Ano: M.R. Garey and D.S. Johnson,
"Computers and Intractability: A guide to the Theory of NP-completeness"
W.H. Freeman & Co, 1979.

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 $\Pi \in \mathrm{NP}$, 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 $<v_1, v_2, \ldots, v_n>$ 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.

$$\begin{tikzcd}
 & \text{SATISFIABILITY} \arrow[d] & & \\
 & \text{3SAT} \arrow[dl] \arrow[dr] & & \\
\text{3DM} \arrow[d] & & \text{VC} \arrow[dl] \arrow[dr] & \\
\text{PARTITION} & \text{HC} & & \text{CLIQUE}
\end{tikzcd}$$

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

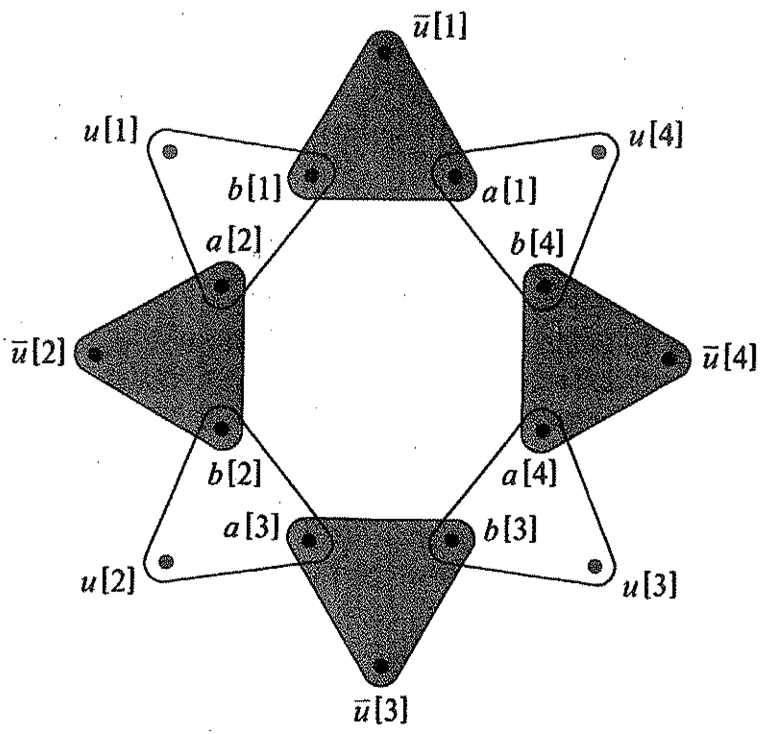

Figure 3.2 Truth setting component T_i when $m=4$ (subscripts have been deleted for simplicity). Either all the sets of T_i^t (the shaded sets) or all the sets of T_i^f (the unshaded sets) must be chosen, leaving uncovered all the $u_i[j]$ or all the $\bar{u}_i[j]$, respectively.

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 $z_j \in \{u_i, \overline{u}_i : 1 \leqslant i \leqslant n\} \cap c_j$ 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 $\overline{u}_i[j]$. 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 $V_i=\{u_i,\overline{u}_i\}$ and $E_i=\{\{u_i,\overline{u}_i\}\}$, 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 $\overline{u}_i$ 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 $C=\{\{u_1,\bar{u}_3,\bar{u}_4\},\{\bar{u}_1,u_2,\bar{u}_4\}\}$.

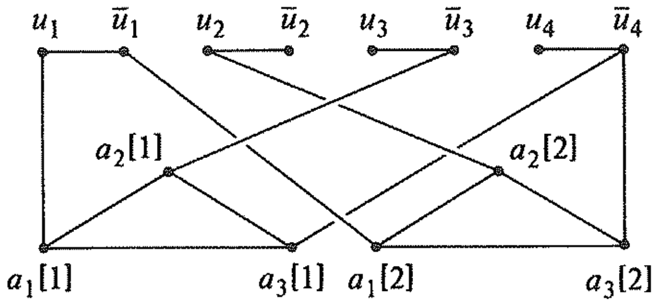

Figure 3.3 VERTEX COVER instance resulting from 3SAT instance in which $U=\{u_1,u_2,u_3,u_4\}$, $C=\{\{u_1,\bar{u}_3,\bar{u}_4\},\{\bar{u}_1,u_2,\bar{u}_4\}\}$. Here $K=n+2m=8$.

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. ■

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 $<v_1, v_2, \ldots, v_n>$ 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,

$$\begin{aligned} E'_e = \; & \{\{(u,e,i),(u,e,i{+}1)\},\{(v,e,i),(v,e,i{+}1)\}: 1 \leqslant i \leqslant 5\} \\ & \cup \{\{(u,e,3),(v,e,1)\},\{(v,e,3),(u,e,1)\}\} \\ & \cup \{\{(u,e,6),(v,e,4)\},\{(v,e,6),(u,e,4)\}\} \end{aligned}$$

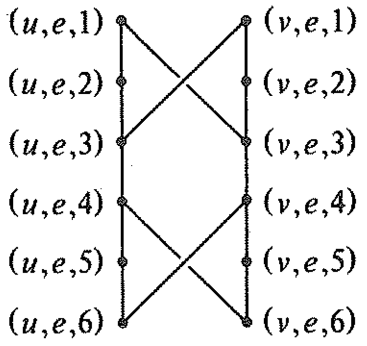

Figure 3.4 Cover-testing component for edge $e = \{u,v\}$ used in transforming VERTEX COVER to HAMILTONIAN CIRCUIT.

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 $x = v$.

