4. The Set Covering Problem

If one considers the transpose of the adjacency matrix $\mathbf{A}^t$, with all diagonal elements set to 1, then the problem of finding the minimum dominating set is equivalent to the problem of choosing the least number of columns so that every row contains an entry of 1 under at least one of the chosen columns. This last problem of finding the minimum number of columns to "cover" all the rows has been studied extensively under the name of the *Set Covering Problem* (SCP).

In the general SCP the matrix of 0–1 coefficients need not be square as is the case with the adjacency matrix. Moreover, associated with each column j, (vertex x_j in our case), is a cost c_j, and what is required is to choose a cover (dominating vertex set in the graph case) with the least total cost. One may—at first sight—suppose that since the problem of finding the minimum dominating vertex set is a very special covering problem with $c_j = 1$ for all $j = 1, 2, \ldots, n$, the finding of this set may in fact be much simpler than the solution of the general SCP. However, this is not generally the case and in this section we will, therefore, be concerned with the solution of the general SCP.

4.1 Problem formulation

The SCP owes its name to the following set-theoretic interpretation. Given a set $R = \{r_1, \ldots, r_M\}$, and a family $\mathscr{S} = \{S_1, \ldots, S_N\}$ of sets $S_j \subset R$ any subfamily $\mathscr{S}' = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$ of $\mathscr{S}$ such that

$$\bigcup_{i=1}^{k} S_{j_i} = R \tag{3.12}$$

is called a *set-covering of* R, and the S_{j_i} are called the *covering sets*. If, in addition to equation (3.12), $\mathscr{S}'$ also satisfies

$$S_{j_h} \cap S_{j_l} = \phi, \qquad \forall h, l \in \{1, \ldots, k\}, \qquad h \neq l \tag{3.13}$$

i.e. the S_{j_i} $(i = 1, \ldots, k)$ are pairwise disjoint, then $\mathscr{S}'$ is called a *set-partitioning of* R.

If, with each $S_j \in \mathscr{S}$, there is associated a (positive) cost c_j, the SCP is to find that set-covering of R which has minimum cost, the cost of $\mathscr{S}' = \{S_{j_1}, \ldots, S_{j_k}\}$ being $\sum_{i=1}^{k} c_{j_i}$. The *Set Partitioning Problem* (SPP) is defined correspondingly.

In the matrix form referred to earlier—where the rows of an M-row N-column 0–1 matrix $[t_{ij}]$ are to be covered by columns—the SCP can be formulated as a 0–1 linear program:

Minimize: $$z = \sum_{j=1}^{N} c_j \xi_j$$

subject to: $$\sum_{j=1}^{N} t_{ij}\xi_j \geqslant 1 \quad i = 1, 2, \ldots, M. \tag{3.14}$$

where $c_j \geqslant 0$, and ξ_j is 1, (0) depending on $S_j \in \mathscr{S}'$, $(S_j \notin \mathscr{S}')$ and where $t_{ij} = 1, (0)$ depending on $r_i \in S_j$, $(r_i \notin S_j)$.

For the SPP, the inequalities (3.14) become

$$\sum_{j=1}^{N} t_{ij}\xi_j = 1, \quad i = 1, 2, \ldots, M \tag{3.15}$$

4.2 Problem reduction

Due to the nature of the SCP it is often possible to make certain well-known *a priori* deductions and reductions. [6, 26, 27, 28, 30, 50, 51]

Thus:

(1) If for some $r_i \in R$, $r_i \notin S_j \; \forall j = 1, \ldots, N$, then r_i cannot be covered and the problem has no solution.

(2) If $\exists r_i \in R$ so that $r_i \in S_k$, and $r_i \notin S_j$, $\forall j \neq k$, then S_k must be in all solutions and the problem may be reduced by setting $R = R - \{r_i\}$ and $\mathscr{S} = \mathscr{S} - \{S_k\}$.

(3) Let $V_i = \{j | r_i \in S_j\}$. Then, if $\exists p, q \in \{1, \ldots, M\}$ with $V_p \subseteq V_q$, r_q may be deleted from R, since any set that covers r_p must also cover r_q, i.e. r_q is dominated by r_p.

(4) If, for some family of sets $\bar{\mathscr{S}} \subset \mathscr{S}$ we have $\bigcup_{S_j \in \bar{\mathscr{S}}} S_j \supseteq S_k$ and $\sum_{S_j \in \bar{\mathscr{S}}} c_j \leqslant c_k$ for any $S_k \in \mathscr{S} - \bar{\mathscr{S}}$, then S_k may be deleted from $\mathscr{S}$ since it is dominated by $\bigcup_{S_j \in \bar{\mathscr{S}}} S_j$.

Let us now assume that these reductions, if applicable, have been applied and the original SCP reformulated in its irreducible form.

4.3 A tree search algorithm for the SPP

As noted earlier, the SPP is closely related to the SCP, being essentially an SCP with an additional (no-overcovering) restriction. This restriction is advantageous when we attempt to solve the problem by a tree search method, since it may allow the early abandonment of potential branches of the tree. Therefore, we shall first discuss a tree search algorithm for solving the SPP, and then show how it can be extended to solve the SCP.

Simple tree search methods for solving the SPP were proposed by Garfinkel and Nemhauser [27] and Pierce [48], and are essentially as follows:

At the start of the process, we make up M "blocks" of columns, one block for each element r_k of R. The kth block consists of those sets of $\mathscr{S}$ (represented by columns) which contain element r_k but which do not contain any lower-numbered elements $r_1, \ldots, r_{k-1}$. Each set (column), therefore, appears in

exactly one block, and the totality of blocks can in general be arranged in tableau form as shown in Table 3.1 although some block(s) may be nonexistent in a particular problem.

During the course of the algorithm, blocks are searched sequentially, with block k not being searched unless every element r_i, $1 \leqslant i \leqslant k-1$, has already been covered by the partial solution. Thus, if any set in block k were to contain elements indexed less than k, the set would have to be discarded (at this stage) due to the no-overcovering requirement.

The sets within each block are arranged (heuristically) in ascending order of their cost and the sets are now renumbered so that S_j will from now on be used to mean the set corresponding to the jth column of the tableau.

