REVUE FRANÇAISE D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. SÉRIE VERTE

HERVÉ THIRIEZ

The set covering problem : a group theoretic approach

Revue française d'informatique et de recherche opérationnelle. Série verte, tome 5, n° V3 (1971), p. 83-103

<http://www.numdam.org/item?id=RO_1971__5_3_83_0>

© AFCET, 1971, tous droits réservés.

L'accès aux archives de la revue « Revue française d'informatique et de recherche opérationnelle. Série verte » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

R.I.R.O.
(5e année, V-3, 1971. p. 83-104)

THE SET COVERING PROBLEM : A GROUP THEORETIC APPROACH

Hervé THIRIEZ (1)

Résumé. — *A solution technique for the general set covering problem is offered ; the approach is based on the group theoretic ideas developed by Gomory* [7] *and Shapiro* [19, 20]. *Important simplifications are allowed by the 0-1 nature of the problem ; they are such that, in many cases, a continuous linear programming package is the only tool required, even for the solution of problems with several hundred rows and several thousand columns.*

This paper contains three sections : in the first one, different solution techniques for the set covering problem are described ; this section may be skipped, as it is relatively independent of the other two. The second one presents an adaptation of the group theoretic approach to the solution of set covering problems. In the last section, the author descibes several programs adapting his technique to different configurations of set covering problems (size, type of cost vector, ...) and comments his computational experience.

The author wishes to express his gratitude to B. Roy, who made this paper possible by his helpful comments and his contribution to the redaction of the first section.

INTRODUCTION

The general set covering problem has the form :

Min $c \cdot x$

such that : $A \cdot x \geqslant b$ or $A \cdot x = b$

x Boolean (0-1)

where : A is an mxn Boolean matrix

b is an m-dimensional vector filled with 1's

c and x are n-dimensional vectors

Set covering formulations appear with problems of delivery, circuit design, scheduling, ... When $A \cdot x = b$, the problem is also called a « partitioning » problem [4]. The author of the current paper worked with particular emphasis on the aircrew scheduling problem [1, 21, 22], where :

(1) Professeur d'Analyse Opérationnelle au C.E.S.A.

— each row of A represents a flight from one city to another at a given date, day and hour;

— each column of A is a sequence of flights, a « rotation », that a crew flies before returning to its base airport : there is a « 1 » in each row corresponding to a flight covered by the rotation;

— the airline selects a set of rotations covering all the flights at a minimal cost.

The contraints may be inequalities : it is cheaper in some cases to send a second crew as passengers on one flight. When inequalities are used, it is advisable to modify the cost vector to prevent the situation from happening too often; e.g., add to each column cost a deadheading (sending a crew as passengers) cost multiplied by the number of 1's in the column.

1. SOLUTION TECHNIQUES

There is a limited number of basically different solution techniques; however, each of them offers a wide range of possible variations. This section describes the basic techniques, and tries to evaluate them.

1.1. Cutting plane methods

Cutting plane methods [6, 8, 10] use the following approach :

(1) Drop the « x Boolean » constraint, to obtain a continuous linear program.

(2) Solve the program. If the solution is integer, it is the optimal integer solution, terminate. Otherwise, go to (3).

(3) Add a constraint which will reduce the polyhedron defined by ($A \cdot x \geqslant b$ or $A \cdot x = b$; $x \geqslant 0$) without cutting off optimal integer solutions. Go to (2).

The constraint obtained in (3) is deduced from the linear constraints and the integrality constraints. Constraining x to be integer or Boolean is equivalent, since the cost vector is always nonnegative in set covering problems. Many good cutting plane algorithms have been created in the last few years; the drawback of the method is that, for most of the cutting plane methods, no integer solution is found until the optimal solution.

1.2. Branch and bound methods

Branch and bound methods [2, 11] are best illustrated by figure 1. A tree is created, where the origin node corresponds to the solution of the continuous LP (linear program) obtained by dropping the « x Boolean » constraint. Assume that, in that solution, variable x_j is not integer. Two new LPs are solved; x_j is fixed to « 0 » in the first one, and to « 1 » in the second. Two new nodes of the tree are thus defined. Clearly, at any moment, each node of the tree either is

a terminal node, or branches off to two other nodes. The branch and bound algorithm is explained after the figure.

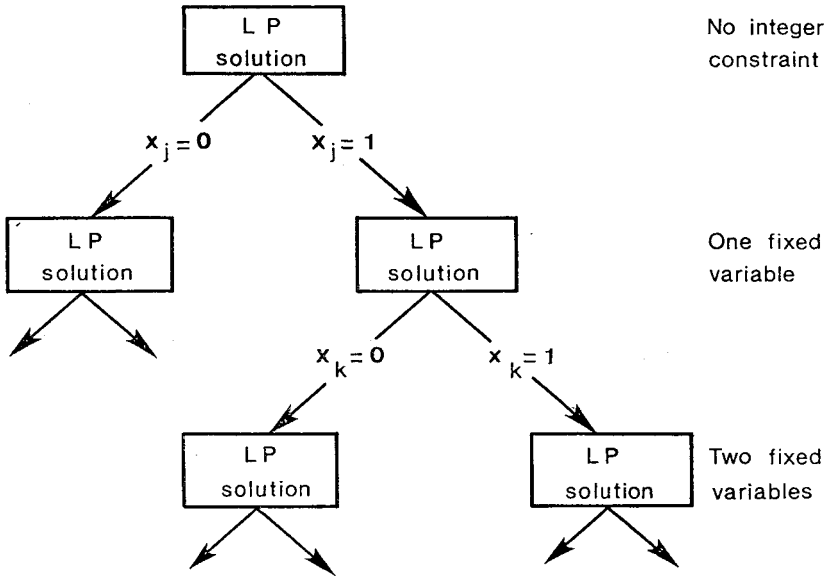

Figure 1.
Branch and Bound tree

Basic algorithm for branch and bound :

(1) Solve the continuous LP. If the solution is integer, terminate. Otherwise, create the origin node of the tree and go to (2).

