NASA CR-
147551

SET COVERING ALGORITHM, A SUBPROGRAM OF THE SCHEDULING ALGORITHM FOR MISSION PLANNING AND LOGISTIC EVALUATION

JOB ORDER 81-197

```
(NASA-CR-147551)  SET COVERING ALGORITHM, A              N76-22225
SUBPROGRAM OF THE SCHEDULING ALGORITHM FOR
MISSION PLANNING AND LOGISTIC EVALUATION
(Lockheed Electronics Co.)   79 p HC $5.00               Unclas
                                    CSCL 22A G3/12       27397
```

Prepared by

Lockheed Electronics Company, Inc.
Aerospace Systems Division
Houston, Texas

Under Contract NAS 9-12200

For

MISSION PLANNING AND ANALYSIS DIVISION

National Aeronautics and Space Administration
LYNDON B. JOHNSON SPACE CENTER
Houston, Texas
March 1976

LEC-6842

SET COVERING ALGORITHM, A SUBPROGRAM OF THE SCHEDULING ALGORITHM FOR MISSION PLANNING AND LOGISTICS EVALUATION

Job Order 81-197

PREPARED BY

Han Chang

H. Chang
Support Software Section

APPROVED BY

[illegible]

J. P. Davis, Supervisor
Support Software Section

F. N. Barnes

F. N. Barnes, Manager
Dynamic Systems Department

Prepared By

Lockheed Electronics Company, Inc.

For

Mission Planning and Analysis Division

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

March 1976

LEC-6842

TECHNICAL REPORT INDEX/ABSTRACT
(See instructions on reverse side.)

1. TITLE AND SUBTITLE OF DOCUMENT SET COVERING ALGORITHM, A SUBPROGRAM OF THE SCHEDULING ALGORITHM FOR MISSION PLANNING AND LOGISTICS EVALUATION	2. JSC NO. JSC-09[illegible]50
3. CONTRACTOR/ORGANIZATION NAME Lockheed Electronics Company, Inc.	4. CONTRACT OR GRANT NO. NAS 9-12200
5. CONTRACTOR/ORIGINATOR DOCUMENT NO. LEC-6842	6. PUBLICATION DATE (THIS ISSUE) March 1976
7. SECURITY CLASSIFICATION Unclassified	8. OPR (OFFICE OF PRIMARY RESPONSIBILITY) R. S. Davis
9. LIMITATIONS GOVERNMENT HAS UNLIMITED RIGHTS [✓ YES] NO IF NO, STATE LIMITATIONS AND AUTHORITY	10. AUTHOR(S) H. Chang
11. DOCUMENT CONTRACT REFERENCES	12. HARDWARE CONFIGURATION
WORK BREAKDOWN STRUCTURE NO. NA	SYSTEM NA
CONTRACT EXHIBIT NO.	SUBSYSTEM
DRL NO. AND REVISION	MAJOR EQUIPMENT GROUP
DRL LINE ITEM NO.	

13. ABSTRACT

This program uses Lemke, Salkin and Spielberg's (ref. 1) Set Covering Algorithm (SCA) to optimize a traffic model problem in the Scheduling Algorithm for Mission Planning and Logistics Evaluation (SAMPLE). SCA forms a submodule of SAMPLE and provides for input and output, subroutines, and an interactive feature for performing the optimization and arranging the results in a readily understandable form for output.

14. SUBJECT TERMS

CONTENTS

TABLES

FIGURES

SET COVERING ALGORITHM, A SUBPROGRAM OF THE SCHEDULING ALGORITHM FOR MISSION PLANNING AND LOGISTICS EVALUATION

1. INTRODUCTION

This documentation provides a general description of the Set Covering Algorithm (SCA) computer program, and includes functional specifications, functional design and flow, and a discussion of the program logic. The SCA program is a submodule of the Scheduling Algorithm for Mission Planning and Logistics Evaluation program (SAMPLE) and has been designed as a continuation of the Mission Payloads (MPLS). The MPLS uses input payload data to form a set of feasible combinations which are collections of payloads that meet certain system constraints (e.g., Shuttle weight-to-orbit capability); from this combination set the SCA selects a subset with minimum cost such that all payloads are contained without redundancy. The subset of feasible combinations is called a traffic model. To date, the program has had two main uses:

a. To provide input data for the Operations Simulation and Resource Scheduling (OSARS) submodule, and

b. To provide the user a tutorial option so that he can choose an alternate traffic model in case a particular traffic model cannot be scheduled by the OSARS.

The SCA program was begun in 1974 with an input of less than 50 payloads per year and the assumption of unity cost coefficients in the objective function. Presently, this program has been expanded to solve 100 payloads per year with an option of five different performance criteria in the objective function. Since a general SCA program solves a problem with inequality constraints, an appropriate change has been made to convert it to solve the traffic model problem, which has all equality constraints.

2. PROGRAM DESCRIPTION

2.1 DEFINITIONS AND SYMBOLS

2.1.1 DEFINITIONS

Basis	- A basis for a n-dimensional euclidean space, E^n, is a linearly independent subset of vectors from E^n which spans the entire space
Canceled variable	- A variable is said to be canceled if its value is set to be zero
Ceiling	- The lowest upper bound
Decision variable	- A zero-one variable which corresponds to a feasible mission
Dominated column	- In a matrix A or $\{A_j\}$ $j = 1, n$ A_j is said to be dominated by A_i if $A_j \leq A_i$ for $i \neq j$. A_j is called a dominated column of A_i and A_i is the dominating column of A_j.
Extreme point	- The corner points of a convex polyhedron.
Fractional variable	- The nonintegral variable in the linear programming solution
Free variable	- A variable which has not been fixed at any value
Level	- The number of variables fixed at one
Minimum-cost-per-constraint-satisfied rule	- In the sub-subproblem, for each variable the value of its cost coefficient divided by the number of constraints is calculated. The variable with the minimum of such value is set to one first.
Occurrence table	- A table for each payload of all feasible combinations that include that payload
Partial solution	- A set of variables fixed at one or zero

Preferred row	- The row containing the smallest number of ones in the current subproblem's constraint matrix
Preferred set	- The set of columns in the constraint matrix which contains the entry of one in the preferred row
Slack variable	- In general, it is desirable to convert any inequalities in the constraints into equations which are much more convenient to work with in the linear programming problem. The conversion is carried out by introducing some additional variables which are called slack variables.
Subproblem	- The problem contains only free variables with certain constraints deleted from the main problem by the satisfaction of the partial solution
Sub-subproblem	- The problem contains the columns of A associated with the positive fractional variables and the rows corresponding to the constraints they satisfy
Upper bound	- A known value that the solution of the Set Covering Problem will not exceed

2.1.2 SYMBOLS

A	An m×n matrix with zeros and ones as elements; constraint matrix
c	An m×1 matrix of all positive numbers; cost coefficient vector
e	An m×1 matrix of all ones
F	Indices of free variables
I	An m×m identity matrix
i	The row index of matrix A
j	The column index of matrix A
ℓ	Level counter