TABLE 3.1. Initial tableau

	Block 1	Block 2	Block 3	Block 4	
r_1	1111	0			
r_2	0 or 1	1111	0		
r_3		0 or 1	1111	0	
r_4			0 or 1	1111	
.				0 or 1	
.					etc.
r_M					

At any stage of the tree search a current "best" solution $\check{B}$ of cost $\check{z}$ is known, where $\check{B}$ represents the family of corresponding covering sets. If B and z represent the corresponding values in the current state of the search, and E is a set representing those elements (i.e. rows) r_i covered by the sets in B, then a simple tree search algorithm proceeds as follows:

Initialization

Step 1. Set up the initial tableau and begin with the partial solution $B = \phi$, $E = \phi$, $z = 0$ and $\check{z} = \infty$.

Augmention:

Step 2. Find $p = \min [i | r_i \notin E]$. Set a marker at the top (lowest cost set) of block p.

Step 3. Beginning at the marked position in block p, examine its sets S_j^p, say, in increasing order of j.

(i) If set S_j^p is found such that $S_j^p \cap E = \phi$, and $z + c_j^p < \check{z}$ (where c_j^p is the cost of S_j^p), go to step 5.

(ii) Otherwise, if block p is exhausted or a set S_j^p reached for which $z + c_j^p \geqslant \check{z}$ go to step 4.

Backtrack

Step 4. B cannot lead to a better solution.

If $B = \phi$ (i.e. block 1 has been exhausted), terminate with the optimal solution $\check{B}$. Otherwise remove the last set, S_k^l say, added into B, put $p = l$, place a marker on set S_{k+1}^l, remove the previous marker in block l and go to step 3.

Test for new solution

Step 5. Update $B = B \cup \{S_j^p\}$, $E = E \cup S_j^p$, $z = z + c_j^p$.

If $E = R$ a better solution has been found. Set $\check{B} = B$, $\check{z} = z$ and go to step 4. Otherwise goto step 2.

As the search terminates with the exhaustion of block 1 (see step 4 above), it would seem worthwhile to arrange the blocks in ascending order according to the number of columns (sets) in each. This can be achieved by renumbering the elements (rows) $r_1, \ldots, r_M$ in increasing order of number of sets in S containing that element, before setting up the initial tableau.

Before showing how this SPP algorithm can be extended for the case of the SCP, we briefly mention some of the other methods proposed for solving the SPP. In [49], Pierce and Lasky give some modifications to the above basic algorithm, including subsidiary use of a linear program; while, in [45], Michaud describes another implicit enumeration algorithm which is based on a linear programming problem corresponding to the SPP, with the block structure given above being used in a secondary role.

Other algorithms involving simplex-type iterations have been proposed, both primal, [4, 5], and dual, [58, 37] whereas Jensen [37] has had success with a dynamic programming method for a certain type of SPP's.

4.4 A tree search algorithm for the SCP

The only step of the algorithm described in Section 4.3 above which is particular to the SPP, is step 3(i). If the "no-overcovering" requirement $S_j^p \cap E = \phi$ in that step were to be removed, then that algorithm could be used for the SCP. However, the setting up of the initial tableau would now have to be somewhat different from that shown in Table 3.2. Thus, let $S_j = \{r_{j_1}, r_{j_2}, r_{j_3}, \ldots\}$, with $j_1 < j_2 < j_3 \ldots$. It is not now sufficient to include S_j only in block j_1, because with the "no-overcovering" requirement removed,

it is not now possible to exclude S_j from consideration at, say, block j_2 if r_{j_1} has already been covered by the partial solution. Hence S_j must be entered once in each of the blocks $j_1, j_2, j_3, \ldots$. On the other hand, since an element r_{j_α} of S_j would, by the sequential nature of the search, be covered before branching on the sets of block β ($\beta > j_\alpha$) is considered, it is now possible to remove all the elements x_{j_α} from a set S_j when entering S_j into any block $\beta > j_\alpha$ without affecting the solution of the problem in any way. This apparently trivial removal is in fact computationally very beneficial, since the initial tableau of the SCP now set up can be reduced by the preliminary reduction tests of Section 4.2 to much smaller dimensions that would have been possible without the removal of the elements x_{j_α} ($j_\alpha < \beta$) from the sets of block β [48].

It should be pointed out here that the algorithm of Section 4.3 is a primitive tree search devoid of any sophistication, and although with careful coding it can be made quite effective for the SPP [27] it is not, as it stands, an efficient algorithm for solving SCP's of even moderate size. We will now discuss some dominance tests and the calculation of lower bounds which can be used to limit the tree search and improve the efficiency of the basic algorithm.

4.4.1 Some Dominance Tests. We will first illustrate these dominance tests by an example before stating the general results. Consider a case where block 1 contains (among others), sets $S_1^1 = \{r_1, r_4, r_6\}$ and $S_2^1 = \{r_1, r_3, r_4\}$ of costs 3 and 4 respectively, and where block 2 contains (among others), sets $S_3^2 = \{r_2, r_3, r_5\}$ and $S_4^2 = \{r_2, r_4, r_5\}$ each of cost 2.

During the course of the algorithm of Section 4.3 a stage will be reached where $B_a = \{S_1^1, S_3^2\}$, $E_a = \{r_1, r_2, \ldots, r_6\}$, $z_a = 5$; and we will branch forward from this stage until either we find a better solution than the current $\check{B}$, or we realize that S_1^1 and S_3^2 cannot both appear in an optimal solution.

Many steps later, we will reach a stage where

$$B_b = \{S_2^1, S_3^2\}, \qquad E_b = \{r_1, \ldots, r_5\}, \qquad z_b = 6.$$

At this point it becomes apparent that any further branching from this stage is unnecessary, since $E_b \subseteq E_a$ and $z_b \geqslant z_a$. A similar situation occurs when the stage is reached where

$$B_c = \{S_2^1, S_4^2\}, \qquad E_c = \{r_1, \ldots, r_5\}, \qquad z_c = 6.$$

Thus, it may be worthwhile to keep, for each value of $z = 1, 2, \ldots \check{z}$ some (perhaps incomplete) list of the maximal E's which have already been achieved for this z, (where by maximal is meant a set not included in another set also in the list). These lists of E's can then be used to limit the search by eliminating branches that later on prove fruitless. It is generally impractical, however, to save *all* the maximal sets E, at any cost level. In fact if this is done, the method would be similar to a Dynamic Programming approach to the

problem as a whole (or, alternatively, could be considered as a full breadth-first tree search).