(2) Select the cheapest terminal node : if its LP solution S is integer, it is an optimal integer solution, terminate. Otherwise, go to (3).

(3) Select a variable x_j not integer in S. Solve the two LPs corresponding to the continuous LP, plus the constraints S was subjected to, plus $x_j = 0$ (for the first one) and $x_j = 1$ (for the second one). Two new nodes have been defined. Go to (2).

Evidently, the optimal integer solution is eventually obtained, after a finite number of LP solutions. The drawbach of branch and bound is the necessity of keeping in memory all the relevant information on the tree and the solutions at each terminal node.

The author of the current paper obtained very good results with branch and bound, using Healy's reduction technique [9] : with the reduced costs and dual values in the LP results at a node, one may project lower bounds on the value of the objective function if a variable not integer in the solution were

pushed to 0 or to 1. In the same manner, one may project a lower bound on the value of the objective function if a variable at a level of 0 (resp. 1) in the solution were fixed to 1 (resp. 0) instead. The solution technique may then be :

(1) Use a heuristic method to find a good integer solution rapidly.

(2) Keeping this solution as a bound (until a better one is found), use the branch and bound approach as defined above, with the following modification : at each node, for the variables not yet fixed, project the bounds on the objective function corresponding to their 0 and 1 values; then :

a) for one variable, both lower bounds are more expensive than the best integer solution yet. No better solution may be obtained from this node of the tree. Select the next cheapest terminal node.

or *b*) for some variables, the lower bound on the objective function corresponding to their activity of 0 (resp. 1) is more expensive than the best integer solution yet. Fix them to 1 (resp. 0).

or *c*) all the lower bounds are cheaper than the best integer solution yet; take no specific action.

A problem of 104 rows and 236 columns was reduced right after its continuous LP solution to 26 rows and 63 columns with this technique (if only free variables and nonredundant constraints are considered). In that case, there was even no need for a branch and bound technique, since the LP solution obtained with the reduced problem was directly integer!

1.3. Implicit enumeration

The main contributions to the development of implicit enumeration must be acknowledged to Balas [3] and Geoffrion [5]. The idea is that, with n columns, there are 2^n possible integer solutions, usually too many for an exhaustive search, but not if many of them may be rejected in large blocks. The goal of implicit enumeration is to avoid the backtracking and memory management problems of branch and bound.

The method is described with the presentation used by Geoffrion [5].

First, a number of definitions must be made :

— A partial solution S is an assignment of binary values to a subset of the n variables.

— A free variable is a variable not assigned any value by S.

— A completion of a partial solution is a solution determined by S together with a binary specification of the values of the free variables.

— A partial solution is « fathomed » if all its completions have been considered implicitely or explicitly. (S, z) represents a partial solution S and its cost z.

— Notational convention : in S, j denotes $x_j = 1$ and $-j$ denotes $x_j = 0$.

Example : if $n = 4$ (4 variables), $S = (3, -4, 1)$ is a partial solution for which $x_1 = 1$, $x_3 = 1$, $x_4 = 0$ and x_2 is free. There are two possible completions : $S_1 = (3, -4, 1, 2)$ and $S_2 = (3, -4, 1, -2)$.

A sequence of partial solutions is generated and all their possible completions are considered. The best current feasible solution is stored together with its cost. Partial solutions are progressively completed. At each step, one of three situations arises :

a) a better feasible solution is found; it then replaces the current optimal solution S^*. The next partial solution, defined in the following, is then considered;

b) or it is clear that all completions of a partial solution will be infeasible or more expensive than S^*; go to the next partial solution;

c) or nothing can be said about S; assign a binary value to one of the free variables which therefore augments S. Test to find out whether. (*a*), (*b*), or (*c*) is now valid.

At some point, there will be no partial solution left to be considered. All solutions will have been implicitly or explicitly covered. The optimal solution is the final S^*.

The representation of S must be such that it is possible to recognize whether its other binary value has already been assigned to a given variable, the other variables being equal. For example, a variable will be underlined if the partial solution formed by the variables preceding it in S with their current value and the variable at its other binary value has already been fathomed. Example : $S = (3, -4, \underline{1})$ indicates that $(3, -4, -1)$ has already been fathomed.

The next partial solution is obtained by complementing the rightmost not underlined variable of S and dropping all elements to its right. To complement, underline the variable and assign to it its other binary value. The next partial solution of $(3, -4, 1)$ is $(3, \underline{4})$. It is clear that, through this procedure, the whole set of solutions has been fathomed when a partial solution has been evaluated for which all variables are underlined.

This procedure allows a complete search with a minimum of backtracking effort and table management. Computational speed is sacrificed for this advantage, since the progression in the three is determined by the choice of the initial solution.

1.4. Graph theory (B. Roy)

B. Roy offers in [17, vol. 2, section VI.B] a solution technique based on a graph theoretic representation of the problem. A tripartite graph is drawn, with an origin node z, a set of « column » nodes representing the columns of the A matrix, and a set of « row » nodes standing for its rows. There is an arc

linking z to each column node, and an arc links a column node j to a row node i whenever $a_{ij} = 1$ (where a_{ij} is the element in the i^{th} row and j^{th} column of matrix A).

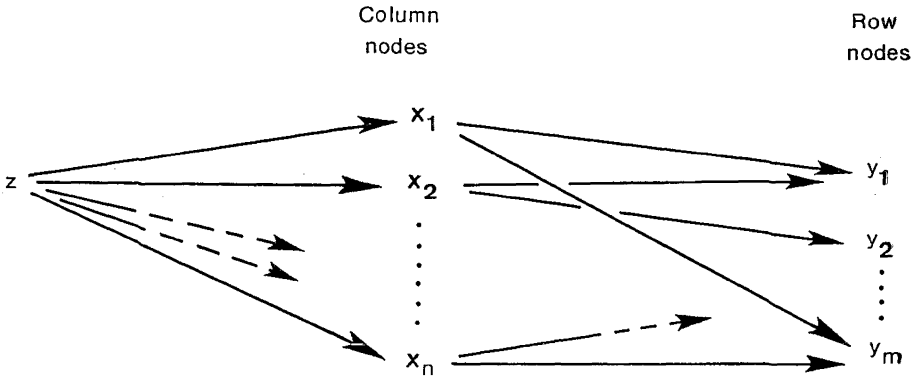

Figure 2.