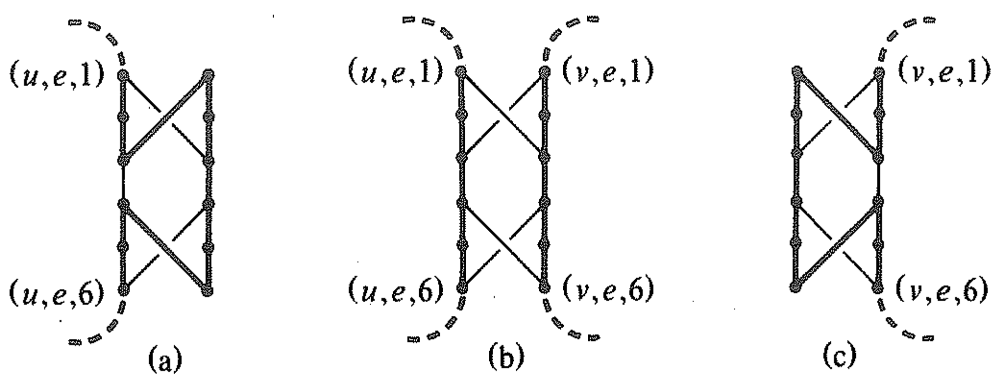

Figure 3.5 The three possible configurations of a Hamiltonian circuit within the cover-testing component for edge $e=\{u,v\}$, corresponding to the cases in which (a) u belongs to the cover but v does not, (b) both u and v belong to the cover, and (c) v belongs to the cover but u does not.

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

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

and

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

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

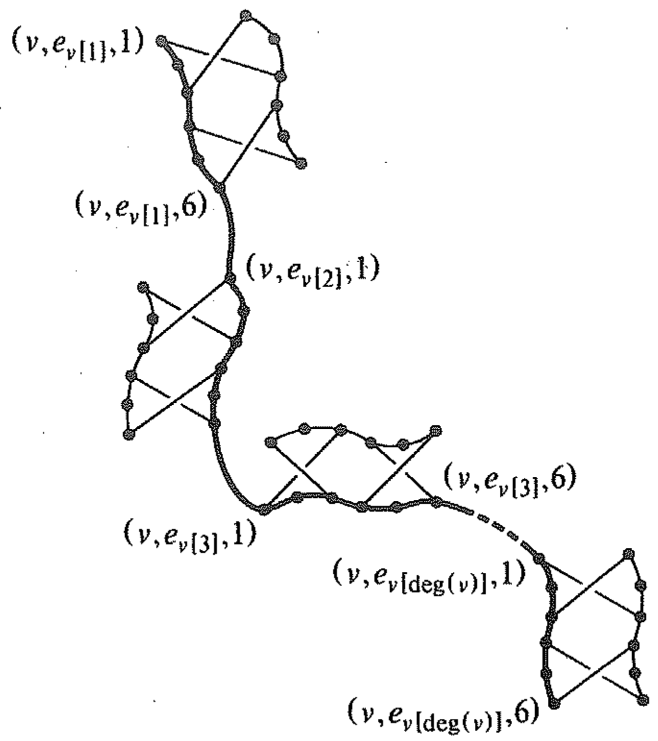

Figure 3.6 Path joining all the cover-testing components for edges from E having vertex v as an endpoint.

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.

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.

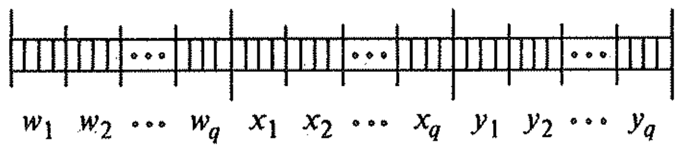

Figure 3.7 Labeling of the $3q$ "zones," each containing $p = \lceil \log_2(k+1) \rceil$ bits of the binary representation for $s(a)$, used in transforming 3DM to PARTITION.

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. ■

3.2 Some Techniques for Proving NP-Completeness

The techniques used for proving NP-completeness results vary almost as widely as the NP-complete problems themselves, and we cannot hope to illustrate them all here. However, there are several general types of proofs that occur frequently and that can provide a suggestive framework for deciding how to go about proving a new problem NP-complete. We call these (a) restriction, (b) local replacement, and (c) component design.

In this section we shall indicate what we mean by each of these proof types, primarily by giving examples. It would be sheer folly to attempt to define them explicitly. Many proofs can be interpreted in ways that would place them arbitrarily in any one of the three categories. Other proofs depend on decidedly problem-specific methods, so that no such limited set of categories could possibly include them in a natural way. Thus, we caution the reader *not* to interpret this as a way to classify all NP-completeness proofs. Rather, our sole intent is to illustrate several ways of thinking about NP-completeness proofs that the authors (and others) have found to be both intuitively appealing and constructive.

For brevity in what follows, we shall be omitting from all our proofs the verification that the given problem is in NP. Each of the problems we consider is easily seen to be solvable in polynomial time by a nondeterministic algorithm, and the reader should have no difficulty supplying such an algorithm whenever required.

3.2.1 Restriction

Proof by restriction is the simplest, and perhaps the most frequently applicable, of our three proof types. An NP-completeness proof by restriction for a given problem $\Pi \in \text{NP}$ consists simply of showing that Π contains a known NP-complete problem Π' as a special case. The heart of such a proof lies in the specification of the additional restrictions to be placed on the instances of Π so that the resulting restricted problem will be identical to Π'. We do not require that the restricted problem and the known NP-complete problem be *exact* duplicates of one another, but rather that there be an "obvious" one-to-one correspondence between their instances that preserves "yes" and "no" answers. This one-to-one correspondence, which provides the required transformation from Π' to Π, is usually so apparent that it need not even be given explicitly.