Let us now assume that we have saved some list ($L(z_i)$ of previous sets E that have been attained during the course of the algorithm at a summed-cost level z_i. Suppose a stage is reached where $E = E'$, $B = B'$, $z = z'$, and we are about to examine block k (i.e. $k = \min\{i | r_i \notin E'\}$ at step 2 of the algorithm of Section 4.3), and to consider set S_j^k of cost c_j^k for the next branching. If $z' + c_j^k < \check{z}$, the original algorithm would have branched forward from this stage, and updated

$$E = E' \cup S_j^k, \qquad B = B \cup \{S_j^k\}, \qquad z = z + c_j^k,$$

regardless of any other considerations.

However, one could now also ensure (at step 3), that:

$$E' \cup S_j^k \nsubseteq E_l; \quad \forall E_l \in L(z_i), \quad \text{and for all } z_i, \quad z' < z_i \leqslant z' + c_j^k \qquad (3.16)$$

before branching forward. If the set S_j^k fails the above test, then it is rejected and instead the next set, S_{j+1}^k, of block k is considered, etc. If S_j^k passes test (3.16), one could branch forward on S_j^k, and continue as previously, after updating $L(z)$ by introducing the set $E' \cup S_j^k$ into $L(z' + c_j^k)$.

Since, as mentioned earlier, it is practically impossible to store the complete lists $L(z_i)$, some heuristic criterion has to be used to decide on the size of these lists, and how they should be updated during the search.

(*A*) *List size.* It is obviously better to eliminate sets S_j^k which are potential tree branchings at an early level in the tree, as this would discard larger portions of the potential search-tree. It would thus seem intuitively worthwhile to keep larger-sized lists $L(z_i)$ for the smaller z_i. This argument is further reinforced by the fact that test (3.16) is more likely to be effective when the sets T contain only a few elements, which is generally the case for small z_i-levels.

(*B*) *Updating of the Lists.* At a particular z_i-level, the current E-set (E' say) is more likely to be a subset of a set E'' (in the list $L(z_i)$), which was derived from the same general part of the search-tree. This is so because E' and E'' have in common a large section of the path from the start-node to their respective tree nodes. Thus, it would seem to be advantageous to arrange the lists $L(z_i)$ as stacks and to use a FIFO (first in, first out) policy, with the bottom-most set being discarded in the event of stack overflow.

4.4.2 THE CALCULATION OF A LOWER BOUND. At some stage of the search, given by B', E', z', and where block k is the next block to be considered, a lower bound h on the minimum value of z, can be calculated and used to

limit the tree search as follows:

Consider an uncovered element $r_i \in R - E'$ which does not appear in any of the sets of those blocks $k, k+1, \ldots, i-1$, corresponding to elements not yet covered by the partial solution. The element r_i cannot then be covered unless some set S^i_j of block i is chosen to add to B' at a future stage. Thus, for each such element r_i, construct a row for matrix $D = [d_{qs}]$ and a row for a second matrix $D' = [d'_{qs}]$, where d_{qs} is the number of elements in set S^q_s and d'_{qs} is its cost.

In addition, append an extra row, θ say, to D and D' with $d_{\theta s} = s$ for all $s = 0, 1, \ldots, M - |E'|$ and $d'_{\theta s} = s \cdot \min [c^i_j / |S^i_j|]$, the minimum being taken over all sets S^i_j with $r_i \notin E'$. Note that although the number of elements in a row q_1, of D (or D') may be different from that of another row q_2, we assume here that 0's and ∞'s are entered at the end of some rows of D and D' respectively, so that all rows have f (say) elements and the matrices become rectangular.

The following observations can now be made: since the optimal solution to the current subproblems must cover $M - |E'|$ elements, a choice of one entry from each row i of D, which satisfies

$$(d_{1s_1} + d_{2s_2} + \ldots + d_{\theta s_\theta}) \geqslant M - |E'|$$

and which minimizes the corresponding cost,

$$v = (d'_{1s_1} + \ldots + d'_{\theta s_\theta})$$

gives v as a lower bound to the optimal cost of the (covering) subproblem. Note that the assumption is made that the sets corresponding to the allocations in the rows of D are disjoint, which is obviously the best possible situation. The last row θ simply ensures that, if

$$\sum_{i=1}^{\theta-1} d_{is_i} < M - |E'|,$$

the remaining elements are covered in the best possible way, i.e. at minimal cost-per-additional-element-covered.

The minimum value of

$$\sum_{i=1}^{\theta} d'_{is_i} \quad \text{subject to} \quad \sum_{i=1}^{\theta} d_{is_i} \geqslant M - |E'|$$

can easily be derived by a dynamic programming algorithm as follows:

Let $g_\rho(v)$ be the maximum number of elements that can be covered using only the first ρ rows of D (i.e. only ρ of the blocks in the subproblem), and whose total cost does not exceed v. $g_\rho(v)$ can then be calculated iteratively as:

$$g_\rho(v) = \max_{s=1,\ldots,f} [d_{\rho s} + g_{\rho-1}(v - d'_{\rho s})], \tag{3.17}$$

where $g_0(v)$ is initialized to 0 for all v.

Determining func. toughest part of Dynamic Prging

Hence, the lowest value, v^*, of v, for which $g_\theta(v) \geqslant M - |E'|$ is then the required lower bound, h, and can be easily obtained from the dynamic programming tableau derived from the above iterative equation. One should note that only that range of v where $0 \leqslant v < \check{z} - z'$ need be considered, since if $h \geqslant \check{z} - z'$ (i.e. $g_\theta(v) < M - |E'|$ for $v \geqslant \check{z} - z'$), a backtracking step may be taken immediately.

4.5 Computational performance of the SCP algorithm

The computational performance of the algorithm described in Section 4.4 for the SCP is shown in Table 3.2 The numbers of rows and columns shown are those resulting after the reduction tests have been applied. All problems are randomly generated and have costs $c_j = 1$ for all columns j.