Graphic representation

An « externally stable set » to a graph is a set of nodes such that each node in the graph belongs to it, or is the destination node of an arc for which the origin belongs to the e.s.s. (externally stable set).

There is an equivalence between an e.s.s. (formed by z and a subset of column nodes) of the graph and a covering set, i.e. a subset of columns of A covering all the rows.

Roy's technique is similar to an implicit enumeration process : a partial e.s.s. is first formed by z and all the column nodes which are the only ones to cover some of the row nodes; then, a column node which covers at least one new row node is added, ... At each step, the situations called *a*), *b*), *c*) in the implicit enumeration section appear. The difference between the two methods is mainly that :

— at each step, i.e. each time a new node enters the partial e.s.s., a reduction procedure takes place :

a) for each row node covered by only one remaining column node, enter the column node in the partial e.s.s.

b) the corresponding column nodes and the row nodes they cover are deleted from the graph;

— at each step, there is a non-inclusion test which finds out whether the addition of the column node to the partial e.s.s. makes it possible to reject one or more of its column nodes (the new column node may cover all the row nodes covered by another column node in the partial e.s.s., in the most simple case).

It must be emphasized that the reduction procedure and the non-inclusion test take place each time the partial e.s.s. is augmented by one element.

The procedure, as it is described by B. Roy, also has the property that it finds all covers costing less than a given value. This is useful when there are alternate objective functions, which may or may not be quantified.

The last property, and the most important, is that this approach can deal with additional constraints, the coefficients of which are not necessarily Boolean. A paper [18] will be published on that subject.

In conclusion, it can be said that, as with implicit enumeration, this approach is probably too slow for the large-sized general set covering problems. However, it must be considered as an important contribution to the solution of constrained set covering problems.

1.5. Heuristic methods

For a long time, heuristic methods [1, 4] have been used : they provide good solutions faster than most methods, but do not necessarily guarantee optimality. They are still used when people have problems too complicated to solve otherwise, or when they are ignorant of the existence of better methods.

These methods obtain feasible integer solutions based on a limited search of the feasible space; to find these solutions, selection criteria are applied which *should* bring one close to the optimal integer answer.

One typical heuristic method is to solve the continuous LP problem and manually round up the noninteger values in the solution vector so as to obtain a feasible integer answer.

Another heuristic, used by Air France, is to fix to « 1 » all the variables having an activity of « 1 » at the continuous optimum; the problem is reduced correspondingly, and an integer optimization code solves the remainder. Of course, this approach may result in non-optimal answers.

The author tested the group theoretic approach against heuristic methods used by several airlines : group theory was always faster (for the problems it could solve), usually at least twice as fast.

1.6. Other methods

There are methods which do not enter any of the preceding categories. Pierce [14] created a purely enumerative procedure to solve the partitioning problem; according to comparative testing made at M.I.T., it seemed to perform better than group theory for problems with few (less than 50) rows and several hundred columns; in these cases. Pierce's method was faster than the LP solution itself! Other enumerative schemes were developed by the airlines [1].

The drawback with most of these methods is their limitation, usually, to handling only the $A \cdot x \geqslant b$ or the $A \cdot x = b$ case.

An exception is the group theoretic approach, which is described in the following section.

2. THE GROUP THEORETIC APPROACH

Subsections 2.1 through 2.3 describe the group theoretic approach in its generality, i.e. where A, b and x are only constrained to be integer. Subsections 2.4 and on present the simplifications due to the Boolean nature of A, b and x.

2.1. General presentation

The group theoretic approach is based [7, 19, 20] on the transformation of the set covering problem from its canonical form into the « group theoretic » form.

Let us denote by N the set of nonnegative integer values : an element, a vector, or a matrix belonging to N is exclusively made up of nonnegative integer values.

The canonical form is the following :

$$\begin{aligned} &\text{Min } c \cdot x \\ (1) \qquad &\text{such that : } A \cdot x = b \\ &x \in N \end{aligned}$$

where : A is an $m \cdot (m + n)$-dimensional integer matrix;

c is an $(m + n)$-dimensional nonnegative integer cost vector : $c \in N$;

x is an $(m + n)$-dimensional vector;

b is an m-dimensional integer vector.

There are $m + n$ unknowns, since a unit matrix (with a high cost coefficient) is added to guarantee a feasible solution to (1), when the partitioning problem is solved. Otherwise, the additional variables are the slacks. The continuous LP problem, obtained by dropping the $x \in N$ constraint, is :

$$\begin{aligned} &\text{Min } c \cdot x \\ (2) \qquad &\text{s.t. } A \cdot x = b \\ &x \geqslant 0 \end{aligned}$$

The technique of the group theoretic method consists in first solving (2); starting from the continuous optimum, a new problem is derived. The optimal solution to that problem defines the optimal solution to (1) in relation to the continuous optimum.

2.2. Group theoretic formulation

Let B be the basis of the optimal solution to (2). A set of conditions therefore holds true :

> B is a nonsingular $m \cdot m$ matrix : B^{-1} is defined;
> $B^{-1}b \geqslant 0$ since it is the solution vector to (2);
> $\bar{c}_j = c_j - c_B B^{-1} a_j \geqslant 0 \; \forall \, j$, where a_j is the j^{th} column of A and c_B the part of the cost vector corresponding to the basic variables: the reduced cost vector is nonnegative at the optimum.

Let us partition A, x and c with respect to the columns in and out of B, in (1); the following formulation, equivalent to (1), is obtained :

$$\begin{aligned} &\text{Min } (c_B x_B + c_R x_R) \\ (1a) \qquad &\text{s.t.} \quad B x_B + R x_R = b \\ &\qquad\quad x_B, x_R \in N \end{aligned}$$

where $A = (B, R)$

$x = (x_B, x_R)$

$c = (c_B, c_R)$ allows us to retrieve formulation (1).