We have already seen several examples of this type of proof. In Section 3.1.2, the problem EXACT COVER BY 3-SETS was shown to be NP-complete by restricting its instances to 3-sets that contain one element from a set W, one from a set X, and one from a set Y, where W, X, and Y are disjoint sets having the same cardinality, thereby obtaining a problem identical to the 3DM problem. In Section 3.1.4, DIRECTED HAMILTONIAN

CIRCUIT was shown to be NP-complete by restricting its instances to directed graphs in which each arc (u,v) occurs only in conjunction with the oppositely directed arc (v,u), thereby obtaining a problem identical to the undirected HAMILTONIAN CIRCUIT problem.

Thus proofs by restriction can be seen to embody a different way of looking at things than the standard NP-completeness proofs. Instead of trying to discover a way of transforming a known NP-complete problem to our target problem, we focus on the target problem itself and attempt to restrict away its "inessential" aspects until a known NP-complete problem appears.

We now give a number of additional examples of problems proved NP-complete by restriction, stating each proof with the brevity it deserves.

(1) **MINIMUM COVER**
INSTANCE: Collection C of subsets of a set S, positive integer K.
QUESTION: Does C contain a *cover* for S of size K or less, that is, a subset $C' \subseteq C$ with $|C'| \leqslant K$ and such that $\bigcup_{c \in C'} c = S$?
Proof: Restrict to X3C by allowing only instances having $|c|=3$ for all $c \in C$ and having $K = |S|/3$.

(2) **HITTING SET**
INSTANCE: Collection C of subsets of a set S, positive integer K.
QUESTION: Does S contain a *hitting set* for C of size K or less, that is, a subset $S' \subseteq S$ with $|S'| \leqslant K$ and such that S' contains at least one element from each subset in C?
Proof: Restrict to VC by allowing only instances having $|c|=2$ for all $c \in C$.

(3) **SUBGRAPH ISOMORPHISM**
INSTANCE: Two graphs, $G=(V_1,E_1)$ and $H=(V_2,E_2)$.
QUESTION: Does G contain a subgraph *isomorphic* to H, that is, a subset $V \subseteq V_1$ and a subset $E \subseteq E_1$ such that $|V|=|V_2|, |E|=|E_2|$, and there exists a one-to-one function $f: V_2 \rightarrow V$ satisfying $\{u,v\} \in E_2$ if and only if $\{f(u),f(v)\} \in E$?
Proof: Restrict to CLIQUE by allowing only instances for which H is a complete graph, that is, E_2 contains all possible edges joining two members of V_2.

(4) **BOUNDED DEGREE SPANNING TREE**
INSTANCE: A graph $G=(V,E)$ and a positive integer $K \leqslant |V|-1$.
QUESTION: Is there a *spanning tree* for G in which no vertex has degree exceeding K, that is, a subset $E' \subseteq E$ such that $|E'|=|V|-1$, the graph $G'=(V,E')$ is connected, and no vertex in V is included in more than K edges from E'?
Proof: Restrict to HAMILTONIAN PATH by allowing only instances in which $K=2$.

(5) **MINIMUM EQUIVALENT DIGRAPH**
INSTANCE: A directed graph $G=(V,A)$ and a positive integer $K \leqslant |A|$.
QUESTION: Is there a directed graph $G'=(V,A')$ such that $A' \subseteq A$, $|A'| \leqslant K$, and such that, for every pair of vertices u and v in V, G' contains a directed path from u to v if and only if G contains a directed path from u to v.
Proof: Restrict to DIRECTED HAMILTONIAN CIRCUIT by allowing only instances in which G is strongly connected, that is, contains a path from every vertex u to every vertex v, and $K=|V|$. Note that this is actually a restriction to DIRECTED HAMILTONIAN CIRCUIT FOR STRONGLY CONNECTED DIGRAPHS, but the NP-completeness of that problem follows immediately from the constructions we gave for HC and DIRECTED HC.

(6) **KNAPSACK**
INSTANCE: A finite set U, a "size" $s(u) \in Z^+$ and a "value" $v(u) \in Z^+$ for each $u \in U$, a size constraint $B \in Z^+$, and a value goal $K \in Z^+$.
QUESTION: Is there a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s(u) \leqslant B \quad \text{and} \quad \sum_{u \in U'} v(u) \geqslant K$$

Proof: Restrict to PARTITION by allowing only instances in which $s(u)=v(u)$ for all $u \in U$ and $B=K=½\sum_{u \in U} s(u)$.

(7) **MULTIPROCESSOR SCHEDULING**
INSTANCE: A finite set A of "tasks," a "length" $l(a) \in Z^+$ for each $a \in A$, a number $m \in Z^+$ of "processors," and a "deadline" $D \in Z^+$.
QUESTION: Is there a partition $A=A_1 \cup A_2 \cup \cdots \cup A_m$ of A into m disjoint sets such that

$$\max\left\{\sum_{a \in A_i} l(a) : 1 \leqslant i \leqslant m\right\} \leqslant D \ ?$$

Proof: Restrict to PARTITION by allowing only instances in which $m=2$ and $D=½\sum_{a \in A} l(a)$.

As a final comment, we observe that, of all the approaches to proving NP-completeness we shall discuss, proof by restriction is the one that would profit most from an extensive knowledge of the class of known NP-complete problems — beyond the basic six and their variants. Many problems that arise in practice are simply more complicated versions of problems

that appear on our lists of NP-complete problems, and the ability to recognize this can often lead to a quick NP-completeness proof by restriction.

3.2.2 Local Replacement

In proofs by local replacement, the transformations are sufficiently nontrivial to warrant spelling out in the standard proof format, but they still tend to be relatively uncomplicated. All we do is pick some aspect of the known NP-complete problem instance to make up a collection of *basic units*, and we obtain the corresponding instance of the target problem by replacing each basic unit, in a uniform way, with a different structure. The transformation from SAT to 3SAT in Section 3.1.1 was of this type. In that transformation, the basic units of an instance of SAT were the clauses, and each clause was replaced by a collection of clauses according to the same general rule. The key point to observe is that each replacement constituted only local modification of structure. The replacements were essentially independent of one another, except insofar as they reflected parts of the original instance that were not changed.