TABLE 3.2 Computing times of the SCP algorithm for random problems

Problem	Number of rows	Number of columns	Density	Computing time†	Nodes in tree search‡
1	15	20	0·22	0·05	0·02
2	15	25	0·25	0·04	0·01
3	15	25	0·30	0·06	0·02
4	20	60	0·18	0·25	0·07
5	20	50	0·20	0·11	0·02
6	20	75	0·23	0·35	0·05
7	25	110	0·17	0·60	0·06
8	25	120	0·19	1·30	0·16
9	25	170	0·22	1·50	0·11
10	30	80	0·12	1·00	0·20
11	30	180	0·15	10·50	1·60
12	30	250	0·17	6·00	0·52
13	30	340	0·21	9·20	0·30
14	30	475	0·20	11·00	0·09
15	30	500	0·23	22·00	0·40
16	30	700	0·23	105·50	3·02
17	30	725	0·25	32·50	0·12
18	30	875	0·26	57·20	0·18
19	30	1000	0·27	72·80	0·19
20	35	160	0·09	18·50	2·18
21	35	290	0·13	28·00	2·03
22	35	460	0·16	75·50	2·95
23	35	585	0·18	129·50	3·96
24	35	780	0·19	141·20	5·03
25	35	1075	0·20	110·00	0·55

† CDC 6600 seconds.
‡ Number of nodes generated by the tree (thousands).

Other methods of solving the SCP are given by Lemke, Salkin and Spielberg [41] who use a different type of tree search with an embedded linear program. The solution of this LP is used as a lower bound during the search and is also used in determining the next forward tree branching from the current node. Cutting-plane approaches, similar in principle to those used for general 0–1 programming [32] are given by House, Nelson and Rado [35] and by Bellmore and Ratliff [9]. A computational survey and comparison of these methods is given by Christofides and Korman [19].

5. Applications of the Covering Problem

5.1 Choice of interpreters

Let us assume that an organization wants to hire interpreters to translate from French, German, Greek, Italian, Spanish, Russian and Chinese into English, and suppose there are five prospective applicants A, B, C, D and E. Each applicant requires a certain salary and speaks only a subset of the above set of languages. What is then required is to find what candidates to hire in order to be able to translate from all the above languages into English with the least cost of hiring. This is quite obviously a covering problem.

If, for example, the salary requirements of all the applicants are the same and the languages they speak are as shown in the T matrix below, then the solution to the above problem is to hire persons B, C and D.

	Person →				
Language ↓	A	B^*	C^*	D^*	E
French	1	0	1	1	0
German	1	1	0	0	0
Greek	0	1	0	0	0
Italian	1	0	0	1	0
Spanish	0	0	1	0	0
Russian	0	1	1	0	1
Chinese	0	0	0	1	1

5.2 Information retrieval [21]

Let us suppose that a number of pieces of information are stored on N files of length $c_j = 1, 2, \ldots, N$ and that each piece is stored on at least one file. At some time a request for M pieces of information is made and these can be obtained by searching the files in a number of ways. In order to obtain all M pieces of information with the least length of files that have to be searched a SCP will have to be solved where an element t_{ij} of the matrix T is 1 if information i is on file j and 0 otherwise.

5.3 Airline crew scheduling [1, 55, 64, 65]

Suppose that the vertices of a nondirected graph G represent the places that an airline can visit and the arcs of G represent the flight "legs" that the airline must operate at a given time. Any path in this graph (satisfying a number of constraints that may be imposed in practice) forms a feasible flight schedule which can be operated. Let there be N such feasible schedules, the cost of the jth one being calculated in some way to be c_j. The problem of finding the least-cost set of schedules so that every flight leg is operated at least once is then a covering problem where the matrix T contains an entry of unity in position (i,j) if flight leg i is contained in schedule j, and zero otherwise.

A variant to this problem (which also appears often in aircrew scheduling) is derived by insisting that a flight leg can appear once and only once in the chosen set of schedules. This leads to a SPP corresponding to the SCP defined above.

5.4 Simplification of logical (boolean) expressions [30, 50, 51, 67, 44, 43]

When simplifying a logical expression E, which is (say), given in disjunctive normal form, one need only consider the set P, of prime implicants of E. A "simplest" form for E is then obtained by finding the lowest-cardinality subset $\check{P} \subseteq P$ subject to the condition that every (conjunctive) term of E be 'covered' by (i.e. by implied by) at least one prime implicant in $\check{P}$. This is obviously a SCP, where once more the column costs $c_j = 1$ for all $j = 1, \ldots, N$.

A problem equivalent to finding a minimal representation to logical expressions occurs in the design of optimal switching circuits.

5.5 Vehicle dispatching [7, 47, 24]

In the vehicle dispatching problem a graph represents a road network with vertices representing customers except one vertex which represents the depot. A vehicle leaves the depot, supplies goods to some customers and returns to the depot thus traversing a circuit. If the "cost" of circuit j is c_j (for example c_j could be the mileage of the circuit or the time to complete it), how many vehicles should be used on which circuits so as to supply all customers (once only) in any one day and minimize cost? This problem can obviously be considered as a SPP whose columns represent all possible feasible (because of the practical constraints), circuits starting and finishing at the depot. The rows represent the customers.

An almost identical problem involves the routing of a cable to carry electricity from a substation (depot) to customers in "ring-circuits" [15]. Other practical applications of the SCP or SPP include: assembly line balancing [59, 25], political districting [29], the calculation of bounds in general integer programs [18], network planning [20] and network attack or defence [8, 10].

5.6 Other graph covering problems

A large number of problems in graph theory can be formulated as SCP's although many of these problems could be solved more efficiently by alternative graph-theoretic means discussed in other parts of this book. The purpose of the present section is to show how some of these problems are related via the SCP.

5.6.1 THE MINIMUM DOMINATING VERTEX SET. As mentioned in Section 4 of the present Chapter, the problem of finding the minimum dominating vertex set of a graph G is a SCP with the T matrix being the transpose of the adjacency matrix of G with all diagonal entries set to 1.

5.6.2 THE MAXIMUM INDEPENDENT VERTEX SET. One way of finding the maximum independent vertex set of a graph $G = (X, \Gamma)$ is to generate all maximal independent vertex sets and choose the one of largest cardinality. Another way is to consider the problem as a SCP where columns of the matrix T represent vertices of G and rows represent links, with $t_{ij} = 1$ if vertex x_j is a terminal vertex of link a_i and 0 otherwise.