M	A nonnegative number in the objective function, set to be very large to solve a linear programming problem with equality constraints; in this program, set $M = 500$
m	Number of constraints in the set covering problem
n	Number of decision variables in the set covering problem
s	An $m \times 1$ matrix of slack variables
y	The column vector of zero-one variables
z	Current value of objective functional, i.e., $c'y$
z^*	Upper bound for $c'y$
$\alpha = 1(0)$	Reaching a point on a forward (backward) step
γ	Iteration counter
$\eta(\ell,j)$	The number of constraints of the level ℓ subproblem variable j can satisfy
$\xi(\ell)$	Lower bound for the minimal functional of level ℓ subproblem
σ	Sequence vector, recording nonzero components of y
τ	Tolerance parameter about z^*
$\phi(\ell)$	Number of constraints in the current (level ℓ) subproblem

2.2 GENERAL DESCRIPTION

2.2.1 PROGRAM CAPABILITIES

The SCA program optimizes a traffic model problem over an objective function which consists of a set of user-selected performance criteria such as Orbital Maneuvering System (OMS) weight, load factor, or Shuttle cargo bay utilization for each feasible combination. This problem also includes a set of constraints which assures no redundancy of payloads in the traffic model. For application, a set covering algorithm developed by C. E. Lemke, H. M. Salkin, and K. Spielberg (ref. 1) has been adopted. The main advantage of this algorithm is that it permits a rather efficient and simple solution procedure that is basically a (zero, one) branch and bound search logic coupled with linear programming (LP) and suboptimization techniques. The suboptimization technique can construct very good integer solutions from the solutions to LP subproblems.

The formulation of a traffic model problem can be exactly fitted into the mathematical model of a Set Covering Problem (SCP). Before a formulation example is presented, a general understanding of the form of this model will be helpful.

The SCA solves the SCP which has the form:

$$\begin{array}{ll} \text{min} & c'x + Me's \\ \text{subject to} & Ax - Is = e,\ s \geq 0 \\ \text{and} & x_j = 0 \text{ or } 1 \end{array}$$

When this model is applied to the traffic model problem, x_j is taken as a decision variable on a particular feasible combination j (or flight j). Flight j is considered to be chosen when $x_j = 1$, otherwise $x_j = 0$. For each payload, there is a correspondent constraint which insures the nonredundancy of this payload in the traffic model. Vector c stores the performance criteria for all of the feasible missions; M is set to a large value to assure that constraints are satisfied. The application of SCA to the traffic model can be easily demonstrated by the following example.

Assume that three payloads have to be launched in a particular month. After checking all of the possible combinations, only five are considered to be candidates. These candidate flights are called feasible combinations; a summary of the payloads they carry is given in the following table.

Feasible combination (j)	Payloads
1	No. A
2	No. A, No. B
3	No. B, No. C
4	No. B
5	No. C

Assume that associated with each feasible combination j, there is a cost factor c_j. The problem is to formulate a mathematical model for determining the traffic model which gives the minimum cost. In this example, c_j is set to unity, which implies that we are seeking a traffic model which consists of the minimum number of feasible combinations. Let x_j $(j = 1, 2, 3, 4, 5)$ be the decision variable over the selection of a particular feasible combination j in the traffic model. A unity value of x_j implies that feasible combination j has been selected to be a member of the traffic model. Since cost has been chosen as a measure of effectiveness, the object is to minimize

$$z = x_1 + x_2 + x_3 + x_4 + x_5$$

subject to the restrictions developed in the following paragraphs.

The constraint in this situation is that the same payload cannot be contained in more than one feasible combination in the traffic model. The mathematical statements of the restrictions for three payloads are

A: $$x_1 + x_2 = 1$$

B: $x_2 + x_3 + x_4 = 1$

C: $x_3 + x_5 = 1$

Finally, there are the binary restrictions, i.e., $x_j = 0$ or 1. Therefore, in summary, the mathematical model for this problem is the following. Minimize

$$z = x_1 + x_2 + x_3 + x_4 + x_5$$

subject to:

$$\begin{aligned} x_1 + x_2 \qquad\qquad\qquad &= 1 \\ x_2 + x_3 + x_4 \qquad &= 1 \\ x_3 \qquad + x_5 &= 1 \end{aligned}$$

and $x_j = 0$ or 1, $j = 1, 2, 3, 4, 5$

This formulation can also be written in a matrix form:

$$\begin{array}{ll} \text{min} & c'x + Me's \\ \text{subject to} & Ax - Is = e,\ s \geq 0 \\ & x_j = 0 \text{ or } 1 \end{array}$$

where

$$c' = [1, 1, 1, 1, 1] \qquad x' = [x_1, x_2, x_3, x_4, x_5]$$

$$e' = [1, 1, 1] \qquad s' = [s_1, s_2, s_3]$$

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The three feasible solutions are $x' = [1, 0, 1, 0, 0]$, $[0, 1, 0, 0, 1]$, and $[1, 0, 0, 1, 1]$, but the optimum is either of the first two as they give the minimum number of feasible combinations. In this simple example, feasible solutions can be easily noted by observation, but in dealing with a large-size problem of 100 payloads and 500 combinations, there are 2^{500} possible solutions, so a more efficient approach such as the SCA must be employed. The details of this algorithm are given in the technical description.

2.2.2 OPERATIONAL CAPABILITIES

The operational capabilities of the SCA program were designed to permit the user to specify his particular optimization problem according to his needs. One of the features is the user's selection of performance criteria over objective function. For example, if the user likes to see a traffic model of a minimum number of missions, he will have the choice of using unity as a performance criterion, or he may use the surplus Shuttle bay length as a performance criterion to find a traffic model which gives the maximum utilization of space. At present, the SCA permits the user selection of one of the following performance criteria:

a. Unity	minimum number of missions
b. One minus load factor;	maximum utilization of Shuttle cargo weight allowance
c. OMS weight:	the minimum OMS weight
d. Payload margin:	minimum payload margin
e. One minus payload length:	maximum utilization of Shuttle cargo bay

This option can be even more selective if the user has other criteria to be defined in the future. Another available option is the suppression or specification of certain feasible combinations in the traffic model; the program gives the user an error message if he attempts to suppress or specify combinations which would lead to an infeasible solution that is not a traffic model.

2.3 TECHNICAL DESCRIPTION

2.3.1 ANALYSIS

The analysis for this program is quite lengthy. For a thorough discussion of the Set Covering Algorithm, the reader is referred to C. E. Lemke, H. M. Salkin, and K. Spielberg's "Set Covering by Single-Branch Enumeration with Linear Programming Subproblems," Operations Research Vol. 19, pp. 998-1022, 1971 (ref. 1).

As this model is formulated, the set covering problem is to minimize $c'y$ constrained by $Ay \geq e$, $y_j \varepsilon \{0,1\}$, $j \varepsilon \{1,2,..n\}$, which will be referred as $(\hat{I})$. $A = (a_{ij})$ is an m by n matrix with $a_{ij} = 0$ or 1, and e is an m by 1 column of all 1's. Here c is the cost coefficients which are to be positive numbers.

