On a Program for Ray-Chaudhuri's Algorithm for a Minimum Cover of an Abstract Complex*

By DOMINIQUE C. FOATA

University of North Carolina, Chapel Hill, N. C.

I. Introduction

The present paper shows how Ray-Chaudhuri's Algorithm [1] has been programmed. A program written in IT language and a detailed flow chart are available at the Computation Center of the University of North Carolina [2].

As Ray-Chaudhuri [1] showed in his paper, this algorithm may be used in various fields of mathematics and engineering. For instance, it can be used in many problems of coding, and also in the construction of block designs and projective geometries. More generally, it is applicable in all the fields in which the notion of minimum covering occurs.

II. Definitions

Let us define c-cover, and minimum c-cover. Let X be a set of m points: $X = \{x_1, x_2, \cdots, x_m\}$, and $\mathfrak{A}$ a class of n subsets of X: $\mathfrak{A} = \{A_1, A_2, \cdots, A_n\}$. Such a couple $(X, \mathfrak{A})$ is called a *complex.*

Assume that to each point x_j of X is assigned a weight c_j, c_j a positive integer, $j = 1, 2, \cdots, m$.

Then a subclass $\mathfrak{A}_1$ of $\mathfrak{A}$ is called a *c-cover of X* if each point x_j of X is contained at least in c_j sets of $\mathfrak{A}_1$; or still, if each point x_j of X is "covered" at least with c_j sets of $\mathfrak{A}_1$ $(j = 1, 2, \cdots, m)$.

Moreover, *a minimum c-cover* of X is naturally defined as a c-cover with a minimum number of sets.

Ray-Chaudhuri's algorithm consists of finding a minimum c-cover, starting with some c-cover.

III. Description of the Algorithm

The algorithm is described in the attached rough flow chart (Figure 1). We start with a c-cover $\mathfrak{A}_1$ of the complex $(X, \mathfrak{A})$. $\mathfrak{B}_1$ is the complement of $\mathfrak{A}_1$: $\mathfrak{B}_1 = \mathfrak{A} - \mathfrak{A}_1$.

As shown in [1], the algorithm consists of looking for *alternating chains*, i.e. sequences of distinct sets of $\mathfrak{A}$: $A_1, B_1, A_2, B_2, \cdots$ where the A's belong to $\mathfrak{A}_1$, and the B's to $\mathfrak{B}_1$ and where at each iteration i the set A_i satisfies the property $\mathfrak{A}_i(F_{i-1})$ and the set B_i, the property $\mathfrak{B}_i(D_i)$, $(i > 1)$. These two properties will be explained later.

After each choice of A_i, it is tested whether the class

$$\mathfrak{A}_1 - \left\{\bigcup_{j=1}^{i} A_j\right\} \cup \left\{\bigcup_{j=1}^{i-1} B_j\right\}$$

is a c-cover. If yes, we have a new c-cover whose cardinality is less, since

$$\mathfrak{A}_1 - \left\{\bigcup_{j=1}^{i} A_j\right\} \cup \left\{\bigcup_{j=1}^{i-1} B_j\right\}$$

is the class obtained from $\mathfrak{A}_1$ by dropping the i sets A of the chain and adding the $(i-1)$ sets B of the chain.

Hence, we can start again with this new c-cover. If a chain contains such an A, it is called reducible (with respect to the c-cover $\mathfrak{A}_1$).

The main theorem of Ray-Chaudhuri which implies the algorithm is:

$\mathfrak{A}_1$ is a minimum c-cover if and only if there is no reducible chain with respect to $\mathfrak{A}_1$.

Hence we must make an exhaustive search of reducible chains. The flow chart (Figure 1) shows how all the alternating chains are built and checked to be reducible. If there is no reducible chain, we come out with a minimum c-cover.

