a)$, used in transforming 3DM to PARTITION.

The representation for $s(a_i)$ depends on the corresponding triple $m_i = (w_{f(i)}, x_{g(i)}, y_{h(i)}) \in M$ (where f, g, and h are just the functions that give the subscripts of the first, second, and third components for each m_i). It has a 1 in the rightmost bit position of the zones labeled by $w_{f(i)}$, $x_{g(i)}$, and $y_{h(i)}$ and 0's everywhere else. Alternatively, we can write

$$s(a_i) = 2^{p(3q-f(i))} + 2^{p(2q-g(i))} + 2^{p(q-h(i))}$$

Since each $s(a_i)$ can be expressed in binary with no more than $3pq$ bits, it

is clear that $s(a_i)$ can be constructed from the given 3DM instance in polynomial time.

The important thing to observe about this part of the construction is that, if we sum up all the entries in any zone, over all elements of $\{a_i: 1 \leqslant i \leqslant k\}$, the total can never exceed $k = 2^p - 1$. Hence, in adding up $\sum_{a \in A'} s(a)$ for any subset $A' \subseteq \{a_i: 1 \leqslant i \leqslant k\}$, there will never be any "carries" from one zone to the next. It follows that if we let

$$B = \sum_{j=0}^{3q-1} 2^{pj}$$

(which is the number whose binary representation has a 1 in the rightmost position of every zone), then any subset $A' \subseteq \{a_i: 1 \leqslant i \leqslant k\}$ will satisfy

$$\sum_{a \in A'} s(a) = B$$

if and only if $M' = \{m_i: a_i \in A'\}$ is a matching for M.

The final step of the construction specifies the last two elements of A. These are denoted by b_1 and b_2 and have sizes defined by

$$s(b_1) = 2\left(\sum_{i=1}^{k} s(a_i)\right) - B$$

and

$$s(b_2) = \left(\sum_{i=1}^{k} s(a_i)\right) + B$$

Both of these can be specified in binary with no more than $(3pq+1)$ bits and thus can be constructed in time polynomial in the size of the given 3DM instance.

Now suppose we have a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$$

Then both of these sums must be equal to $2\sum_{i=1}^{k} s(a_i)$, and one of the two sets, A' or $A - A'$, contains b_1 but not b_2. It follows that the remaining elements of that set form a subset of $\{a_i: 1 \leqslant i \leqslant k\}$ whose sizes sum to B, and hence, by our previous comments, that subset corresponds to a matching M' in M. Conversely, if $M' \subseteq M$ is a matching, then the set $\{b_1\} \cup \{a_i: m_i \in M'\}$ forms the desired set A' for the PARTITION instance. Therefore, 3DM $\propto$ PARTITION, and the theorem is proved. ■