In the application to the traffic model problem, a small variation about the formulation should be realized. Assuming that

(a) A row of A corresponds to a payload

(b) A column of A corresponds to a possible assionment for a feasible combination

(c) The ones in the column denote payloads that could be handled by such an assignment

(d) In a particular solution y, variable $y_j = 1$ corresponds to actually using such a assignment, whereas $y_j = 0$ means that such an assignment will not be used,

one will note that the constraints $Ay \geq e$ do not require the solution of $(\hat{I})$ to be a traffic model, so we are really interested in solving a set covering problem with equality constraints, i.e., $Ay = e$, which is usually called a set partition problem. However, with a little modification in the formulation, a set partition problem can be solved by SCA without any change of the algorithm's generality.

Before we discuss the algorithm, certain facts about the development of SCA and the application of SCA to the set partition problem will be introduced. Compared to the problem $(\hat{I})$, an LP problem is defined as

$$(I_{LP}):\ \min\ (c'x:\ Ax \geq e,\ x \geq 0)$$

1. $(\hat{I})$ is feasible if and only if (I_{LP}) is feasible. (Here feasible means that a feasible solution exists).
2. Assume that $Z^*_{\hat{I}}$ and Z^*_{LP} are the optimal solution values of $(\hat{I})$ and (I_{LP}), respectively; then Z^*_{LP} is always a lower bound of $Z^*_{\hat{I}}$, i.e., $Z^*_{\hat{I}} \geq Z^*_{LP}$.
3. If (I_{LP}) is solved and integral feasible, then $(\hat{I})$ is solved and $Z^*_{\hat{I}} = Z^*_{LP}$.
4. If (I_{LP}) does not give an optimal integer solution, i.e., some x_j of x^*_{LP} are not integral, then a rounded-up solution can always be obtained by setting all the nonintegral variables to 1 ($y_j = 1$ whenever $x_j > 1$).

   This rounded-up solution is $(\hat{I})$-feasible and an upper-bound of $(\hat{I})$.
5. A rounded-up solution obtained from a non-integral extreme point x of (I_{LP}) can always be reduced to another $(\hat{I})$-feasible solution with a smaller cost. We have $Ax \geq e$ and $0 \leq x \leq e$. Consider the columns of A associated with the positive nonintegral x_j variables, and the rows of A corresponding to the constraints that these variables explicitly satisfy. Identify the part of fractional x and the corresponding matrix selected as above by a superscript *. Then we have $A^*x^* \geq e$ and $0 < x^* < e$; i.e., every row of A^* has at least two 1's. Therefore, setting some of the x^* variables to 1 and maintaining $A^*x^* \geq e$ gives a better integer solution than the rounded-up solution which calls for setting all x^* variables to 1. This reduced integer solution is called a purified solution which is reached by successively setting fractional variables to 1 by the "minimum-cost-per-constraint-satisfied" rule until all variables are either 0 or 1 and all constraints are satisfied.

The procedure to reduce the rounded-up solution to a purified solution is called "purification."

6. Let $\bar{y}$ be any feasible solution to $(\hat{I})$. If $\bar{y}$ is not an extreme point for (I_{LP}), i.e., the column of A corresponding to $\bar{y}_j = 1$ and the columns of -I corresponding to $s_i \geq 1$ from a linear dependent set, it can be reduced to a feasible solution y^* for $(\hat{I})$, which is an extreme point for (I_{LP}) and yields a better value for $\hat{Z}_I$. We demonstrate this because the purified solution may not be an extreme point. Let $\bar{y}$ be any $(\hat{I})$-feasible solution that is not an extreme point for (I_{LP}). Since $A\bar{y} - Is = e$, some of the slack variables must be positive; otherwise, $A\bar{y} = e$, which implies $\bar{y}$ is an extreme point for (I_{LP}) since the columns of A corresponding to $\bar{y}_j = 1$ will be linearly independent. Permuting rows to get positive slack variables last, one obtains

$$A\bar{y} - Is = \begin{pmatrix} E_1 \\ E_2 \end{pmatrix}\bar{y} - \begin{pmatrix} 0 \\ I \end{pmatrix}\bar{s} = \begin{pmatrix} e \\ e \end{pmatrix}$$

with $\bar{s} = \left\{A_i/s_i \geq 1\right\}$. Then a permutation of columns to get positive $\bar{y}_j$'s first, leads to

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}\bar{y} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} e \\ 0 \end{pmatrix} = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} e$$

so that

$$A\bar{y} - Is = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} e + \begin{pmatrix} 0 \\ -I \end{pmatrix}\bar{s} = \begin{pmatrix} e \\ e \end{pmatrix}, \qquad \bar{s} \geq e \qquad \text{(i)}$$

The last expression means that (1) $A_{21}e - \bar{s} = e$ (with $\bar{s} \geq e$), i.e., each row of A_{21} contains at least two 1's and (2) $A_{11}e = e$, i.e., each row of A_{11} contains exactly one 1. Thus, permuting rows and columns further, one may exhibit

$$A_{11} = \begin{pmatrix} I & 0 \\ A_{11}^1 & 0 \end{pmatrix}; \qquad \text{and} \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{11}^1 & 0 \\ A^1 & 0 \end{pmatrix}$$

with the above, relation (i) may be rewritten as:

$$\begin{pmatrix} I & 0 \\ A_{11}^1 & 0 \\ A_{21}^1 & A_{21}^2 \end{pmatrix} \begin{pmatrix} e \\ e \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ I \end{pmatrix} \bar{s} = \begin{pmatrix} e \\ e \\ e \end{pmatrix} \tag{ii}$$

Now the columns of

$$\begin{pmatrix} I & 0 \\ A_{11}^1 & 0 \\ A_{21}^1 & -I \end{pmatrix}$$

clearly form a linear independent set, whereas those of

$$\begin{pmatrix} I & 0 & 0 \\ A_{11}^1 & 0 & 0 \\ A_{21}^1 & -I & A_{21}^2 \end{pmatrix}$$

are linearly dependent by the hypothesis that $\bar{y}$ is not an extreme point. Therefore, A^2_{21} has at least one column, say column A^2_j. Then $\bar{s} \geq e \geq A^2_j$ demonstrates that deleting the A^2_j column and replacing $\bar{s}$ by $\bar{s}' = \bar{s} - A^2_j$, yields a new feasible solution, with cost reduced by the cost of the deleted column.

Now suppose the above procedure is repeated. Then one is either again not at an extreme point and may obtain another cost reduction, or one is at an extreme point by virtue of either (a) $\bar{s} = 0$, or (b) $A^2_{21} = 0$.

As an example, consider $c' = (1,2,1,1,2,3,1)$ and

$$\begin{array}{cc} & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ A = & \left[\begin{array}{ccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}\right] \end{array}$$

with (column) labels above.