It is then apparent [23] that if $\check{X} \subseteq X$ is the set of columns in the minimum solution of this SCP, the set $X - \check{X}$ is the maximum independent vertex set of G. This is so because if $\check{X}$ is a set cover of the links of G, then every link has at least one of its terminal vertices in $\check{X}$ and hence two vertices in $X - \check{X}$ cannot be adjacent, i.e. $X - \check{X}$ is independent. Moreover, if $X - \check{X}$ is independent, each link can have at most one terminal vertex in $X - \check{X}$ and hence $\check{X}$ is a set cover of the links. Hence, the complement of every vertex set cover of the links of G is an independent set, and since $\check{X}$ is the minimum cardinality such cover $X - \check{X}$ is the maximum independent set.

5.6.3 THE MINIMUM COVERING AND MAXIMUM MATCHING (see Chapter 12). What is in the literature referred to as a *minimum "covering"* is a set E of links of a graph $G = (X, A)$ so that every vertex of G is a terminal vertex of at least one link in E and $|E|$ is minimum. Thus, since E could be considered as dominating the vertices of G, that E^* with minimum $|E|$ could—in the terminology used in this Chapter—be called a *minimum dominating link set*. A problem equivalent to that of finding a minimum covering is that of finding a set M of links of a graph G so that no two links in M are adjacent. M is called a matching and that set M^* with maximum cardinality is the *maximum matching*, which in the current termonology could also be called a *maximum independent link set*. That the maximum matching and minimum covering problems are equivalent is demonstrated in Chapter 12 dealing with matchings. It is shown there that if in a minimum covering E^* the degree of a vertex x_i is $d^{E^*}(x_i)$—considering only the links in E^*—then if for every x_i with $d^{E^*}(x_i) > 1$

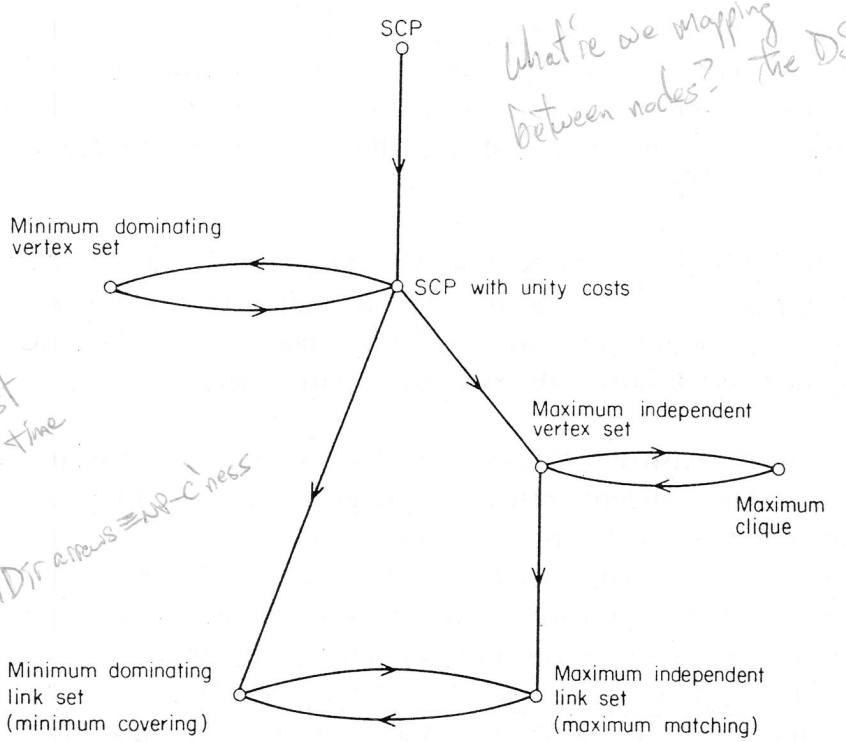

FIG. 3.5. Problem-solution implication diagram

a total of $d^{E^*}(x_i) - 1$ links incident at x_i are removed, the remaining set of links form a maximum matching. Inversely, if M^* is a maximum matching, and for every vertex x_i with $d^{M^*}(x_i) = 0$ a link incident at x_i is added, the resulting set of links is a minimum covering.

Maximum matchings and minimum coverings can be formulated as SCP's. In the case of the coverings, the columns of T represent links of the graph G, and the rows of T represent vertices, where $t_{ij} = 1$ if vertex x_i is incident with link a_j and 0 otherwise. For this case the matrix T is, therefore, the incidence matrix of G.

Figure 3.5 shows a problem-solution implication diagram, where an arc from problem α to problem β represents the fact that a solution to α implies the solution of β [34, 39]. The relation between maximum independent vertex sets and maximum cliques is via the complementary graph, and that between maximum independent vertex sets and maximum matchings is via the line graph of G. (See Chapter 10 for the definition of the line graph.)

5.6.4 COVERING A GRAPH WITH SUBGRAPHS. A whole family of problems involves the covering (or partitioning) of the vertices or links of a graph G with subgraphs or partial graphs of G having prescribed properties. In this

context the columns of the T matrix of the SCP (or SPP) represent all the subgraphs or partial graphs of G having the prescribed properties, and the rows of the matrix represent the vertices or links of G. In Chapter 4, for example, the problems of finding the chromatic number of a graph G is considered as a SCP with the rows of the T matrix representing vertices and the columns representing maximal independent vertex sets of G (i.e. maximal totally-disconnected subgraphs).

Other problems involve the covering of the links of G with partial graphs having diameter less than or equal to λ [13], covering the links of G with "star" trees or covering them with simple paths and circuits [16, 42].

6. Problems P3

1. Enumerate all maximal independent sets of the graph H shown in Fig. 3.6, and hence find the independence number $\alpha[G]$.

2. Use the method of Section 2.3.2 to list all maximal independent sets of the graph of Fig. 3.7. (Note graph symmetry.)

3. Show that for a given graph $G = (X, \Gamma)$, an independent set $A \subset X$ is maximum if and only if for any arbitrary independent set $B \subset X - A$ we have:

$$B \leqslant |\Gamma(B) \cup A|$$

(See Ref. [63].)

4. In the complete nondirected graph K_n show that each link is contained in exactly $(n - 2)$ circuits of cardinality 3(i.e. triangles).

5. It is easy to show that the converse of statement 4 above is not true, i.e. if every link in a graph G is part of $n - 2$ circuits of cardinality 3, G is not necessarily K_n.