Let us flesh these generalities out with some more examples. The following decision problem corresponds to a problem of minimizing the number of multiplications needed to compute a given collection of products of elementary terms, where the multiplication operation is assumed to be associative and commutative:

ENSEMBLE COMPUTATION
INSTANCE: A collection C of subsets of a finite set A and a positive integer J.
QUESTION: Is there a sequence

$$<z_1=x_1 \cup y_1, z_2=x_2 \cup y_2, \ldots, z_j=x_j \cup y_j>$$

of $j \leqslant J$ union operations, where each x_i and y_i is either $\{a\}$ for some $a \in A$ or z_k for some $k<i$, such that x_i and y_i are disjoint for $1 \leqslant i \leqslant j$ and such that for every subset $c \in C$ there is some z_i, $1 \leqslant i \leqslant j$, that is identical to c ?

Theorem 3.6 ENSEMBLE COMPUTATION is NP-complete.
Proof: We transform VERTEX COVER to ENSEMBLE COMPUTATION. Let the graph $G=(V,E)$ and the positive integer $K \leqslant |V|$ constitute an arbitrary instance of VC.

The basic units of the instance of VC are the edges of G. Let a_0 be some new element not in V. The local replacement just substitutes for each edge $\{u,v\} \in E$ the subset $\{a_0,u,v\} \in C$. The instance of ENSEMBLE COMPUTATION is completely specified by:

$$A = V \cup \{a_0\}$$
$$C = \{\{a_0, u, v\} : \{u, v\} \in E\}$$
$$J = K + |E|$$

It is easy to see that this instance can be constructed in polynomial time. We claim that G has a vertex cover of size K or less if and only if the desired sequence of $j \leqslant J$ operations exists for C.

First, suppose V' is a vertex cover for G of size K or less. Since we can add additional vertices to V' and it will remain a vertex cover, there is no loss of generality in assuming that $|V'| = K$. Label the elements of V' as $v_1, v_2, \ldots, v_K$ and label the edges in E as $e_1, e_2, \ldots, e_m$, where $m = |E|$. Since V' is a vertex cover, each edge e_j contains at least one element from V'. Thus we can write each e_j as $e_j = \{u_j, v_{r[j]}\}$, where $r[j]$ is an integer satisfying $1 \leqslant r[j] \leqslant K$. The following sequence of $K + |E| = J$ operations is easily seen to have all the required properties:

$$<z_1 = \{a_0\} \cup \{v_1\},\ z_2 = \{a_0\} \cup \{v_2\},\ \ldots,\ z_k = \{a_0\} \cup \{v_K\},$$
$$z_{K+1} = \{u_1\} \cup z_{r[1]},\ z_{K+2} = \{u_2\} \cup z_{r[2]},\ \ldots,\ z_J = \{u_m\} \cup z_{r[m]}>$$

Conversely, suppose $S = <z_1 = x_1 \cup y_1, \ldots, z_j = x_j \cup y_j>$ is the desired sequence of $j \leqslant J$ operations for the ENSEMBLE COMPUTATION instance. Furthermore, let us assume that S is the shortest such sequence for this instance and that, among all such minimum sequences, S contains the fewest possible operations of the form $z_i = \{u\} \cup \{v\}$ for $u, v \in V$. Our first claim is that S can contain *no* operations of this latter form. For suppose that $z_i = \{u\} \cup \{v\}$ with $u, v \in V$ is included. Since $\{u, v\}$ is not in C and since S has minimum length, we must have $\{u, v\} \in E$, and $\{a_0, u, v\} = \{a_0\} \cup z_i$ (or $z_i \cup \{a_0\}$) must occur later in S. However, since $\{u, v\}$ is a subset of only one member of C, z_i cannot be used in any other operation in this minimum length sequence. It follows that we can replace the two operations

$$z_i = \{u\} \cup \{v\} \text{ and } \{a_0, u, v\} = \{a_0\} \cup z_i$$

by

$$z_i = \{a_0\} \cup \{u\} \text{ and } \{a_0, u, v\} = \{v\} \cup z_i$$

thereby reducing the number of proscribed operations without lengthening the overall sequence, a contradiction to the choice of S. Hence S consists only of operations having one of the two forms, $z_i = \{a_0\} \cup \{u\}$ for $u \in V$ or $\{a_0, u, v\} = \{v\} \cup z_i$ for $\{u, v\} \in E$ (where we disregard the relative order of the two operands in each case). Because $|C| = |E|$ and because every member of C contains three elements, S must contain exactly $|E|$ operations of the latter form and exactly $j - |E| \leqslant J - |E| = K$ of the former.

Therefore the set

$$V' = \{u \in V\colon z_i = \{a_0\} \cup \{u\} \text{ is an operation in } S\}$$

contains at most K vertices from V and, as can be verified easily from the construction of C, must be a vertex cover for G. ■

Another example of a polynomial time transformation using local replacement, this time from EXACT COVER BY 3-SETS, is the following:

PARTITION INTO TRIANGLES
INSTANCE: A graph $G=(V,E)$, with $|V|=3q$ for a positive integer q.
QUESTION: Is there a partition of V into q disjoint sets $V_1, V_2, \ldots, V_q$ of three vertices each such that, for each $V_i = \{v_{i[1]}, v_{i[2]}, v_{i[3]}\}$, the three edges $\{v_{i[1]}, v_{i[2]}\}$, $\{v_{i[1]}, v_{i[3]}\}$, and $\{v_{i[2]}, v_{i[3]}\}$ all belong to E?

Theorem 3.7 PARTITION INTO TRIANGLES is NP-complete.
Proof: We transform EXACT COVER BY 3-SETS to PARTITION INTO TRIANGLES. Let the set X with $|X|=3q$ and the collection C of 3-element subsets of X be an arbitrary instance of X3C. We shall construct a graph $G=(V,E)$, with $|V|=3q'$, such that the desired partition exists for G if and only if C contains an exact cover.