$\bar{y} = (1,1,0,1,1,0,0)$ is $(\hat{I})$ feasible with $c'\bar{y} = 6$ and $s = (0,0,0,2,1)$. Hence

$$\begin{array}{cc} \text{(column)} & \begin{array}{cccc|ccc} 1 & 2 & 4 & 5 & 3 & 6 & 7 \end{array} \\ \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = & \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right], \end{array}$$

$$A_{11} = \begin{pmatrix} I & 0 \\ A_{11}^1 & 0 \end{pmatrix} = \left[\begin{array}{cc:cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 \end{array}\right],$$

$$\begin{bmatrix} I & 0 \\ A_{11}^1 & 0 \\ A_{21}^1 & A_{21}^2 \end{bmatrix} = \begin{array}{c} \text{(column)}\ \ 1 \quad 2 \quad 4 \quad 5 \\ \left[\begin{array}{cc:cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right] \end{array}$$

Therefore, set $\bar{y}_4 = 0$; then $\begin{pmatrix} \bar{s}'_4 \\ \bar{s}'_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, or $s' = (0,0,0,1,0)$;

$\bar{y}' = (1,1,0,0,1,0,0)$ and $c'\bar{y}' = 5$. Repeating the procedure, we have

$$\begin{bmatrix} I & 0 \\ A_{11}^1 & 0 \\ A_{21}^1 & A_{21}^2 \end{bmatrix} = \begin{array}{c|cc:c} \text{row} \backslash \text{column} & 1 & 2 & 5 \\ \hline 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ \hdashline 3 & 1 & 0 & 0 \\ 5 & 0 & 1 & 0 \\ \hdashline 4 & 1 & 0 & 1 \end{array}$$

Finally, set $\bar{y}'_5 = 0$; then $\bar{s}''_4 = 1-1=0$ or $s'' = (0,0,0,0,0)$, and $\bar{y}'' = (1,1,0,0,0)$ is an extreme point for (I_{LP}) and $c'\bar{y}'' = 3 > c'\bar{y} = 6$.

7. Noticing the payload nonredundancy requirement in the traffic model, we now consider a set partition problem ($Ay = e$) in its relation to the current problem ($\hat{I}$). The actual slack cost is, of course, zero. However, by assigning a "high penalty" cost M into all the slack variables, one will tend to get a minimal ($\hat{I}$) solution with very few positive slack variables. For M large enough, we can either find the minimal solution or show that it is infeasible. Thus, the equality problem is equivalent to

$$(I)^E: \ \min_y (c'y + Me's: \ Ay - Is = e, \ y_j = 0, 1; \ M \text{ large})$$

or a zero-slack cost problem by multiplying the rows of $Ay - Is - e = 0$ by M and adding the sum to $c'y$, or, $(I)^E$ is equivalent to

$$(\hat{I})^E \ \min_y \ (\hat{c}'y): Ay - Is = e, \ y_j = 0,1) - Me'e \ (M \text{ large})$$

Here $\hat{c} = c' + Me'A$; i.e., to each cost element c_j one adds M times the number of 1's in the associated A columns.

As an example, consider

$$(I_1) \qquad \min 5y_1 + 4y_2 + 1y_3 + 2y_4$$

$$\begin{aligned} \text{subject to} \quad & y_1 \phantom{{}+ y_2 + y_3} + y_4 \geq 1, \\ & \phantom{y_1 + {}} y_2 + y_3 \phantom{{}+ y_4} \geq 1, \\ & y_1 \phantom{{}+ y_2} + y_3 + y_4 \geq 1, \\ & y_1, y_2, y_3, y_4 = 0 \text{ or } 1 \end{aligned}$$

Then the minimal y is $(0,0,1,1)$ and $c'y = 3$ with $s = (0,0,1)$. Consider the problem with equality constraints, and set $M = c'e = 12$, (I_1) becomes

$$(I_1)^E: \ \min 5y_1 + 4y_2 + 1y_3 + 2y_4 + 12s_1 + 12s_2 + 12s_3$$

$$\begin{array}{lllllllllll} \text{subject to} & y_1 & & & + y_4 & - s_1 & & & = 1 \\ & & y_2 & + y_3 & & & - s_2 & & = 1 \\ & y_1 & & + y_3 & + y_4 & & & - s_3 & = 1 \end{array}$$
$$y_1, y_2, y_3, y_4 = 0 \text{ or } 1; \quad s_1, s_2, s_3 = 0 \text{ or } 1$$

After each row of constraints is multiplied by 12 and added to the objective function, we get

$$(\hat{I}_1)^E \ \min \ 17y_1 + 16y_2 + 25y_3 + 26y_4 - 36$$

$$\begin{array}{lllllllllll} \text{subject to} & y_1 & & & + y_4 & -s_1 & & & = 1 \\ & & y_2 & + y_3 & & & -s_2 & & = 1 \\ & y_1 & & + y_3 & + y_4 & & & -s_3 & = 1 \end{array}$$
$$y_1, y_2, y_3, y_4 = 0 \text{ or } 1; \ s_1, s_2, s_3 = 0 \text{ or } 1$$

The minimal y is (0,1,0,1) and $\hat{c}'y - 36 = 6$ with $s=(0,0,0)$. The above procedure permits the solution of the Set Partition Problem (SPP) by means of an algorithm (SCA) oriented toward the set covering problem. It is recommended that M be set to 500 in the traffic model problem.

2.3.2 METHOD OF SOLUTION

To solve problem $(\hat{I})$, namely, $(\hat{I})$: $\min (c'y: Ay \geq e, y_j \in \{0,1\}, j \in \{1,2,\ldots,n\})$, an enumerative single-branch scheme is employed.

The search starts at the origin (node 0), with all y_j "free", i.e., tentatively considered to be either 0 or 1 and possibly to be fixed at value 1 on a forward step. At the general node v (on "level" ℓ), ℓ components of y have been explicitly fixed at 1 on forward steps. Others may have been "canceled" (fixed at 0) at level ℓ, as one has ascertained that corresponding forward steps would not lead to a solution.

The task at node v is to:

(i) Solve a basic problem (I_{LP}) with fixed variables substituted, and to look for improved solutions as outlined in 2.3.1. If the objective function of (I_{LP}) exceeds an available bound on $(\hat{I})$, or (I_{LP}) is infeasible, one may "backtrack", i.e., the search reverts to the predecessor node v^-, linked to v by the branch j^* (variable y_{j^*} having been fixed at 1) and the search continues at node v^- with y_{j^*} canceled to 0. (If no predecessor exists, the search terminates and the solution is the existing upper bound.)

(ii) Cancel whatever variables may be canceled from further consideration.

(iii) Select among the remaining variables a "branch" variable j^* to be fixed at 1 on the next forward step. The state of search is essentially recorded in an $(\ell + 1, n)$ η matrix: $\eta(\ell,j) = 0$ when variable j was fixed (i.e., canceled or selected as a branch) at some level up to and including the current level ℓ (i.e., at some predecessor node of v, or at v). Otherwise, $\eta(\ell,j)$ is the current number of unsatisfied constraints that can be satisfied if the free variable j is fixed at 1. Initially, $\eta(0,j) = \sum_{i=1}^{n} a_{ij}$ for all j.