Solving for x_B, a new formulation is :

$$\begin{aligned} &\text{Min } c_B(B^{-1}b - B^{-1}Rx_R) + c_R x_R \\ &\text{s.t.} \quad x_R \in N \\ &\qquad\quad B^{-1}b - B^{-1}Rx_R \in N \end{aligned}$$

The reduced cost vector $\bar{c}_R = c_R - c_B B^{-1} R$ is introduced in the formulation:

$$\begin{aligned} &\text{Min } (c_B B^{-1} b + \bar{c}_R x_R) \\ (1b) \qquad &\text{s.t.} \quad B^{-1}b - B^{-1}Rx_R \in N \\ &\qquad\quad x_R \in N \end{aligned}$$

It is clear that (1 b) is still equivalent to (1). As we can see, the cost of the optimal solution to (1) is equal to the optimal cost for (2) plus the sum of the reduced costs of some nonbasic variables.

The two constraints in (1 b) mean :

— the difference vector between $B^{-1}b$ and $B^{-1}Rx_R$ must be nonnegative integer;

— x_R must be nonnegative integer.

The first of the two constraints may be expressed as the sum of two conditions, $C1$ and $C2$: the difference vector must be integer and must be nonnegative :

($C1$) $B^{-1}Rx_R = B^{-1}b$ (mod 1), i.e. modulo an integer vector

($C2$) $B^{-1}Rx_R \leqslant B^{-1}b$

Let $[a]$ be the largest integer vector such that $[a] \leqslant a$.

Since $x_R \in N$, ($C1$) is equivalent to :

$$(B^{-1}R - [B^{-1}R]) \cdot x_R = (B^{-1}b - [B^{-1}b]) \qquad \text{(mod 1)}$$

All the fractions disappear if both sides of the equation are multiplied by D, the determinant of B, since R, b and x_R are integer :

$$D \cdot (B^{-1}R - [B^{-1}R]) \cdot x_R = D \cdot (B^{-1}b - [B^{-1}b]) \qquad \text{(mod } D\text{)}$$

Let us define : $\alpha_j = D(B^{-1}a_j - [B^{-1}a_j])$

$$\alpha_0 = D(B^{-1}b - [B^{-1}b])$$

The group theoretic form is the following :

$$\text{(3)} \qquad \begin{aligned} &\text{Min } \bar{c}_R \cdot x_R \\ &\text{s.t. } \sum_{j \in J} \alpha_j \cdot x_j = \alpha_0 \qquad \text{(mod } D\text{)} \\ &x_j \in N \ \forall j \in J \end{aligned}$$

where J is the set of indices of the nonbasic variables. All the columns of R have indices belonging to J.

There are two differences between (1b) and (3) :

— the objective function of (3) does not include $c_B B^{-1}b$: it is a constant and therefore does not influence the optimization process;

— the nonnegativity constraints for the basic variables do not appear in (3).

The group theoretic method may now be deduced easily.

Algorithm

An optimal solution to (1) is obtained as follows :

a) Solve problem (2), obtain base B and cost $c_B B^{-1}b$;

b) Let $\bar{x}_R$ be a best solution to (3) for which :

$$B^{-1}R\bar{x}_R \leqslant B^{-1}b$$

Then, the solution defined by: $x = (B^{-1}b - B^{-1}R\bar{x}_R; \bar{x}_R)$ is an optimal solution to (1) and costs : $c_B B^{-1}b + \bar{c}_R\bar{x}_R$.

2.3. Optimization process

Clearly, finding the best solution to (3) for which $B^{-1}Rx_R \leqslant B^{-1}b$ may be a rather difficult problem. However, there are theorems in Algebra which greatly simplify the problem. It is not the purpose of this paper to go into their detail, especially since good introductory papers to group theory have been written [7, 19, 20]. These theorems may be found in [13], pages 261 through 282; it is more important here to state clearly their implications.

Proposition 1 : the α_j vectors generate an abelian group which has D elements.

This means that there are at most D different possible α_j columns. Moreover, the group is closed with respect to scalar multiplication, modulo D.

EXAMPLE: consider a problem with five rows where $D = 3$; pick up a nonzero α_j, e.g. $\alpha_4^T = (2\ 1\ 0\ 2\ 1)$ where α_j^T is the transpose of the α_j column. Then, proposition 1 implies that (2 1 0 2 1) or (1 2 0 1 2) or (0 0 0 0 0), i.e. α_4^T or $2 \cdot \alpha_4^T$ (mod 3) or $3 \cdot \alpha_4^T$ (mod 3) are the only possible types of α_j columns.

Proposition 1 is a consequence of the integer nature of the A matrix.

If D is not prime, there are two possibilities :

— all the α_j columns are obtained by the products modulo D of a « basic » column by integers from 1 to D. This situation is discussed in subsection 2.5,

— or, D may be « decomposed » : assume for example $D = 12$. If D decomposes into 2×6, there will be a basic vector for $D_1 = 2$ formed with 0's and 6's, and a basic vector for $D_2 = 6$ formed with the numbers 0, 2, 4, 6, 8, 10. Each α_j cay then be expressed two-dimensionally in terms of the basic vectors α^1 and α^2 : $\alpha_j = (k, p)$; i.e. $\alpha_j = k \cdot \alpha^1 + p \cdot \alpha^2$ (mod 12) where $k \leqslant 2$ and $p \leqslant 6$.

Proposition 2 : D always decomposes into the product $D = q_1 q_2 \ldots q_r$ such that : q_i divides $q_{i+1}\ \forall\ i \leqslant r - 1$.

So, if $D = 12$, it may only either react as though it were prime (only one basic vector) or decompose into $D = 2 \times 6$. It will never decompose into $D = 3 \times 4$.

The following procedure may be used to find the optimal solution to (3) :

a) Find how D decomposes itself : this may be done by visual inspection of the LP results and of some $B^{-1}a_j$ columns. Since the continuous LP problem must be solved anyway, it is cheaper to do that first, in order to avoid starting the group theoretic program if the solution is integer. The process may of course be automated.

b) Find r basic α^i vectors. Express each α_j in function of these vectors as an r-dimensional vector with values ranging respectively from 1 to q_1, 1 to q_2, ...

c) Find the cheapest combination of $\alpha'_j s$ producing α_0, modulo D.