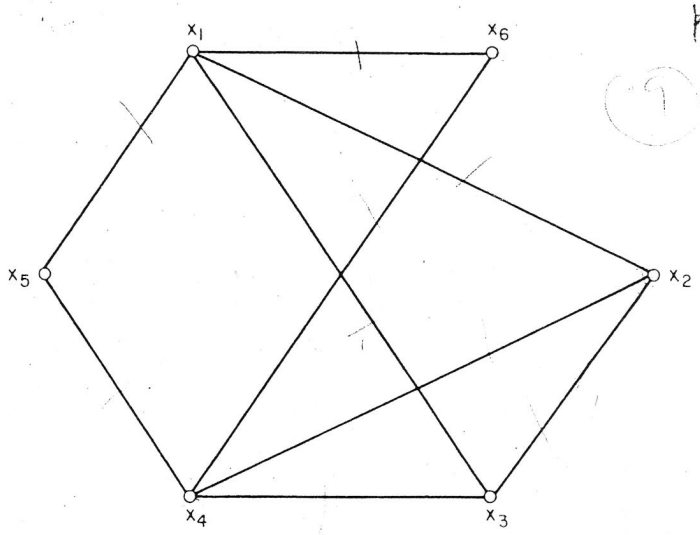

FIG. 3.6

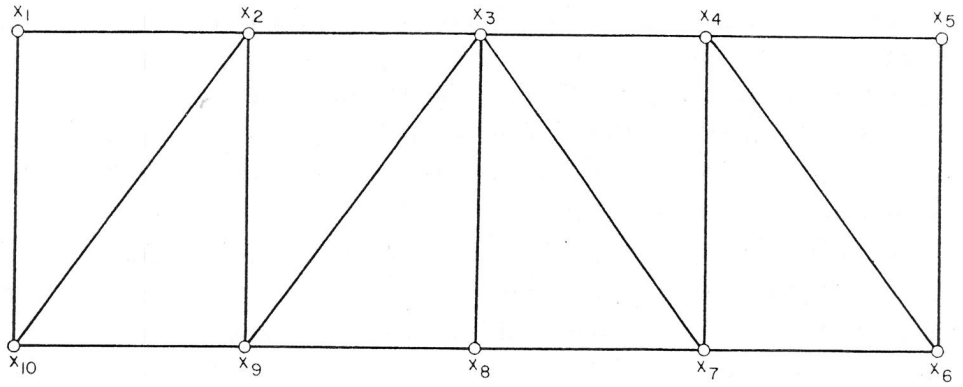

FIG. 3.7

(Figure 3.8 provides a counterexample in which each link is part of 2 such circuits but the graph does not contain K_4). However, one can use statement 4 above in order to find an upper bound on the clique number of a graph G—i.e. the largest possible K_r embedded in G—as follows:

(i) Start with a value of r which is a lower bound on the clique number.
(ii) Remove those links not part of at least $r - 2$ circuits of cardinality 3.
(iii) Repeat (ii) above for $r = r + 1$ etc until less than $r(r - 1)/2$ links are left. The current value of r is then an upper bound on the clique number.

Show that the above algorithm is correct, and apply it to the complementary graph $\tilde{G}$ of the graph G given in Fig. 3.7. Compare the answer with the value of $\alpha[G]$ found in problem 2.

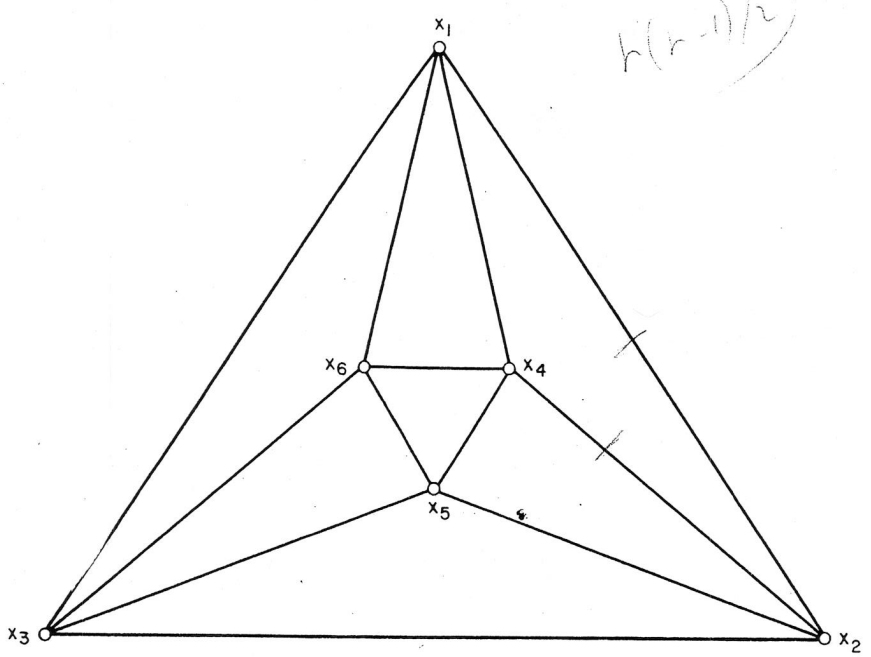

FIG. 3.8

6. A total of n symbols are to be used for transmitting information. A graph $G = (X, A)$ is given where link $(x_i, x_j) \in A$, if and only if symbol x_i could be confused with symbol x_j at the reception. The symbols are to be used in blocks of k-symbol "words". Show that the maximum number of words that could be used for transmitting information without the possibility of confusion, is the independence number $\alpha[G^k]$ of the graph $G^k = G \times G \times \ldots \times G$ (k times), where the product $G' \times G''$ of two graphs $G' = (X', A')$ and $G'' = (X'', A'')$ is the graph $H = (Y, B)$ given by

$$Y = \{x_i x_j | x_i \in X', x_j \in X''\}$$

and

$$B = \{(x_i x_j, x_k x_l) | \text{ either } x_i = x_k \text{ and } (x_j, x_l) \in A''$$
$$\text{or } x_j = x_l \text{ and } (x_i, x_k) \in A''$$
$$\text{or } (x_i, x_k) \in A' \text{ and } (x_j, x_l) \in A''\}$$

(see Ref. [61]).