The current number of unsatisfied constraints is kept in a vector ϕ, i.e., $\phi(o) = m$ and $\phi(\ell) = \phi(\ell-1) - \eta(\ell - 1, j^*)$. For any ℓ,j, the condition $\phi(\ell) \geq \eta(\ell,j) \geq 0$ is always held.

Consider the search at level ℓ prior to a forward step. The following tests may reduce the number of branch alternatives.

a. The subproblem at level ℓ is feasible only if $\sum_{j=1}^{n} \eta(\ell,j) \geq \phi(\ell)$. If this is not met, a backward step may be taken.

b. Let Z^v be the current objective function ($\sum c_j$ over all j with y_j fixed at 1) and Z^* the objective function of the best integer solution. Let F be the set of free variables. Then the double ceiling test, i.e., $Z^v + c_t + \min_{w \in F-\{t\}} c_w \geq Z^*$ permits the cancelation of t if $\eta(\ell,t) < \phi(\ell)$.

c. If at a current subproblem, a variable column dominates another variable and the dominated column has a higher cost than the dominating one, then the dominated variable never need be considered as a branch candidate, that is, if for any level $\ell (\ell \geq 1)$, $\eta(\ell - 1, j^*) \geq \eta(\ell - 1, j_1) > 0$ and $\eta(\ell, j^*) = \eta(\ell, j_1) = 0$ with $c_{j^*} \leq c_{j1}$, then set $\eta(\ell - 1, j_1) = 0$ (i.e., cancel j_1 at level $\ell - 1$). Here y_{j^*} is the branch from node v^- to v. The above is commonly referred to as the local column dominance test.

d. Let $(\hat{I})^\ell$ be the level ℓ integer (set covering) subproblem. Define $(I_{LP})^\ell$ to be the corresponding continuous LP problem. Then as stated in section 2.3.1:

   (i) $(I_{LP})^\ell$ is feasible if and only if $(I)^\ell$ is feasible.

   (ii) If $(I_{LP})^\ell$ is feasible, then the value of the minimal functional serves as a lower bound for the best $(\hat{I})^\ell$ solution. Furthermore, if $(I_{LP})^\ell$ is integer feasible it solves $(I)^\ell$.

2.3.3 THE ALGORITHM

The algorithm is basically a (0,1) search embodying the elaborations discussed previously. At node v, level ℓ, we attempt to solve the linear program $(I_{LP})^\ell$. If it is infeasible or its objective function exceeds the current ceiling (z^*, initially set to $c'e$) the search reverts to v^-. If the minimal solution is (0,1) feasible, it is recorded and a backward step is taken.

Otherwise, we record the column (denoted by j*) associated with the smallest value of the optimal variables, and then extract the sub-subproblem that contains the columns of A associated with the positive fractional variables and the rows corresponding to the constraints they satisfy, and obtain a solution to this problem. This solution should be extreme point which is obtained through the procedure introduced in 2.3.1. If the overall $(\hat{I})$-feasible solution exceeds the ceiling z^*, a backward step is taken.

If node v was just reached on a forward step, we construct the preferred set $P(i^*)$. Preferred row i^* is defined to be the first i such that $\sum_{j\varepsilon F} a_{i^*j} \leq \sum_{j\varepsilon F} a_{ij}$ for all i of which j^*th entry is 1. $P(i^*)$ is the columns containing 1 on the preferred row i^*. We then cancel locally dominated columns. The purpose of constructing the preferred set is if a forward step fails, we cancel j^* and select another branch from the preferred set.

If v was reached from v^-, we branch j^* to reach node v^+. However, if v was reached on a backward step (that implies when we tried to reach v^+ and found that we cannot generate an improved solution), we select a branch by the minimum-cost-per-constraint-satisfied rule from $P(i^*)$ to define v^+. During the branch selection procedure, the double ceiling test is applied so we can hopefully exclude some branch candidate from further consideration. If at any time every variable in the $P(i^*)$ is canceled, we take a backward step.

When a subprogram contains no constraints, we have reached an $(\hat{I})$-feasible solution. In all cases, a (0,1) solution is not recorded until it is reduced to an extreme point of (I_{LP}). The search terminates when the level counter ℓ becomes less than 0.

Linear programming is the crucial feature of this algorithm. Initially, the sub-subprogram is always extracted if (I_{LP}) is not integral feasible. A solution of the sub-subproblem is reached by successively setting variables to 1 by the "minimum-cost-per-constraint-satisfied" rule. Once this solution is reached, we branch forward by setting the minimal fractional variable of the current LP to 1. The hope is that setting the minimum fractional variable to 1 will tend to give the largest alteration in the optimal LP solution which is the lower bound of $(\hat{I})$. When a node is reached for the first time on a backward step, LP is performed with the minimum fractional value at 0. This means that last branch link j^* has been canceled.

The other tests, such as local dominance, double ceiling, and so on, were incorporated more because of their simplicity and ease of applicability than their actual usefulness in curtailing the length of the search. In dealing with the usual large number of variables, search features designed to look at individual variables appear relatively less than attractive as compared with the linear programming part that tends to solve entire subproblems. This is reflected by the fact that for each computer run, 90 percent of the computing time is consumed in LP. The saving of search time because of successful LP solution appears more important than the tests mentioned above; especially after we experienced most of the time that the $(\hat{I})$-feasible solution was directly generated by LP.

A general functional flowchart is given in section 5.1.

2.3.4 OUTLINE OF THE ALGORITHM

I. Initialization

1.1 Set $\eta(1,j) = \sum_{i=1}^{i=m} a_{ij}$ for $j = 1, \ldots, n$; $F = \{1,2,\ldots,n\}$; $s = e$

1.2 $\phi(1) = m, \xi(1) = -1, \sigma(1) = 0$

1.3 $\ell = 0, \gamma = 0, z = 0, z^* = c'e, \alpha = 1$

II. Point Algorithm

2.1 Updating counters $\ell = \ell + 1, \gamma = \gamma + 1$

2.2 Test for solution and lower bounds. If $\phi(\ell) = 0$, go to 4.1 (solution). If $\xi(\ell) \geqq z^* - \tau$, go to 3 (backward step).

2.3 Update ϕ, F. Set $\phi(\ell + 1) = \phi(\ell)$, $F = \{j \,|\, |\eta(\ell,j)| > 0\}$. (Variables with $\eta(\ell,j)$ entries less than 0 are members of the preferred set See 2.12).

2.4 Feasibility test. If $\sum_{i=1}^{i=n} |\eta(\ell,j)| < \phi(\ell)$, go to 3.

2.5 Linear programming. Solve the linear program associated with the current subproblem.

(i) If infeasible or ceiling exceeded, go to 3.

(ii) If (0,1) feasible, record overall $(\hat{I})$ solution, go to 4.1.