The basic units of the X3C instance are the 3-element subsets in C. The local replacement substitutes for each such subset $c_i = \{x_i, y_i, z_i\} \in C$ the collection E_i of 18 edges shown in Figure 3.8. Thus $G=(V,E)$ is defined by

$$V = X \cup \bigcup_{i=1}^{|C|} \{a_i[j]\colon 1 \leqslant j \leqslant 9\}$$

$$E = \bigcup_{i=1}^{|C|} E_i$$

Notice that the only vertices that appear in edges belonging to more than a single E_i are those that are in the set X. Notice also that $|V| = |X| + 9|C| = 3q + 9|C|$ so that $q' = q + 3|C|$. It is not hard to see that this instance of PARTITION INTO TRIANGLES can be constructed in polynomial time from the X3C instance.

If $c_1, c_2, \ldots, c_q$ are the 3-element subsets from C in any exact cover for X, then the corresponding partition $V = V_1 \cup V_2 \cup \cdots \cup V_{q'}$ of V is given by taking

$$\{a_i[1], a_i[2], x_i\},\ \{a_i[4], a_i[5], y_i\}$$
$$\{a_i[7], a_i[8], z_i\},\ \{a_i[3], a_i[6], a_i[9]\}$$

from the vertices meeting E_i whenever $c_i = \{x_i, y_i, z_i\}$ is in the exact cover,

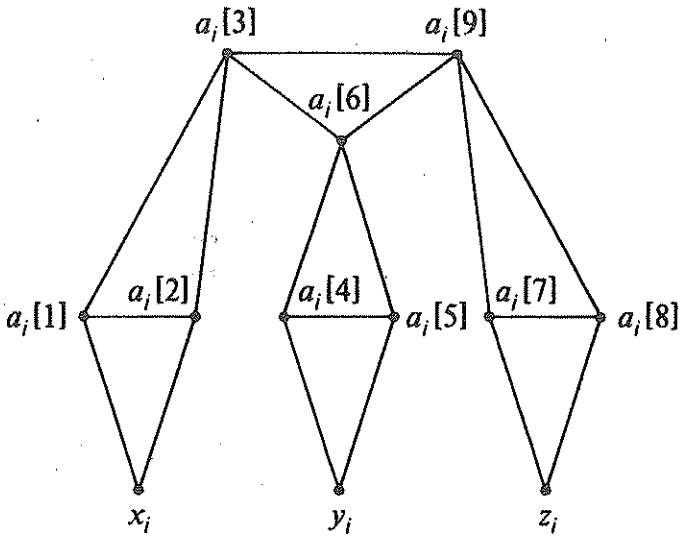

Figure 3.8 Local replacement for $c_i = (x_i, y_i, z_i) \in C$ for transforming X3C to PARTITION INTO TRIANGLES.

and by taking

$$\{a_i[1],a_i[2],a_i[3]\},\ \{a_i[4],a_i[5],a_i[6]\},\ \{a_i[7],a_i[8],a_i[9]\}$$

from the vertices meeting E_i whenever c_i is *not* in the exact cover. This ensures that each element of X is included in exactly one 3-vertex subset in the partition.

Conversely, if $V = V_1 \cup V_2 \cup \cdots \cup V_{q'}$ is any partition of G into triangles, the corresponding exact cover is given by choosing those $c_i \in C$ such that $\{a_i[3],a_i[6],a_i[9]\} = V_j$ for some j, $1 \leqslant j \leqslant q'$. We leave to the reader the straightforward task of verifying that the two partitions we have constructed are as claimed. ■

Both examples we have just seen represent what might be called "pure" local replacement proofs. The structure of the target instance was completely determined by the structure of the given problem instance and the local replacements. It is often advantageous to augment this with a limited amount of additional structure that acts as an "enforcer,"† imposing certain additional restrictions on the ways in which a "yes" answer to the target instance can be obtained. For a target problem having the form "Given an instance I, does there exist an X_I having the desired property?" the enforcer portion of I acts to limit the possible X_I's so that the remaining choices all mirror the choices available in the original problem instance, whereas that portion of I obtained by applying local replacement to the original instance provides the means for making those choices and for ensuring that they have the desired properties. The two elements b_1 and b_2 in the

† A picturesque term suggested by Szymanski [1978].

NP-completeness proof for PARTITION acted as such an enforcer. We give two further examples of local replacement proofs using enforcers, beginning with that for the following scheduling problem:

SEQUENCING WITHIN INTERVALS
INSTANCE: A finite set T of "tasks" and, for each $t \in T$, an integer "release time" $r(t) \geqslant 0$, a "deadline" $d(t) \in Z^+$, and a "length" $l(t) \in Z^+$.
QUESTION: Does there exist a *feasible schedule* for T, that is, a function $\sigma: T \rightarrow Z^+$ such that, for each $t \in T$, $\sigma(t) \geqslant r(t)$, $\sigma(t)+l(t) \leqslant d(t)$, and, if $t' \in T-\{t\}$, then either $\sigma(t')+l(t') \leqslant \sigma(t)$ or $\sigma(t') \geqslant \sigma(t)+l(t)$? (The task t is "executed" from time $\sigma(t)$ to time $\sigma(t)+l(t)$, cannot start executing until time $r(t)$, must be completed by time $d(t)$, and its execution cannot overlap the execution of any other task t'.)

Theorem 3.8 SEQUENCING WITHIN INTERVALS is NP-complete.
Proof: We transform PARTITION to this problem. Let the finite set A and given size $s(a)$ for each $a \in A$ constitute an arbitrary instance of PARTITION, and let $B = \sum_{a \in A} s(a)$.