Example : if $\alpha_0 = (1, 3)$ and $D = 12 = 2 \times 6$, any combination of $\alpha'_j s$ such that the sum of their expressions (in terms of the basic $\alpha^{i'} s$) equals (1, 3) modulo (2, 6) is a potential solution.

In the traditional approach to group theory, a graph is drawn where each node represents one of the D different α_j vectors. A shortest-path technique is used to solve the third part of the optimization procedure (*c*). When solving set covering problems, especially those arising in aircrew scheduling, the procedure is much simpler, since D is often prime, and nearly always a small number anyway. The following subsections describe the simplifications allowed by that property.

Up to this point, A was only required to be integer. In the remainder, only the solution of set covering problems is considered. Therefore, A will be exclusively made up of $0's$, $+1's$ and $-1's$. Consequently, « prime determinants », « cyclic » or « pseudo-cyclic » groups will appear in most problem solutions. Subsections 2.4 through 2.6 show the simplifications of the group theoretic approach allowed by these determinants and/or groups.

2.4. Prime determinant

If the determinant is prime, all the α_j columns are expressed as the product of the α_0 column, chosen as basic vector, by integers from 1 to D; the product is taken modulo D. Let k be the first row in which the basic vector has a nonzero value. Problem (3) becomes :

$$
\begin{aligned}
&\text{Min} \sum_{j \in J} \bar{c}_j x_j \\
(4) \qquad &\text{s.t.} \sum_{j \in J} \bar{\alpha}_j x_j = \bar{\alpha}_0 \qquad (\text{mod } D) \\
&x_j \in N \quad \forall j \in J
\end{aligned}
$$

where $\bar{\alpha}_j = \alpha_{kj}$: the element in the k^{th} row of α_j and where :

$k = \text{Min} \{ i / \alpha_{i0} \neq 0 \}$: the index of the first row where α_0 is nonzero.

The constraint set involving the α_j vectors has been reduced to a single constraint involving the $\bar{\alpha}_j$ values. All the nonbasic variables are now completely identified by their reduced cost and $\bar{\alpha}_j$. The solution to (4), and therefore to (3), is found with a simple inspection of the cost and the $\bar{\alpha}_j$ value for each nonbasic variable.

The solution to (1) being the best solution to (3) for which $B^{-1} R x_R \leqslant B^{-1} b$, it is easily found.

2.5. Cyclic groups

The abelian group determined by the α_j columns is said to be cyclic each time D decomposes into only one element ($r = 1$); that is, when D is prime or reacts as though it were. Then, even if D is not prime, the approach described in 2.4 may be used. It has been said that many groups for which D is not prime are still cyclic. This statement is not corroborated by the author's computational experience, as it is shown in figure 3. The computational experience the author will refer to in the following pages is derived exclusively from aircrew scheduling problems.

2.6. Pseudo-cyclic groups

The α_j columns are derived from the $B^{-1}a_j$ vectors, by definition. The elements of the $B^{-1}a_j$ vectors have a smallest common denominator $\bar{D}$. Clearly, $D = k \cdot \bar{D}$, where $k \in N$.

When $k \geqslant 2$, the determinant may be assumed to be only $\bar{D}$; if the group associated with $\bar{D}$ is cyclic, the group associated with D may be called « pseudo-cyclic ». In that case, the approach described in 2.4 may again be used.

Figure 3 shows the D and $\bar{D}$ values obtained on several problems so far. Each problem with a large D can be expected to have a definitely smaller $\bar{D}$; if $D > \bar{D}$, an alternate continuous LP optimum could probably be found for which has a determinant less than $\bar{D}$, and maybe even D. The smallest value that can possibly be found for D by considering bases of alternate continuous LP optima is the smallest common denominator of fractions appearing in $B^{-1}b$.

Airline	Size (rows × columns)	$\bar{D}$	D	Cyclic?
Air Canada : AC	74 × 739	1	2	Yes
Air France : AF	67 × 536	2	2	Yes
American Airlines : AA-I	104 × 132	2	2	Yes
American Airlines : AA-III	104 × 236	4	4	Yes
American Airlines : AA-II	104 × 236	7	7	Yes
United Airlines : UAL	117 × 4845	12	24	No
British European Airways : BEA-II	98 × 1652	21	126	No
British European Airways : BEA-I..	84 × 854	60	120	No

Figure 3.

D and $\bar{D}$ values for some test problems

2.7. Comments

In conclusion, one may remark that each time the group is cyclic or pseudo-cyclic, the inspection technique represented by (4) may be used. This is quite important : for most set covering problems, it will be the case; since the A matrix is Boolean, there is a high probability that D, or at least $\bar{D}$, be a prime number. If not, there remains the possibility that the group still be cyclic or pseudo-cyclic. The next section will describe the different solution techniques that may be used, as a function of the type of set covering problem (size, value of $\bar{D}$, cost vector, ...) being solved.

It must be remarked that, with the group theoretic method, it is very easy to continue the process and find the best 10 or 20 solutions, or all the solutions cheaper than a predetermined value. The marginal cost for doing it is neglectible : to generate the ten best solutions of a 300 $\times$ 2 000 problem rather than just the cheapest one is a matter of 20 more seconds on a 360/65.

3. SOLUTION TECHNIQUES. COMPUTATIONAL EXPERIENCE

All the programs described below are in Fortran IV, linked to the linear programming package of the IBM 360, i.e. MPS (Mathematical Programming System). The computer times were obtained with the M.I.T. 360/65 : the system had a 512 K core, with 256K directly available to the user.

3.1. Tree of solution techniques

Figure 4 shows the solution techniques proposed by the author for set covering problems, as a function of their characteristics.