7. Given any two graphs G' and G'' show that:

$$\alpha[G' \times G''] \geqslant \alpha[G']\,\alpha[G'']$$

8. (i) Show that in a nondirected graph G, $\alpha[G] \geqslant \beta[G]$, by showing that every maximal independent set is a dominating set.

(ii) Give an example to show that the minimum dominating set is not necessarily independent.

9. On a $k \times k$ chessboard show that it is impossible to place k queens which do not attack each other for the values $k \neq 4$. Find a solution for $k = 5$.

10. Moon and Moser [46] showed that the largest number of cliques ($f(n)$ say), that a graph with n vertices can have is given by:

$$\begin{cases} 3^{n/3} & \text{if } n = 0 \pmod 3 \\ 4 \cdot 3^{(n-4)/3} & \text{if } n = 1 \pmod 3 \\ 2 \cdot 3^{(n-2)/3} & \text{if } n = 2 \pmod 3 \end{cases}$$

and showed that the only graphs G which achieve $f(n)$ are the following:

(a) If $n = 0 \pmod 3$, then G consists of $n/3$ triples of vertices.

(b) If $n = 1 \pmod 3$, there are two possible classes of graphs: Either G consists of one quadruple and $(n - 4)/3$ triples, or it consists of two pairs and $(n - 4)/3$ triples.

(c) If $n = 2 \pmod 3$, then G consists of one pair and $(n - 2)/3$ triples.

In all cases a link (x_i, x_j) exists if and only if x_i and x_j belong to different groups, (pairs, triples, quadruples).

Verify the above statements for the cases of $n = 3$, $n = 4$ and $n = 5$.

11. Using the reduction tests of Section 4.2 eliminate as many rows and columns as possible from the covering problem whose $[c_j]$ and $[t_{ij}]$ matrices are given below. (Note: Apply reduction test 4 only for the case $|\bar{\mathcal{P}}| = 1$.)

$$[t_{ij}] = \begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 5 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 6 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

$$[c_j] = [\,4,\ 7,\ 5,\ 8,\ 3,\ 2,\ 6,\ 5\,]$$

12. Solve the SCP remaining after the reduction in problem 11 above using the method in Section 4.4.

7. References

1. Arabeyre, J. P., Fearnley, J., Steiger, F. C. and Teather, W. (1969). The airline crew scheduling problem: A survey, *Transp. Sci.*, **3**, p. 140.
2. Augustson, J. G. and Minker, J. (1970). An analysis of some graph-theoretical cluster techniques, *Jl. of ACM*, **17**, p. 571.
3. Balas, E. (1970). Project scheduling with resource constraints, *In*: "Applications of mathematical programming techniques", Beale, Ed., English Universities Press. London.
4. Balas, E. and Padberg, M. W. (1972a). On the set covering problem, *Ops. Res.* **20**, p. 1152.
5. Balas, E. and Padberg, M. W. (1972). On the set covering problem. II: An algorithm, Management Sciences Research Report No. 295, Carnegie-Mellon University.
6. Balinski, M. (1965). Integer programming: methods, uses, computation, *Man. Sci.*, **12**, p. 253.
7. Balinski, M. L. and Quandt, R. E. (1964). On an integer program for a delivery problem, *Ops. Res.*, **12**, p. 300.
8. Bellmore, M., Greenberg, H. and Jarvis, J. (1970). Multi-commodity disconnecting sets, *Man. Sci.*, **16**, p. 427.
9. Bellmore, M. and Ratliff, H. D. (1971). Set covering and involuntary bases, *Man. Sci.*, **18**, p. 194.
10. Bellmore, M. and Ratliff, H. D. (1971). Optimal defense of multi-commodity networks, *Man. Sci.*, **18**, p. 174.
11. Berge, C. (1962). "Theory of graphs", Methuen, London.
12. Bonner, R. E. (1964). On some clustering techniques, *IBM Jl. Res. and Dev.*, **8**, p. 22.
13. Bosák, J., Rosa, A. and Znám, Š. (1966). On decompositions of complete graphs into factors with given diameters, *In*: Int. Symp. on theory of graphs, Dunod, Paris, p. 37.
14. Bron, C. and Kerbosch, J. (1973). Algorithm 457—Finding all cliques of an undirected graph, *Comm. of ACM*, **16**, p. 575.
15. Burstall, R. M. (1967). Tree searching methods with an application to a network

design problem, *In*: "Machine Intelligence", Colin and Michie, Eds., Vol. 1, Oliver and Boyd, London.
16. Busacker, R. G. and Saaty, T. L. (1965). "Finite graphs and networks", McGraw-Hill, New York.
17. Chamberlin, D. D. (1971). The single assignment approach to parallel processing, *Proc. of AFIPS Conf.*, **39**, p. 263.
18. Christofides, N. (1971). Zero-one programming using non-binary tree search, *The Computer Jl.*, **14**, p. 418.
19. Christofides, N. and Korman, S. (in press). A computational survey of methods for the set covering problem, *Man. Sci.*
20. Crowston, W. B. (1968). Decision network planning models, Ph.D. Thesis, Carnegie-Mellon University.
21. Day, R. H. (1965). On optimal extracting from a multiple file data storage system: An application of integer programming, *Ops. Res.*, **13**, p. 482.
22. Desler, J. F. (1969). Degree constrained subgraphs, covers, codes and *k*-graphs, Ph.D. Thesis, Northwestern University, Evaston, Illinois.
23. Edmonds, J. (1962). Covers and packings in a family of sets, *Bulletin of American Mathematical Soc.*, **68**, p. 494.
24. Eilon, S., Watson-Gandy, C. D. T. and Christofides, N. (1971). "Distribution management: Mathematical models and practical analysis", Griffin, London.
25. Freeman, D. R. and Jucker, J. V. (1967). The line balancing problem, *Jl. of Industrial Engineering*, **18**, p. 361.
26. Garfinkel, R. S. (1970). Set covering; a survey, presented at XVII TIMS Conf., London.
27. Garfinkel, R. S. and Nemhauser, G. L. (1969). The set partitioning problem: set covering with equality constraints, *Ops. Res.*, **17**, p. 848.
28. Garfinkel, R. S. and Nemhauser, G. L. (1972). "Integer Programming", Wiley, New York.
29. Garfinkel, R. S. (1968). Optimal political districting, Ph.D. Thesis, The John Hopkins University.
30. Gimpel, J. F. (1965). A reduction technique for prime implicant tables, *IEEE Trans*, **EC-14**, p. 535.
31. Glover, F. (1971). A note on extreme point solutions and a paper by Lemke, Salkin and Spielberg, *Ops. Res.*, **19**, p. 1023.
32. Gomory, R. (1963). An algorithm for integer solutions to linear programs, *In*: "Recent Advances in Mathematical Programming", Graves and Wolfe, Eds., McGraw-Hill, New York.
33. Hakimi, S. L. and Frank, H. (1969). Maximum internally stable sets of a graph, *J. of Math. Anal. and Appl.*, **25**, p. 296.
34. Hakimi, S. L. (1971). Steiners' problem in graphs and its implications, *Networks*, **1**, p. 112.
35. House, R., Nelson, L. and Rado, T. (1965). Computer studies of a certain class of linear integer programs, *In*: "Recent Advances in Optimization Techniques", Lavi and Vogel, Eds., Wiley, New York.
36. Jardine, N. and Sibson, R. (1971). "Mathematical taxonomy", Wiley, London.
37. Jensen, P. A. (1971). Optimal network partitioning, *Ops. Res.*, **19**, p. 916.
38. Kapilina, R. I. and Shneyder, B. N. (1970). The problem of internal stability and methods of solving it on a universal digital computer, *In*: "Proc. of Scient. and Techn. Conf. on the results of scientific research in 1968–1969", Section on