At each iteration i, a subclass $\mathfrak{A}_{i+1}$ of $\mathfrak{A}_1$ is defined by induction:

$$\mathfrak{A}_{i+1} \leftarrow \mathfrak{A}_i - \{A_i\}. \qquad \text{(box 5)}$$

or in other words, the class $\mathfrak{A}_{i+1}$ is the class $\mathfrak{A}_1$ in which all

* This research was supported in part by the United States Air Force, Office of Scientific Research of the Air Research and Development Command, under Contract No. AF (638)-213.

the sets A, up to i, of the alternating chain have been dropped. This definition is necessary because of the nonrepetition of an element in the chain. In the same manner, as appears in the flow chart,

$$\mathfrak{B}_i \leftarrow \mathfrak{B}_{i-1} - \{B_{i-1}\}.$$

The criteria to stop a chain are:

(1) After an A_i, the class

$$\mathfrak{A}_1 - \left\{\bigcup_{j=1}^{i} A_j\right\} \cup \left\{\bigcup_{j=1}^{i-1} B_j\right\} = \mathfrak{A}_{i+1} \cup \left\{\bigcup_{j=1}^{i-1} B_j\right\}$$

is a c-cover (or D_i, the set of points not c-covered with $\mathfrak{A}_{i+1} \cup \{\bigcup_{j=1}^{i-1} B_j\}$, is empty), and we start again with a smaller c-cover.

(2) After an A_i, the chain is not reducible at this stage and we test whether we can continue the chain by finding a $B_i \in \mathfrak{B}_i(D_i)$, i.e. a set of $\mathfrak{B}_i$ having a nonempty intersection with the set D_i.

If there does not exist such a B_i, we have to pick, if possible, another A_i from $\mathfrak{A}_i(F_{i-1})$. This is what is meant by the successive boxes:

$$\mathfrak{A}_i \leftarrow \mathfrak{A}_i - \{A_i\}; \quad \mathfrak{A}_i \neq 0; \quad i \leftarrow i-1, \; \mathfrak{A}_{i+1}(F_i) = 0?$$

If there is no remaining set of pick from $\mathfrak{A}_i(F_{i-1})$, and $i > 1$, we have to go back to the set $\mathfrak{B}_{i-1}$ and choose another B_{i-1}.

This corresponds in the flow chart to the path:

$$\mathfrak{A}_i = 0; \quad i > 1; \quad i \leftarrow i-1; \quad \mathfrak{B}_i \leftarrow \mathfrak{B}_i - \{B_i\}; \quad \textbf{go to } 9$$

Finally, if $\mathfrak{A}_i = 0$ and $i = 1$, all the possibilities have been examined and there is no reducible chain. We then out with "C is a minimum c-cover".

(3) After a B_i, a set F_i is then defined as the set of points of X which are covered with more sets A than sets B (of the chain, up to i).

We can continue the chain if there is in $\mathfrak{A}_{i+1}$ a set, call it A_{i+1}, having a non-null intersection with F_i. If there is one, we add it to the tentative chain (statement $A_{i+1} \in \mathfrak{A}_{i+1}(F_i)$), and we continue (**go to** 5). If there is not, we must change, if possible, the preceding B_i and test again. This corresponds to the statements:

$$\mathfrak{A}_{i+1}(F_i) = 0 \quad \text{or} \quad F_i = 0; \quad \mathfrak{B}_i \leftarrow \mathfrak{B}_i - \{B_i\}; \quad \textbf{go to } 9$$