(iii) Otherwise, set $\xi(\ell)$ equal to the minimal LP objective function added to the current value of z; and set j^* corresponding to the smallest fractional value of the LP variables.
Obtain the roundup solution if currently feasible, and go to 2.6; otherwise, go to 2.7 when $\alpha = 0$ and go to 2.10 if $\alpha = 1$.

2.6 Sub-subproblem. Extract and obtain a solution to the sub-subproblem. Record the overall $(\hat{I})$ solution. Go to 4.2.

2.7 Feasibility test. If any $j, \phi(\ell) - |\eta(\ell,j)| = 0$, set $j^* = j$ (solution at next level), and go to 2.13.

2.8 Double-ceiling test. If any j such that $\eta(\ell,j) > 0$, $z + c_j + \min_j c_j \geqq z^* - \tau$, set $\eta(\ell,j) = 0$ (i.e., cancel y_j at the level ℓ); if $\eta(\ell,j) = 0$ for all j, go to 3.

2.9 Select branch $j(\alpha = 0)$. If $\eta(\ell,j) \geqq 0$ for all j (preferred set null), go to 3. Set j^* = first j to satisfy $\left[c_{j^*}/|\eta(\ell,j^*)|\right] \leqq \left[c_j/|\eta(\ell,j)|\right]$ for all j such that $\eta(\ell,j) < 0$ (i.e., for all j in the preferred set).

2.10 Update ϕ, η. Set $\phi(\ell + 1) = \phi(\ell + 1) - |\eta(\ell,j^*)|$, $\eta(\ell + 1,j) = |\eta(\ell,j)|$ for $j = 1,..,n$. If $\phi(\ell + 1) = 0$, go to 2.13. For each i such that $s_i \leqq 0$ (i.e., for all rows of the subproblem) and $a_{ij^*} = 1$, set $\eta(\ell + 1,j)$ to $\eta(\ell + 1,j) - 1$ when $a_{ij} = 1$ and $j \epsilon F$.

2.11 Local dominance test. If, for any $j \epsilon F$, $\eta(\ell + 1,j) = 0$ and $c_{j^*} \leqq c_j$, set $\eta(\ell,j) = 0$. If $\alpha = 0$, go to 2.13.

2.12 Construct preferred set. Set i^* (the preferred row) to be the first i such that $\sum_{j \epsilon F} a_{i^*j} \leqq \sum_{j \epsilon F} a_{ij}$ for all i with $s_i < 0$. Set $\eta(\ell,j)$ to $-\eta(\ell,j)$ for the j such that $a_{i^*j} = 1$.

2.13 Update the slack column and other parameters. Set s_i to $s_i + a_{ij^*}$ for $i = 1,\ldots,m$, $z = z + c_{j^*}$, $\alpha = 1$, $\xi(\ell + 1) = \xi(\ell)$. Go to 2.1.

III. Backward Step

3. Set $\ell = \ell - 2$. If $\ell < 0$, go to 5 (termination). $j^* = \sigma(\ell + 2)$. $z = z - c_{j^*}$, $s_i = s_i - a_{ij^*}$ for $i = 1,\ldots,m$. $\xi(\ell + 2) = \xi(\ell + 1)$. $\alpha = 0$. Go to 2.1.

IV. Feasible Solution

4.1 LP (0,1) feasible, or search reaches solution. Reduce overall solution to extreme point. Record current solution y and $c'y = z$. Set $z^* = z$. Go to 3.

4.2 Search produced via sub-subproblem. Reduce overall solution to an extreme point. Record the overall y and $c'y = z$. Set $z^* = z$. If $\xi(\ell) \geq z^* - r$, go to 3. If $\alpha = 1$ go to 2.10, otherwise ($\alpha = 0$) go to 2.7.

V. Termination

5. Optimal solution ascertained or no feasible solution exists.

2.3.5 AN EXAMPLE

The following example will demonstrate how the algorithm is applied to the traffic model problem. The input obtained from MPLS is an occurrence table which is as shown in table I.

The constraint matrix is displayed as in table II. Assume we intend to find a traffic model with the minimum number of missions and the cost coefficients are in the column matrix of all 1's. Since the traffic model is a set partition problem, we transform the cost function as indicated in section 2.3.1 with $M = 100$. The new cost column is as shown in table II. The search follows the outline of the Set Covering Algorithm. As an example, the enumeration of the problem in table II is demonstrated in table III. In each level on the forward step, the LP subproblem is solved and the lower bound is updated. Ceiling z^* is recorded after the rounded-up solution of LP has been reduced to an extreme point. The search always branches forward until it reaches node 6. At node 6, the lower bound is equal to the ceiling, so a backward step is taken. In each backward step, it is found that $z^* - \xi(\ell) < \tau$, so z^* is the solution.

TABLE I.- OCCURRENCE TABLE

Payloads	Feasible combinations
1	1,16,17,18,19,20,21,64,65,66,67,68,73,74
2	2,16,23,24,25,26,27,28,29,64,65,66,69,70,71,73,74
3	3,17,23,30,31,32,33,34,35,64,67,68,69,70,73,74
4	4,36
5	5,37,38,39,40,41
6	6,37,42,43,44
7	7,38,45,46,47
8	8,48,49,50,51
9	9,48,52,53,54
10	10,22,24,30,55,71
11	11,18,25,31,39.42,45,49,52,56,57,58,72
12	12,19,26,32,59,60,65,67,69,73
13	13,20,27,33,36,55,56,61,62,66,68,70,71,74
14	14,21,28,34,40,43,46,50,53,57,59,61,63,72
15	15,22,29,35,41,44,47,51,54,58,60,62,63,72

TABLE II.- AN EXAMPLE CONSTRAINT MATRIX

[A blank entry in the constraint matrix means zero]

Col.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
c'	(101	101	101	101	101	101	101	101	101	101	101	101	101	101	101	201	201	201	201	201	201	201	201	201	201)
Row																									
1	1															1	1	1	1	1	1				
2		1														1							1	1	1
3			1														1						1		
4				1																					
5					1																				
6						1																			
7							1																		
8								1																	
9									1																
10										1												1		1	
11											1							1							1
12												1							1						
13													1							1					
14														1							1				
15															1							1			

TABLE II.- AN EXAMPLE CONSTRAINT MATRIX (Continued)

[A blank entry in the constraint matrix means zero]

Col.	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
c'	(201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201	201)
Row																									
1																									
2	1	1	1	1																					
3					1	1	1	1	1	1															
4											1														
5												1	1	1	1	1									
6												1					1	1	1						
7													1							1	1	1			
8																							1	1	1
9																							1		
10					1																				
11						1								1			1			1				1	
12	1						1																		
13		1						1			1														
14			1						1						1			1			1				1
15				1						1						1			1			1			

TABLE II.- AN EXAMPLE CONSTRAINT MATRIX (Continued)

[A blank entry in the constraint matrix means zero]