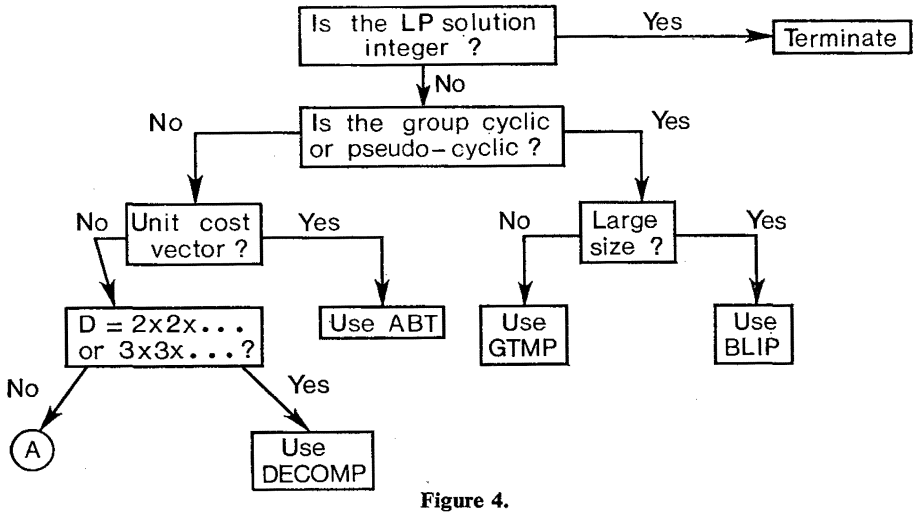

Figure 4.
Solution Techniques

GTMP, BLIP, DECOMP and ABT are the names of solution techniques described in subsections 3.2 through 3.5. Only GTMP and BLIP are completely programmed at this date; this did however not prevent the obtainment of computational experience, as it is shown hereafter.

Ⓐ is the only case in which no particular solution technique is offered by the author : the group is not even pseudo-cyclic, the cost vector is not unity and the determinant cannot be decomposed into $2x2x\ldots$ or $3x3x\ldots$ This situation is not very frequent in set covering probleps with sparse densities; the densities of problems discussed in this section, which are real-life aircrew scheduling problems, vary between 1 % and 5 %. There is one remedy, when Ⓐ happens : if the fractions in the LP solution vector have a small and/or prime smallest common denominator (smaller than $\bar{D}$), an alternate continuous optimum may be found for which the associated group is cyclic or pseudo-cyclic.

A good heuristic for finding such an alternate optimum is to replace as much as possible variables at a level of 0 in the basis by slack variables not yet in the basis. The density of 1's in the basis therefore decreases, since columns with several 1's are replaced by columns with only one « 1 ». Consequently, there is a reasonable chance that $\bar{D}$ becomes smaller.

If this is not sufficient to allow the solution by one of the four techniques proposed in figure 4 :

— either use the general group theoretic method as described by Shapiro in [19, 20];

— or use branch and bound until one of the four methods may be used, or an integer solution found, for the cheapest terminal node.

3.2. GTMP : Group Theoretic Method Program

GTMP solves the small and average size set covering problems for which the group is cyclic or pseudo-cyclic. For sizes over 200 rows or 1 000 columns, BLIP is preferable.

GTMP is called once the LP optimum has been obtained and starts by reading in all the $B^{-1}a_j$ columns corresponding to nonbasic variables.

A first phase of the program obtains the cheapest solution x_R to (4) such that $x_B = B^{-1}b - B^{-1}Rx_R \geqslant 0$ and $\sum_{j\in J} x_j = 1$, i.e. for which a feasible solution to (1) is derived by setting only one nonbasic column x_k to a level of 1 ($\bar{\alpha}_k = \bar{\alpha}_0$). A second phase looks for the cheapest solution to the same problem with $\sum_{j\in J} x_j = 2$ (instead of 1). The « optimal » integer solution is the better of the best solutions with $\sum_{j\in J} x_j = 1$ or 2.

Theoretically, combinations of three or more nonbasic columns should also be considered. Practically, it does not seem necessary : in all problems solved by the author to date with GTMP (or BLIP, for that matter) but one, the optimal integer solution was found in the first phase; in the last case, it was found in the second phase. If it were deemed necessary, optimality could easily be proved by adapting the program to consider all column combinations cheaper than the best one found by setting to « 1 » one or two nonbasics. It was not done, to save computer time during executions. This remark remains valid for the other solution techniques.

Figure 5 describes computational experience with GTMP. The problems were obtained from American Airlines and Air France. AA-I had been solved by the IBM SCA-I code on a similar 360/65 in 250 seconds. The Air France problem was run on a 360/75 with the Air France LP-heuristic code; the best solution was 6106 after 1.2 minutes.

Problem	AA-I	AA-II	AF
Type	$Ax \geqslant 1$	$Ax = 1$	$Ax = 1$
Size	104 × 132	104 × 236	67 × 536
Total Time	25.2 sec.	54.6 sec.	69 sec.
LP time	15.6 sec.	31.8 sec.	28.2 sec.
Time after LP	9.6 sec.	22.8 sec.	40.8 sec.
Integer Optimum	$z = 8\,820$	$z = 14\,145$	$z = 6\,049$
LP Cost	8 817.5	14 000.78	6 041.5

Figure 5.

GTMP

3.3. BLIP : Binary Linear Inspection Program

BLIP obtains the optimal solution of problems of moderate and large size, for cyclic or pseudo-cyclic groups. In a first phase after the LP optimization, BLIP finds the names of all the nonbasic variables for which the associated $\bar{\alpha}_j$ equals $\bar{\alpha}_0$. A second phase retrieves from MPS the $B^{-1}a_j$ columns corresponding to these nonbasics and finds the cheapest solution x_R to (4) such that $B^{-1}b \geqslant B^{-1}Rx_R$ and $\sum_{j \in J} x_j = 1$.

A third phase receives from MPS the $B^{-1}a_j$ columns of all the nonbasics with a reduced cost smaller than the cost of the solution to (4) previously selected. Using these columns, the best solution with $\sum_{j \in J} x_j = 2$ is found. Again, combinations of three or more columns are not considered. The process is very similar to the approach used in GTMP.