Automation Computation and Measuring Procedures, Subsection on Graph Theory, Maskovskiy, Energticheskiy Institut Press.

39. Karp, R. M. (1972). Reducibility of combinatorial problems, *In*: "Complexity of computer computations", Miller and Thatcher, Eds., Plenum Press, New York, p. 85.
40. Lawler, E. L. (1966). Covering problems: Duality relations and a new method of solution, *Jl. of SIAM (Appl. Math.)*, **14**, p. 1115.
41. Lemke, C. E., Salkin, H. M. and Spielberg, K. (1971). Set covering by single branch enumeration with linear programming subproblems, *Ops. Res.*, **19**, p. 998.
42. Lovsaz, L. (1966). On covering of graphs, *In*: "Int. Symp. on theory of graphs", Dunod, Paris, p. 231.
43. Mayoh, B. H. (1967). Simplification of logical expressions, *Jl. of SIAM (Appl. Math.)*, **15**, p. 898.
44. McCluskey, E. J. Jr. (1956). Minimization of boolean functions, Bell. Syst. Tech. Jl., **35**, 1417.
45. Michadu, P. (1972). Exact implicit enumeration method for solving the set partitioning problem, *IBM Jl. of Res. and Dev.*, **16**, p. 573.
46. Moon, J. W. and Moser, L. (1965). On cliques in graphs, *Israel Jl. of Mathematics*, p. 23.
47. Pierce, J. F. (1967). On the truck dispatching problem—Part I, IBM Scientific Centre Technical Report 320–2018.
48. Pierce, J. F. (1968). Application of combinatorial programming to a class of all-zero-one integer programming problems, *Man. Sci.*, **15**, p. 191.
49. Pierce, J. F. and Lasky, J. S. (1973). Improved combinatorial programming algorithms for a class of all-zero-one integer programming problems, *Man. Sci.*, **19**, p. 528.
50. Pyne, J. B. and McCluskey, E. J. Jr. (1961). An essay on prime implicant tables, *Jl. of SIAM (appl. Math.)*, **9**, p. 604.
51. Quine, W. V. (1952). The problem of simplifying truth functions, *American Mathematical Monthly*, **59**, p. 521.
52. Roth, R. (1969). Computer solutions to minimum-cover problems, *Ops. Res.*, **17**, p. 455.
53. Roy, B. (1972). An algorithm for a general constrained set covering problem, *In*: "Computing and Graph Theory", Read, Ed. Academic Press, New York.
54. Roy, B. (1969, 1970). "Algebre moderne et théorie des graphes, Vol 1 and Vol 2, Dunod, Paris.
55. Rubin, J. (1971). A technique for the solution of massive set covering problems, with application to airline crew scheduling, IBM Philadelphia Scientific Center Technical Rept. No. 320-3004.
56. Rutman, R. A. (1964). An algorithm for placement of inter-connected elements based on minimum wire length, *Proc. of AFIPS Conf.*, **20**, p. 477.
57. Salkin, H. M. and Koncal, R. (1970). A pseudo dual all-integer algorithm for the set covering problem, Dept. of O.R. Technical Memo. No. 204, Case Western Reserve University.
58. Salkin, H. M. and Koncal, R. (1971). A dual algorithm for the set covering problem, Dept. of O.R. Technical Memo. No. 250, Case Western Reserve University.
59. Salveson, M. E. (1955). The assembly line balancing problem, *Jl. of Industrial Engineering*, **6**, p. 18.
60. Selby, G. (1970). The use of topological methods in computer-aided circuit layout, Ph.D. Thesis, University of London.

61. Shannon, C. E. (1956). The zero-error capacity of a noisy channel, *Trans. Inst. Elect. Eng.*, **3**, p. 3.
62. Shneyder, B. N. (1970). Algebra of sets applied to solution of graph theory problems, Engineering Cybernetics '70, English Translation by Plenum Press, p. 521.
63. Starobinets, S. M. (1973). On an algorithm for finding the greatest internally stable sets of a graph, Engineering Cybernetics '73, English Translation by Plenum Press, p. 873.
64. Steiger, F. (1965). Optimization of Swiss Air's crew scheduling by an integer linear programming model, Swissair Report O.R. SDK 3.3.911.
65 Steiger, F. and Neiderer, M. (1968). Scheduling air crews by integer programming, presented at IFIP Congress '68, Edinburgh.
66. Steinberg, L. (1961). The backboard wiring problem; a placement algorithm, *Jl. of SIAM (Review)*, **3**, p. 37.
67. Thiriez, H. (1971). The set covering problem—a group theoretic approach, Rev. Française d'Informatique et de Recherche Opérationnelle, **V-3**, p. 83.
68. Wells, M. B. (1962). Application of a finite set covering theorem to the simplification of boolean function expressions, *In*: "Information Processing '62", Popplewell, Ed., North Holland, Amsterdam.