Col.	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74
c'	(201	201	201	201	201	201	201	201	201	201	201	201	201	301	301	301	301	301	301	301	301	301	401	401)
Row																								
1														1	1	1	1	1					1	1
2														1	1	1			1	1	1		1	1
3														1			1	1	1	1			1	1
4																								
5																								
6																								
7																								
8	1																							
9		1	1	1																				
10					1																1			
11		1				1	1	1														1		
12									1	1					1		1		1				1	
13					1	1					1	1				1		1		1	1			1
14			1				1		1		1		1									1		
15	1			1				1		1		1	1									1		

TABLE III.- ENUMERATION HISTORY OF EXAMPLE

Enumeration diagram	Algorithm parameters									
	ℓ	α	$\xi(\ell)$	Z_{RD}	Z^*	$\sigma(\ell)$	P		F_ℓ	j^*
							P_1	P_0		
(Initialization) set $\tau = 0.998$	0	1	-1	-	10^{10}	-	-	-	1,2,...,74	
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & {} \end{tikzcd}$	1	1	6.5	910	207	-	-	-	1,2,...,74	22
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \arrow[r, "y_{38} = 1"] & {} \end{tikzcd}$	2	1	6.5	409	107	22	22	10,15	F_1 - P	38
$\begin{tikzcd} 1 \arrow[r, "y_{22} ="] & 2 \arrow[r, "y_{38} = 1"] & 3 \arrow[r, "y_{48} = 1"] & {} \end{tikzcd}$	3	1	6.5	308	107	38	22,38	5,7,10,15,41,47	F_1 - P	48
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \arrow[r, "y_{38} = 1"] & 3 \arrow[r, "y_{48} = 1"] & 4 \arrow[r, "y_{42} = 1"] & {} \end{tikzcd}$	4	1	6.5	308	107	48	22,38,48	5,7,8,9,10,15,41,47, 51,54	F_1 - P	42
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \arrow[r, "y_{38} = 1"] & 3 \arrow[r, "y_{48} = 1"] & 4 \arrow[r, "y_{42} = 1"] & 5 \arrow[r, "y_{21} = 1"] & {} \end{tikzcd}$	5	1	6.75	710	107	42	22,38,48 42	5,6,7,8,9,10,11,15,37 39,41,44,45,47,48,49 51,52,54,58	F_1 - P	21
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \arrow[r, "y_{38} = 1"] & 3 \arrow[r, "y_{48} = 1"] & 4 \arrow[r, "y_{42} = 1"] & 5 \arrow[r, "y_{21} = 1"] & 6 \end{tikzcd}$	6	0	7	7	7	21	22,38,48, 42,21	1,5,6,7,8,9,10,11,14, 15,18,37,39,40,41,43, 44,45,46,47,49,50,51, 52,53,54,57,58,63,72	F_1 - P	-
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \arrow[r, "y_{38} = 1"] & 3 \arrow[r, "y_{48} = 1"] & 4 \arrow[r, "y_{42} = 1"] & 5 \end{tikzcd}$	5	0	6.75	-	7	-	-	-	-	-
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \arrow[r, "y_{38} = 1"] & 3 \arrow[r, "y_{48} = 1"] & 4 \end{tikzcd}$	4	0	6.5	-	7	-	-	-	-	-
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \arrow[r, "y_{38} = 1"] & 3 \end{tikzcd}$	3	0	6.5	-	7	-	-	-	-	-
$\begin{tikzcd} 1 \arrow[r, "y_{22} = 1"] & 2 \end{tikzcd}$	2	0	6.5	-	7	-	-	-	-	-
$\begin{tikzcd} 1 \end{tikzcd}$	1	0	6.5	-	7	-	-	-	-	-

2.3.6 POSSIBLE IMPROVEMENTS

The following modifications may result in the improvement of program efficiency.

1. Based on the fact that the rounded-up solution is always feasible, in outline 2.5, after obtaining the rounded-up solution, control should always go to 2.6.

2. In the outline 2.11, the local dominance test may lead to an infeasible solution unless we stress column j is dominated by column j^*. For example, consider the problem

$$\begin{array}{llll} \min & 5x_1 + x_2 + 2x_3 & & \\ & x_1 & = 1 \\ & x_1 + x_2 & = 1 \\ & x_1 + x_2 + x_3 & = 1 \end{array}$$

$c_2 = 1 < c_1 = 5$ and assuming $j^* = x_2$; canceling x_1 will lead to an infeasible solution which will not satisfy the first row of constraints.

3. In solving a set partition problem, any $j \varepsilon F$ such that $A_j \cdot A_{j^*}^T \neq 0$ should be canceled on a forward step. By doing that, not only can we exclude more branch candidates, but also we can save the computing time in solving the LP subproblem. For example, in tables II and III, in the first level according to the algorithm we only canceled y_{10} and y_{15}. However, noticing the equality constraints, y_{24}, y_{30}, y_{55}, y_{71}, y_{29}, y_{35}, y_{41}, y_{44}, y_{47}, y_{54}, y_{58}, y_{60}, y_{62}, y_{63} and y_{72} should be canceled also, since any of those variables not equal to zero will result in an infeasible solution by branching on y_{22}.

3. PROGRAM USAGE

3.1 INPUT DESCRIPTION

The two types of input data for the SCA program are the source input from a file on logical unit 2 and the tutorial input data specified by the user. All source input data are nonnegative integars. Logical unit 2 is built as a temporary file in the main program to store the feasible payload combinations from the MPLS. Because of the core storage limitations, the SCA is designed to handle a traffic model problem with a maximum of 500 combinations and 100 payloads. If the number of feasible combinations exceeds 500, the MPLS will reduce it to within that limit. Subroutine TABLE reads in the information on each feasible combination from logical unit 2. Within the information combination ID, different payloads in that combination, and combinations cost coefficients are stored. All the inputs the SCA needs is a vector c, the vector of cost coefficients, and a constraint matrix A which is displayed as an occurrence table. The cost vector is directly stored in the column array KCJ. The matrix A is stored as two vectors, i.e., KARR and NOPERC. For example, suppose A is

$$\begin{array}{c|cccc} \text{column} & 1 & 2 & 3 & 4 \\ \text{row} & & & & \\ \hline 1 & 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 1 \\ 3 & 1 & 0 & 0 & 1 \end{array}$$

Then A is posed as a vector KARR $(J)_{J=1,6}$ = (1,3;2;1;2,3), which contains the indices of rows of all nonzero entries columnwise from left to right. The position vector, NOPERC $(I)_{I=1,5}$ = (1,3,4,5,7) containing $n + 1$ elements, keeps track of the column interval in KARR. This implies that column j of A starts in KARR (NOPERC(j)) and ends in KARR (NOPERC(j+1) - 1). For example, we want to locate the 1's in column 4.

Set j = 4, then KARR(NOPERC(j)) = KARR(5) =2, KARR(NOPERC(j+1) - 1) = KARR(7-1) = KARR(6) = 3; so column 4 has two 1's, on row 2 and 3, respectively.

The tutorial input data required for the SCA program can be either user-specified data from a demand terminal or from card decks. These data are read via logical unit 5 using a free field format. Sample input/output is given in section 5.4 for the reader's reference.