Figure 6 describes computational experience obtained before BLIP was programmed : the times correspond to the sum of the times of the $B^{-1}a_j$ column generations and LP restarts for each phase; the rest of the solution process was done by visual inspection of the listings. The computer time lost by restoring the LP solution at each phase is quite likely greater than the time saved by doing the inspection visually; the computer times listed in figure 6 may therefore be considered to be realistic computer times for BLIP. As a matter of fact, two large problems were solved by BLIP after it was programmed : once the intermediate printouts used for testing the program were deleted, the computer times turned out to be definitely faster than those obtained by visual inspection (since in that case, there were long listings, and the listings proved to be by far the most time-consuming part of the program).

3.4. DECOMP : Decomposition Technique

A frequent situation in large aircrew scheduling problems is when the determinant, or $\bar{D}$, decomposes into $2x2x2x\ldots$ or into $3x3x3x\ldots$ Assume for example that all the fractions in the LP solution and in the $B^{-1}a_j$ columns are halves and that each $B^{-1}a_j$ has fractions in none or just one of three subsets of rows; then, we may consider that $\bar{D} = 2 \times 2 \times 2$: the BLIP approach may be used with the difference that at least one nonbasic covering each subset of rows must be chosen to build a solution.

Problem	Type	Size	LP Time	Time after LP	Method used
UAL	$Ax = 1$	118 × 4 845	20 min.	1 min. 45 sec.	BLIP
AA-S1	$Ax \geqslant 1$	527 × 2 800	191 min.	7.28 min.	BLIP
AA-S2	$Ax \geqslant 1$	497 × 2 909	75 min.	14.62 min.	DECOMP
AA-S3	$Ax \geqslant 1$	1 138 × 3 533	150 min.	16.99 min.	DECOMP
AA-S4	$Ax \geqslant 1$	124 × 7 134	65 min.	14.89 min.	BLIP

Figure 6.

BLIP — DECOMP

3.5. ABT : Automatic Branching Technique

ABT was designed for the solution of problems with a unit cost and a non-(pseudo)-cyclic group. Airlines paying a fixed salary to their crews (e.g. Swissair) have a cost vector made of 1's : their objective is the minimization of the number of crews. If the group obtained after the LP solution is (pseudo)-cyclic, GTMP or BLIP, depending on the size, may be used.

Assume a non-(pseudo)-cyclic group : with a unit cost, it would be impractical to use branch and bound to solve the problem; since all the columns have practically the same eligibility, and since there is usually a large $\frac{\text{number of columns}}{\text{number of rows}}$ ratio, braching on $x_j = 0$ leaves the problem almost unchanged. The author tried branch and bound on problem BEA-II : after seven branches in the direction $x_j = 0$, the cost of the solution was still the same as that of the continuous LP optimum.

Since crew scheduling problems, and many other set covering problems, have a very flat objective function, there is a definite likelihood that many optimal integer solutions exist which cost $[z^o + .999]$ where z^o is the cost of the continuous LP optimum and $[a] = \text{Max } \{ i/i \in N, i \leqslant a \}$.

The ABT approach is described in figure 6 :

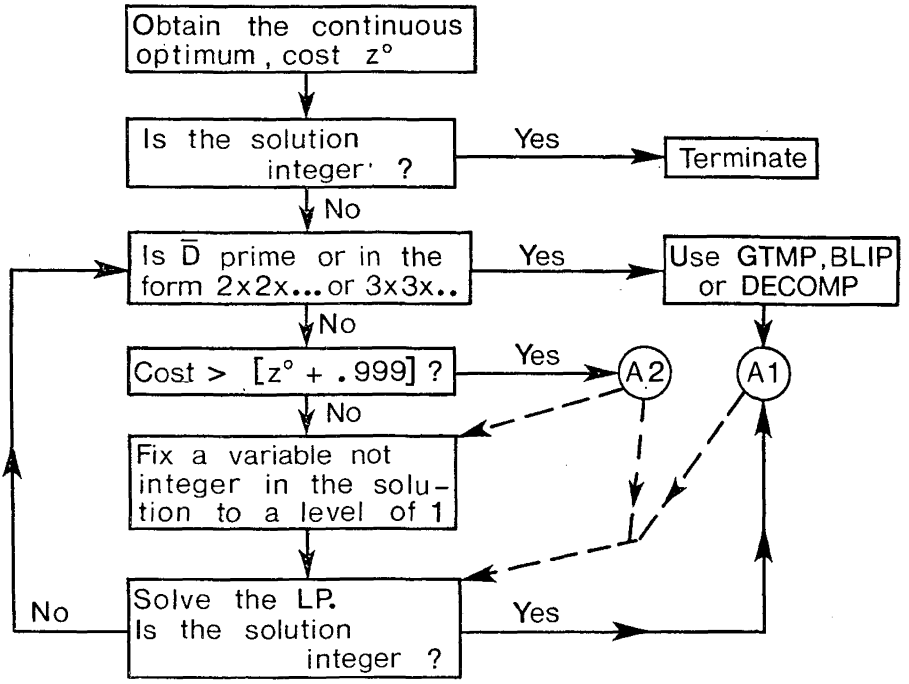

Figure 6.

Automatic Branching Technique

A1 : an integer solution has been found :

— it costs $[z^o + .999]$: accept it as the « optimal integer solution ». Most

likely, it will be optimal; this is not however certain, if z^0 is no more the cost of the continuous optimum (see A2);

— otherwise, if the solution did cost less than that integer value before the last LP solution, there is a good chance that a cheaper solution exists. Select the first variable x_j used for branching in the tree for which only the branch $x_j = 1$ was created : fix $x_j = 0$ and continue the process as before, from that higher point in the tree.

A2 : the cost passes the bound, but no integer solution is found :

— all the $\bar{D}'s$ obtained so far are relatively large : there may be no solution cheaper than the integer bound; set $z^0 = z^0 + 1$ and continue the branching;

— at least one $\bar{D}$ is small : there may be cheaper integer solutions. As in the second step of A1, return to the beginning of the tree and create the first possible $x_j = 0$ branch; then, continue as before.