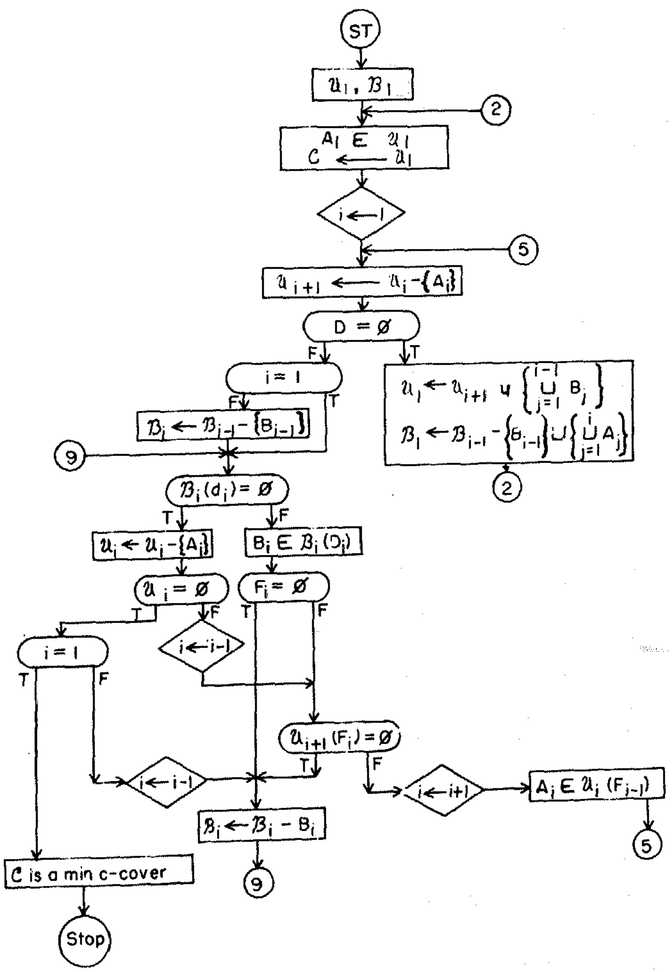

FIG. 1

IV. Programming of the Algorithm

The technique used in programming has been the following. The incidence matrix of the complex $(X, \mathfrak{A})$, i.e. a matrix A with n rows and m columns has been used:

$$A = (a(i, j)),$$

$$i = 1, 2, \cdots, n; \quad j = 1, 2, \cdots, m$$

in which $a(i, j) = 1$ if the point x_i belongs to the set A_j; and 0 otherwise. The rows correspond to the sets and the columns to the points.

Thus, a c-cover $\mathfrak{A}_1$ of the complex will be represented by a set of $k \leqq n$ rows of that matrix, such that each column-sum of this submatrix is greater than or equal to the constant c corresponding to this column.

Starting with a c-cover containing k sets, the k values are stored in $r_{11}, r_{12}, \cdots, r_{1k}$. This forms the class $\mathfrak{A}_1$ and the $(n-k)$ values of $\mathfrak{B}_1$ are stored in $s_{11}, s_{12}, \cdots, s_{1,n-k}$. We keep in memory all the sets $\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_k$ and $\mathfrak{B}_1, \mathfrak{B}_2, \cdots, \mathfrak{B}_{n-k}$ needed for the iteration. Thus, to each $\mathfrak{A}_i$ we reserve $(k - i + 1)$ variables:

$$r_{ii}, r_{i,i+1}, \cdots, r_{ik}$$

and, to each $\mathfrak{B}_i$, $(n - k - i + 1)$ variables:

$$s_{ii}, s_{i,i+1}, \cdots, s_{i,n-k}.$$

In the program, r_{ii} and s_{ii} $(i = 1, 2, \cdots)$ have been reserved for the sets A_1 and B_i of a chain. Under these conventions, the three criteria are:

(1) $D_i = 0$?
$a(r_{i+1,i+1}, j) + a(r_{i+1,i+2}, j) + \cdots + a(r_{i+1,k}, j) + a(s_{11}, j) + a(s_{22}, j) + \cdots + a(s_{i-1,i-1}, j) \geqq C_j$ for all $j = 1, 2, \cdots, m$?

(2) $\mathfrak{B}_i(D_i) \neq 0$?
Does there exist a j and q $(j = 1, 2, \cdots, m;$ $q = i, i+1, \cdots, n-k)$ such that condition (1) is not satisfied *and* $a(s_{iq}, j) = 1$?

(3) $\mathfrak{A}_{i+1}(F_i) \neq 0$?
Does there exist a j and q $(j = 1, 2, \cdots, m; q = i+1, i+2, \cdots, k)$ such that the two following conditions hold:
$a(r_{11}, j) + a(r_{22}, j) + \cdots + a(r_{ii}, j) - [a(s_{11}, j) + a(s_{22}, j) + \cdots + a(s_{ii}, j)] < 0$ and $a(r_{i+1,q}, j) = 1$.

V. Discussion

Using this method of incidence matrices, we can save a large amount of memory, since the incidence matrix only contains 0 and 1's. In the program presently written, each entry of the matrix occupies a register, i.e. a 36-bit location. Hence, we could examine bigger matrices by storing 36 entries of the matrix in a same register. This is a minor change to the program and a simple subroutine can be added to it to take care of this change.

As a conclusion, I would like to mention the great difficulty in handling relations like: *a set A belongs to a class* $\mathfrak{A}$. To program it, with the present basic operations of the computer, it requires quite many statements.