The basic units of the PARTITION instance are the individual elements $a \in A$. The local replacement for each $a \in A$ is a single task t_a with $r(t_a)=0$, $d(t_a)=B+1$, and $l(t_a)=s(a)$. The "enforcer" is a single task $\bar{t}$ with $r(\bar{t}) = \lceil B/2 \rceil$, $d(\bar{t}) = \lceil (B+1)/2 \rceil$, and $l(\bar{t})=1$. Clearly, this instance can be constructed in polynomial time from the PARTITION instance.

The restrictions imposed on feasible schedules by the enforcer are twofold. First, it ensures that a feasible schedule cannot be constructed whenever B is an odd integer (in which case the desired subset for the PARTITION instance cannot exist), because then we would have $r(\bar{t}) = d(\bar{t})$, so that $\bar{t}$ could not possibly be scheduled. Thus from now on, let us assume that B is even. In this case the second restriction comes to the forefront. Since B is even, $r(\bar{t}) = B/2$ and $d(\bar{t}) = r(\bar{t})+1$, so that any feasible schedule must have $\sigma(\bar{t}) = B/2$. This divides the time available for scheduling the remaining tasks into two separate blocks, each of total length $B/2$, as illustrated in Figure 3.9. Thus the scheduling problem is turned into a problem of selecting subsets, those that are scheduled before $\bar{t}$ and those that are scheduled after $\bar{t}$. Since the total amount of time available in the two blocks equals the total length B of the remaining tasks, it follows that each block must be filled up exactly. However, this can be done if and only if there is a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = B/2 = \sum_{a \in A-A'} s(a)$$

Thus the desired subset A' exists for the instance of PARTITION if and only if a feasible schedule exists for the corresponding instance of SEQUENCING WITHIN INTERVALS. ■

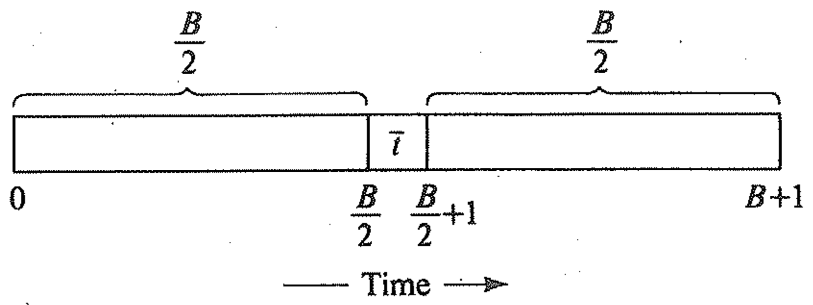

Figure 3.9 Schedule "enforced" by the transformation from PARTITION to SEQUENCING WITHIN INTERVALS.

Our final example of the use of an enforcer in a local replacement proof involves the following problem of diagnostic testing:

MINIMUM TEST COLLECTION
INSTANCE: A finite set A of "possible diagnoses," a collection C of subsets of A, representing binary "tests," and a positive integer $J \leqslant |C|$.
QUESTION: Is there a subcollection $C' \subseteq C$ with $|C'| \leqslant J$ such that, for every pair a_i, a_j of possible diagnoses from A, there is some test $c \in C'$ for which $|\{a_i, a_j\} \cap c| = 1$ (that is, a test c that "distinguishes" between a_i and a_j)?

Theorem 3.9 MINIMUM TEST COLLECTION is NP-complete.
Proof: We transform 3DM to this problem. Let the sets W, X, Y, with $|W| = |X| = |Y| = q$, and the collection $M \subseteq W \times X \times Y$ constitute an arbitrary instance of 3DM.

The basic units of the 3DM instance are the ordered triples in M. The local replacement substitutes for each $m = (w,x,y) \in M$ the subset $\{w,x,y\} \in C$. The enforcer is provided by three additional elements, w_0, x_0, and y_0, not belonging to $W \cup X \cup Y$, and two additional tests, $W \cup \{w_0\}$ and $X \cup \{x_0\}$. The complete MINIMUM TEST COLLECTION instance is defined by:

$$A = W \cup X \cup Y \cup \{w_0, x_0, y_0\}$$
$$C = \{\{w,x,y\} : (w,x,y) \in M\} \cup \{W \cup \{w_0\}, X \cup \{x_0\}\}$$
$$J = q + 2$$

It is easy to see that this instance can be constructed in polynomial time from the given 3DM instance.

Once again the enforcer places certain limitations on the form of the desired entity (in this case, the subcollection C' of tests). First, C' must contain both $W \cup \{w_0\}$ and $X \cup \{x_0\}$, since they are the only tests that

distinguish y_0 from w_0 and x_0. Then, since w_0, x_0, and y_0 are not contained in any other tests in C, each element of $W \cup X \cup Y$ must be distinguished from the appropriate one of w_0, x_0, or y_0 by being included in some additional test $c \in C' - \{W \cup \{w_0\}, X \cup \{x_0\}\}$. At most $J-2=q$ such additional tests can be included. Because each of the remaining tests in C contains exactly one member from each of W, X, and Y, and because W, X, and Y are disjoint sets, having q members each, it follows that any such additional q tests in C' must correspond to q triples that form a matching for M. Conversely, given any matching for M, the corresponding q tests from C can be used to complete the desired collection of $J=q+2$ tests. Thus M contains a matching if and only if the required subcollection of tests from C exists. ■

Although the enforcers in both our examples are quite simple, the reader should be placed on notice that this need not always be the case. A particularly complicated enforcing structure is used in the NP-completeness proof for PLANAR DIRECTED HAMILTONIAN PATH in [Garey, Johnson, and Stockmeyer, 1976]. Other relatively complicated enforcers can be found in [Liu and Geldmacher, 1978], [Garey, Johnson, and Sethi, 1976], and [Garey, Graham, Johnson, and Knuth, 1978].