Again, a program guaranteeing optimality could easily be written. It would only require that all the terminal nodes of the tree cost more than $[z^0 + .999]$ before the value of z^0 is changed : the branches of the type $x_j = 0$ would also have to be generated exhaustively. This modification would of course be quite expensive in terms of computer time.

Only two problems were provided to the author for which ABT could be used. In both cases, the solution time was more than satisfactory compared to what it was using BEA's program based on the House, Nelson and Rado [10] technique. For example, the best solution found for BEA-II using it was 28, after 12.24 minutes of Univac 494!

Problem	Type	Size	LP Time	Time (*) After LP	Integer Solution
BEA-I	$Ax = 1$	84×854	1.57 min.	2.06 min.	23
BEA-II	$Ax = 1$	$98 \times 1\,652$	3.29 min.	6.55 min.	26

Figure 7.
ABT Computational Experience

As with the BLIP computer times, these figures were obtained with visual inspection, the program beeing unwritten. However, it is certain that the time needed by the computer to scan the solution vector, find a fraction and set it to one will be much smaller than the time needed to regenerate the preceding LP optimum each time a variable is fixed to 1.

(*) Includes the time needed to restart the preceding LP optimum before each branching and solve the LP after each branching.

The two integer solutions found were optimal. The objective fuction for set covering problems with a unit cost vector is so flat that, in most cases, several integer solutions exist which cost no more than $[z^o + .999]$. ABT is therefore a much better method than it would seem at first glance.

CONCLUSION

This paper has presented a set of solution techniques based on group theory for the solution of set covering problems. The major drawbacks of the proposed approach are the following :

— it does not have the capacity of handling constraints with non-Boolean coefficients, since the determinant values would then become prohibitive;

— there is a percentage of problems (very small, however) which may not be easily solved by the method : see *A* in figure 4;

— unless specifically required, optimality is not proved; it is however very likely : there is no counterexample to date, with real-life problems.

The advantages of the method are :

— it is very efficient in terms of speed : the bottleneck in terms of solution time is the time needed to obtain the continuous optimum, when large problems are solved. The ratio (total solution time/continuous LP solution time) decreases with size;

— the marginal cost of generating alternate integer optima or solutions close to the optimum is neglectible.

BIBLIOGRAPHY

[1] J. P. ARABEYRE, J. FEARNLEY, F. STEIGER and W. TEATHER, « The airline crew scheduling problem : a survey », *Transportation Science*, 3, n° 2, May 1969.

[2] P. BERTHIER, PHONG TRUAN NGHIEM et B. ROY, « Programmes linéaires en nombres entiers et procédures S.E.P. », *METRA*, vol. 4, n° 3, 1965.

[3] E. BALAS, « An additive algorithm for solving linear programs with 0-1 variables », *Operations Research*, 13, n° 4, 1965.

[4] R. S. GARFINKEL and G. L. NEMHAUSER, « The set partitioning problem : set covering with equality constraints », *Operations Research*, 17, pp. 848-856, 1969.

[5] A. M. GEOFFRION, An Improved Implicit Enumeration Approach for Integer Programming, The Rand Corporation, RM-5644-PR, June 1968.

[6] F. GLOVER. A Multiplan Dual Algorithm for the 0-1 Integer Programming Problem. Case Institute of Technology, Management Science Report n° 25, 1965.

[7] R. E. GOMORY, « On the relation between integer and noninteger solutions to linear programs », Proc. Nat. Acad. Sci., 53, pp. 260-295, 1965.

[8] R. E. GOMORY, « An Algorithm for Integer Solutions to Linear Programs », pp. 269-302 in *Recent Advances in Mathematical Programming*, Graves, R. L., and Wolfe, P., (Eds.), McGraw-Hill, 1963.

[9] W. C. HEALY Jr., « Multiple choice programming », *Operations Research*, 12, pp. 122-138, 1964.

[10] R. W. HOUSE, L. D. NELSON and T. RADO, « Computer Studies of a Certain Class of Linear Integer Problems », *Recent Advances in Optimization Techniques*, Lavi, A., and Vogl, T., (Eds.), Wiley, 1966.

[11] A. H. LAND and A. G. DOIG, « An automatic method of solving discrete linear programming problems », *Econometrica*, 28, pp. 497-520, 1960.

[12] E. L. LAWLER and M. D. BELL, « A method for solving discrete optimization problems », *Operations Research*, 14, nº 6, 1966.

[13] G. D. MOSTOW, J. H. SAMPSON and J. P. MEYER, Fundamental Structures of Algebra, McGraw-Hill, 1963.

[14] J. F. PIERCE, « Application of combinatorial programming to a class of all-zero-one integer programming problems », *Management Science*, 15, pp. 191-209, 1968.

[15] R. ROTH, « Computer solutions to minimum cover problems », *Operations Research*, 17, pp. 455-466, 1969.

[16] B. ROY et R. BENAYOUN, « Programmes linéaires en variables bivalentes et continues sur un graphe (Programme Poligami) », *METRA*, vol. 6, nº 4, 1967.

[17] B. ROY, Algèbre Moderne et Théorie des Graphes, Dunod éd., 1970.

[18] B. ROY, An Algorithm for a General Constrained Set Covering Problem, Computing and Graph Theory (To be published), Academic Press, New York.

[19] J. F. SHAPIRO, « Dynamic programming algorithms for the integer programming problem-I : the integer programming problem viewed as a knapsack type problem », *Operations Research*, 16, 1968.

[20] J. F. SHAPIRO, « Group theoretic algorithms for the integer programming problem-II : extension to a general algorithm », *Operations Research*, 16, nº 5, 1968.

[21] H. M. THIRIEZ, Implicit Enumeration Applied to the Crew Scheduling Problem, Dept. of Aeronautics, M.I.T., 1968.

[22] H. M. THIRIEZ, Airline Crew Scheduling : a Group Theoretic Approach, Ph. D. Thesis, M.I.T. FTL-R69-1, 1969.