REFERENCES

1. Ray-Chaudhuri, D. K., An algorithm for a minimum cover of an abstract complex. Institute of Statistics, University of North Carolina, Mimeo Series 275, Feb. 1961. (Submitted for publication.)
2. Automatic programming using the IT Compilers for the Univac 1105 and IBM 650. Computation Center, University of North Carolina, Chapel Hill, N. C., Oct. 1960.

APPENDIX

Detailed Flow Chart and IT Program for Ray-Chaudhuri's Algorithm

In the main paper, the algorithm has been described together with the techniques used in programming.

In this appendix we give in more detail the techniques which are more involved with the use of IT language.

First, the variable assignment has been the following:

N1 = m (number of columns—of points)
N2 = n (number of rows—of sets)
N3 = k (number of sets in the initial c-cover)
N4 = q
N5 = w (number of sets in $\mathfrak{B}_1$)
N6 = j (index for the columns)
N7 = i (index for the iteration)
N8 = t
N9 = λ (number of sets in $\mathfrak{A}_1$)
N10 = base for the r_{ij}
N11 = base for the s_{ij}
N12 = base for the u_{ij}
N13 = base for the f_i
N14 = base for the v_i
N15 = base for the e_i
N16 = base for the d
N17, N18, N19 = (various temporary variables)
N20 = γ
N21 = P
N(21+1) $\cdots$ N(21+N3) = ($i_1, i_2, \ldots, i_k$—the values of the sets in $\mathfrak{A}_1$)
N(21+N3+1) $\cdots$ N(21+N2) = ($i_{k+1}, \ldots, i_n$—the values of the sets in $\mathfrak{B}_1$)

Remark. This number 21 is the smallest possible. If we need more N variables by adding some new statements to the program, it would be easy to replace 21 by another number. All the other N variables would be shifted.

N(N10 + N4 + ((N7 − 1) × N9) − ((N7 × (N7 − 1))/2)) = r_{iq} $(i \geq q)$ N4 = q N7 = i
N(N11 + N4 + ((N7 − 1) × N5) − ((N7 × (N7 − 1))/2)) = s_{iq} (N5 = w)

Note: There are $\lambda(\lambda + 1)/2$ variables r_{iq} and $w(w + 1)/2$ variables s_{iq}.

N(N12 + N7) = u_i (index of $\mathfrak{A}_i$)
N(N13 + N7) = f_i (index of columns within $\mathfrak{A}_i$)
N(N14 + N7) = v_i (index of $\mathfrak{B}_i$)
N(N15 + N7) = e_i (index of columns within $\mathfrak{B}_i$)
N(N16 + N7) = d_i (each new c-cover is stored in these d_i variables)

Note: Using the matrix notation, the index of the columns N6 ranges from 0 to N1 − 1. On the flow chart, j ranges from 1 to m. Also, using the matrix notation, the entries of the incidence matrix have been expressed in floating point.

The m constants, $c_1, c_2, \cdots, c_m$ are stored in: $Z0, Z1, \cdots, Z(N1 - 1)$. $a(i, j) = YM(i, j)$; i.e., $a(i, j)$ ranges from $Y0$ up to $Y(N1 \times N2 - 1)$. The auxiliary variables, $S_1, S_2, \cdots, S_{j-1}, \cdots, S_m$ are assigned $Z(N1 + 0), Z(N1 + 1), \cdots, Z(N1 + N6), \cdots, Z(N1 + N2 - 1)$.

The INPUT of the program must contain:

N1, N2	(n and m)
Z0, Z1, $\cdots$, Z(N1 − 1)	(the c's)
Y0, Y1, $\cdots$, Y(N1 × N2 − 1)	(the incidence matrix)
N3	(k the number of sets of the c-cover we start with)
N22, N23, $\cdots$, N(21 + N3)	(the sets of the initial c-cover)
N(21 + N3 + 1), $\cdots$, N(21 + N2)	(the sets of the initial $\mathfrak{B}_1$)

The program is available on a magnetic tape labeled: FOATA R149—6/28/61 for the Univac 1105 digital computer, Computation Center, University of North Carolina. This program was first submitted as a term project for the course Mathematics 169—"The Theory and Practice of Digital Computers", given at the University of North Carolina at Chapel Hill.