3.2.3 Component Design

Our last type of proof, and the one that tends to be the most complicated, is component design. The NP-completeness proofs given in Section 3.1 for 3-DIMENSIONAL MATCHING, VERTEX COVER, and HAMILTONIAN CIRCUIT are typical examples of this type of proof.

The basic idea is to use the constituents of the target problem instance to design certain "components" that can be combined to "realize" instances of the known NP-complete problem. In these three examples, there are two basic types of components, ones that can be viewed as "making choices" (for example, selecting vertices, choosing truth values for variables) and ones for "testing properties" (for example, checking that each edge is covered, checking that each clause is satisfied). These components are joined together in a target instance in such a way that the choices are communicated to the property testers, and the property testers then check whether the choices made satisfy the required constraints. Interactions between components occur both through direct connections (such as the edges linking the truth setting components to the satisfaction testing components in the transformation from 3SAT to VC) and through global constraints (such as the overall bound K in the transformation from 3SAT to VC, which, together with the structure of the components, ensures that each truth setting component contains exactly one vertex from the cover and that each satisfaction testing component contains exactly two vertices from the cover).

More generally, any proof in which the constructed instance can be viewed as a collection of components, each performing some function in terms of the given instance, can be regarded as a component design proof. The generic transformation used to prove Cook's Theorem in Chapter 2 is a good example of this, with each of the six clause groups being one type of component.

Since component design proofs tend to be rather lengthy and since we have already given a number of examples of such proofs, we shall confine ourselves to a single additional example in this section. (More can be found in [Sethi, 1975], [Even, Itai, and Shamir, 1976], [Garey, Johnson, and Tarjan, 1976] and [Stockmeyer, 1973].) This final example is quite different from the standard ones, and illustrates an approach that has been useful for transforming CLIQUE to several other problems. The target problem is a scheduling problem related to the problem of SEQUENCING WITHIN INTERVALS proved NP-complete in the preceding subsection.

MINIMUM TARDINESS SEQUENCING

INSTANCE: A set T of "tasks," each $t \in T$ having "length" 1 and a "deadline" $d(t) \in Z^+$, a partial order $<$ on T, and a non-negative integer $K \leqslant |T|$.

QUESTION: Is there a "schedule" $\sigma: T \rightarrow \{0,1, \ldots, |T|-1\}$ such that $\sigma(t) \neq \sigma(t')$ whenever $t \neq t'$, such that $\sigma(t) < \sigma(t')$ whenever $t < t'$, and such that $|\{t \in T: \sigma(t)+1 > d(t)\}| \leqslant K$?

Theorem 3.10 MINIMUM TARDINESS SEQUENCING is NP-complete.

Proof: Let the graph $G=(V,E)$ and the positive integer $J \leqslant |V|$ constitute an arbitrary instance of CLIQUE. The corresponding instance of MINIMUM TARDINESS SEQUENCING has task set $T = V \cup E$, $K = |E| - (J(J-1)/2)$, and partial order and deadlines defined as follows:

$$t < t' \iff t \in V, t' \in E, \text{ and vertex } t \text{ is an endpoint of edge } t'$$

$$d(t) = \begin{cases} J(J+1)/2 & \text{if } t \in E \\ |V|+|E| & \text{if } t \in V \end{cases}$$

Thus the "component" corresponding to each vertex is a single task with deadline $|V|+|E|$, and the "component" corresponding to each edge is a single task with deadline $J(J+1)/2$. The task corresponding to an edge is forced by the partial order to occur after the tasks corresponding to its two endpoints in the desired schedule, and only edge tasks are in danger of being tardy (being completed after their deadlines).

It is convenient to view the desired schedule schematically, as shown in Figure 3.10. We can think of the portion of the schedule before the edge task deadline as our "clique selection component." There is room for $J(J+1)/2$ tasks before this deadline. In order to have no more than the

specified number of tardy tasks, at least $J(J-1)/2$ of these "early" tasks must be edge tasks. However, if an edge task precedes this deadline, then so must the vertex tasks corresponding to its endpoints. The minimum possible number of vertices that can be involved in $J(J-1)/2$ distinct edges is J (which can happen if and only if those edges form a complete graph on those J vertices). This implies that there must be *at least* J vertex tasks among the "early" tasks. However, there is room for *at most*

$$(J(J+1)/2) - (J(J-1)/2) = J$$

vertex tasks before the edge task deadline. Therefore, any such schedule must have *exactly* J vertex tasks and *exactly* $J(J-1)/2$ edge tasks before this deadline, and these must correspond to a J-vertex clique in G. Conversely, if G contains a complete subgraph of size J, the desired schedule can be constructed as in Figure 3.10. ■

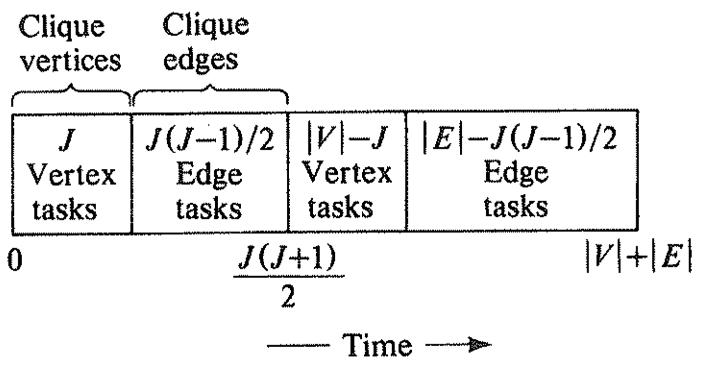

Figure 3.10 Diagram of the desired schedule for an instance of MINIMUM TARDINESS SEQUENCING corresponding to a CLIQUE of size J.

3.3 Some Suggested Exercises

In this section we present the definitions of twelve NP-complete problems and leave to the reader the task of proving that they are NP-complete. None of these problems requires a complicated proof, so we encourage the reader to attempt them all. For the purposes of these exercises, only those "known" NP-complete problems mentioned in Section 3.1 should be used. As a hint for how to proceed, we have grouped the problems according to our own preferred proof technique, but the reader should feel free to ignore these hints whenever an alternative approach seems worthy of pursuit. Those desiring additional (or more difficult) exercises can choose from the lists included in the Appendix, keeping in mind that these lists contain some problems for which only quite elaborate proofs are known.

Restriction

1. **LONGEST PATH**
INSTANCE: Graph $G=(V,E)$, positive integer $K \leqslant |V|$.
QUESTION: Does G contain a simple path (that is, a path encountering no vertex more than once) with K or more edges?

2. **SET PACKING**
INSTANCE: Collection C of finite sets, positive integer $K \leqslant |C|$.
QUESTION: Does C contain K disjoint sets?

3. **PARTITION INTO HAMILTONIAN SUBGRAPHS**
INSTANCE: Graph $G=(V,E)$, positive integer $K \leqslant |V|$.
QUESTION: Can the vertices of G be partitioned into $k \leqslant K$ disjoint sets $V_1, V_2, \ldots, V_k$ such that, for $1 \leqslant i \leqslant k$, the subgraph induced by V_i contains a Hamiltonian circuit?

4. **LARGEST COMMON SUBGRAPH**
INSTANCE: Graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$, positive integer K.
QUESTION: Do there exist subsets $E_1' \subseteq E_1$ and $E_2' \subseteq E_2$ such that $|E_1'|=|E_2'| \geqslant K$ and such that the two subgraphs $G_1'=(V_1,E_1')$ and $G_2'=(V_2,E_2')$ are isomorphic?

5. **MINIMUM SUM OF SQUARES**
INSTANCE: Finite set A, "size" $s(a) \in Z^+$ for each $a \in A$, positive integers K and J.
QUESTION: Can the elements of A be partitioned into K disjoint sets $A_1, A_2, \ldots, A_K$ such that $\sum_{i=1}^{K} \left(\sum_{a \in A_i} s(a) \right)^2 \leqslant J$?

Local Replacement

6. **FEEDBACK VERTEX SET**
INSTANCE: Directed graph $G=(V,A)$, positive integer $K \leqslant |V|$.
QUESTION: Is there a subset $V' \subseteq V$ such that $|V'| \leqslant K$ and such that every directed circuit in G includes at least one vertex from V'?

7. **EXACT COVER BY 4-SETS**
INSTANCE: Finite set X with $|X|=4q$, q an integer, and a collection C of 4-element subsets of X.
QUESTION: Is there a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

8. **DOMINATING SET**
INSTANCE: Graph $G=(V,E)$, positive integer $K \leqslant |V|$.
QUESTION: Is there a subset $V' \subseteq V$ such that $|V'| \leqslant K$ and such that every vertex $v \in V-V'$ is joined to at least one member of V' by an edge in E?

9. **STEINER TREE IN GRAPHS**
INSTANCE: Graph $G=(V,E)$, subset $R \subseteq V$, positive integer $K \leqslant |V|-1$.
QUESTION: Is there a subtree of G that includes all the vertices of R and that contains no more than K edges?

10. **STAR-FREE REGULAR EXPRESSION INEQUIVALENCE**
INSTANCE: Two star-free regular expressions E_1 and E_2 over a finite alphabet Σ, where such expressions are defined by (1) any single symbol $\sigma \in \Sigma$ is a star-free regular expression, and (2) if e_1 and e_2 are star-free regular expressions, then the strings $e_1 e_2$ and $(e_1 \vee e_2)$ are star-free regular expressions.
QUESTION: Do E_1 and E_2 represent different languages over Σ, where the language represented by $\sigma \in \Sigma$ is $\{\sigma\}$, and, if e_1 and e_2 represent the languages L_1 and L_2 respectively, then $e_1 e_2$ represents the language $\{xy : x \in L_1 \text{ and } y \in L_2\}$ and $(e_1 \vee e_2)$ represents the language $L_1 \cup L_2$?

Component Design

11. **SET SPLITTING**
INSTANCE: Collection C of subsets of a finite set S.
QUESTION: Is there a partition of S into two subsets S_1 and S_2 such that no subset in C is entirely contained in either S_1 or S_2?
Hint: Use 3SAT.

12. **PARTITION INTO PATHS OF LENGTH 2**
INSTANCE: Graph $G=(V,E)$, with $|V|=3q$ for a positive integer q.
QUESTION: Is there a partition of V into q disjoint sets $V_1, V_2, \ldots, V_q$ of three vertices each so that, for each $V_i = \{v_{i[1]}, v_{i[2]}, v_{i[3]}\}$, at least two of the three edges $\{v_{i[1]}, v_{i[2]}\}$, $\{v_{i[1]}, v_{i[3]}\}$, and $\{v_{i[2]}, v_{i[3]}\}$ belong to E?
Hint: Use 3DM.

13. **GRAPH GRUNDY NUMBERING**
INSTANCE: Directed graph $G=(V,A)$.
QUESTION: Is there a labeling $L: V \rightarrow Z^+$ (where the same label may be assigned to more than one vertex) such that, for each $v \in V$, $L(v)$ is the least non-negative integer not in the set $\{L(u) : u \in V, (v,u) \in A\}$?
Hint: Use 3SAT.

14. **GRAPH 3-COLORABILITY**
INSTANCE: Graph $G=(V,E)$.
QUESTION: Is G 3-colorable, that is, does there exist a function $f: V \rightarrow \{1,2,3\}$ such that $f(u) \neq f(v)$ whenever $\{u,v\} \in E$?
Hint: Use 3SAT.