3.2 PROGRAM RUN PREPARATION

The SCA has been implemented on the UNIVAC 1110 EXEC 8 system as a subprogram of SAMPLE. FORTRAN V standard logical input and output devices are used for tutorial input (logical 5) and printed output (logical 6). For the source input of the SCA, logical 2 is used.

In order to eliminate the reiteration of usage instructions which have already been elaborated in the SAMPLE User's Guide (ref. 2), a discussion of the natures of those different interactive options will be introduced as follows instead.

3.2.1 INTERACTIVE OPTIONS

1: USE PREVIOUSLY DEFINED FEASIBLE COMBINATIONS
2: USE INTERACTIVE FEATURE IN TRAFFIC MODEL
3: NONE OF THE ABOVE

Option 1 will enable the user to use a data file input which contains previously defined feasible combinations for SCA execution. In this way, the execution time of the MPLS can be saved.

Option 2 mainly supports the communication between the SCA and the OSARS. In case the traffic model cannot be scheduled by the OSARS, or is not desired for some other reason, this option gives the user the means of changing the traffic model.

Option 3 implies that the user does not desire to select option 1 or 2.

3.2.2 COST CRITERIA OPTIONS

This option gives user the choice of one of the performance criteria against which traffic model will be generated. The meanings of different criteria have been discussed in section 2.2.2.

3.2.3 CRITERIA FOR FLIGHT/COMBINATION SELECTION OPTION

```
CHOOSE CRITERIA FOR FLIGHT/COMBINATION SELECTION:

     1:  MAXIMUM NUMBER OF PAYLOADS
     2:  MAXIMUM PRIORITY
     3:  MINIMUM COST
     4:  MINIMUM COST PER PAYLOAD
     5:  NONE OF THE ABOVE
```

These criteria help the user to make the decision about which mission he likes to enter in the traffic model. If the user is interested in adding certain feasible combinations which contain the largest number of payloads, he can select option 1. Then the program will print out five feasible combinations with maximum number of payloads. Presently, the priority is determined by the number of payloads, so option 1 and option 2 give the same output. By selecting option 3, the user will get a list of five feasible combinations with minimum cost coefficient. Option 4 will give a list of five feasible combinations cost coefficient per payload. Control goes to the Manual Flight/ Combination Option when 5 is chosen.

3.2.4 MANUAL FLIGHT/COMBINATION OPTIONS

The available options are:

```
  N:  ENTER COMBINATION "N"
 -1:  NEW SELECTION CRITERIA ARE DESIRED
 -2:  CONTINUE ON TO TERMINATION
 -3:  VIEW ALL COMBINATIONS SPECIFIED SO FAR
 -4:  VIEW INFORMATION ON COMBINATIONS SPECIFIED SO FAR
 -5:  REMOVE LAST SPECIFIED COMBINATION
```

This part follows right after "Criteria for Flight/Combination Selection Option" in which the user has viewed the relevant information on the combination he possibly adds in the traffic model. If the user would like to add a specific mission in the traffic model, he responds by entering the feasible combination number "N." The program will take that feasible combination into the SCA's partial solution. If the user still wants to see more relevant criterion selection information, he just enters -1. The control goes back to Criteria for Flight/Combination Selection Option. Option -2 makes the control continue to find a traffic model with the specified partial solution. A traffic model is built around the partial solution; local optimality replaces global optimality. Option -3 gives the user a chance to look through all the missions already specified in the partial solution so he will not enter any of those missions again. By selecting option -4, the user will see a list of missions specified in the partial solution and their relevant information on Shuttle sequence, inclination, payload margin, and so on. After the user looks through the detailed information on those specified combinations and he is not satisfied with the mission he just added on the partial solution, he can enter option -5 to remove it.

3.2.5 TRAFFIC MODEL INFORMATION OPTION

The tutorial of this option is

```
     DO YOU WISH TO SEE INFORMATION ON THESE MISSIONS?
           0:  NONE
          -1:  PRINT ALL
          -2:  PRINT ALL AND SAVE ON SCRATCH FILE
          -3:  SAVE ON SCRATCH FILE ONLY
           N:  ENTER MISSION "N"   .
```

This option follows the solution of the traffic model. If the user does not need to see any detail information on the traffic model, he just enters 0. He can enter -1 to see them all or enter N, the combination number, to get the information on a particular one. Options -2 and -3 provide the user an opportunity to store the traffic model on a scratch file (logic unit 1) for further analysis.

3.2.6 MISSION OMIT OPTION

The display statement is

    WHICH MISSIONS DO YOU WANT OMITTED?

After the user viewed some information on the traffic model, he may need to delete certain combinations by use of this option.

3.2.7 TERMINATE OPTION

The display statement is

    SELECT AN OPTION: (3 TO TERMINATE)

Input 3 to terminate execution.

3.3 <u>OUTPUT DESCRIPTION</u>

3.3.1 NORMAL OUTPUT

Normal output for the SCA program can be classified into five basic types:

1. Source input data - The initial output of the SCA is the source input data which is displayed in the occurrence table. The title of that table is printed out as "n OCCURRENCE TABLE," where n is the year with which the particular case is executed. This is immediately followed by "PAYLOAD" and "COMBINATIONS." Under the column of "PAYLOAD" are printed out the payload identifications. Under the column of "COMBINATIONS" are combination numbers which carry that payload.

2. Tutorial instructions data - These data are printed out in the alphanumeric format and provide the user a guide of various interactive selections during the execution of the SCA.

3. Criteria for flight/combination selection data - The output is written out in two columns; the first column contains the payload identification, and the second column displays the corresponding criterion.

4. Combination information data - The output of these data is in alphanumerical format and displays the relevant information about the mission in an understandable form. This output is requested by the user in the Manual Flight/Combination Option.

5. Traffic model data - These data give the total number of combinations in the traffic model, the mission identifications, and the total cost of this traffic model.

All of the five output types can be identified in a sample input/output in section 5.4.

3.3.2 ABNORMAL OUTPUT

Diagnostic messages from subroutines of the SCA are listed below.

Diagnostic Message	Subroutine	Description/Action
THIS MISSION IS UNACCEPTABLE BECAUSE PAYLOAD XX IS DUPLICATED	SET	In the flight/combination option, the user entered more than one combination which covers the same payload. XX is the payload ID.
ERROR, ALL ROWS ARE NOT COVERED, RETURN TO PREVIOUS SOLUTION, PLS NOT COVERED ARE XX, YY, ZZ	SET	The user wanted to omit some combinations that will cause some payloads not to be covered. The program will return to a previous partial solution. If the user wants to omit some combinations this time, he should refer to the occurrence table and be sure all payloads can be covered. XX, YY, ZZ are the payloads which failed to be covered.
REDIMENSION KARR (XX) TO KARR (YY)	SET	The total number of unity entries in the contraint matrix YY which exceeds the dimensioned space XX.

Diagnostic Message	Subroutine	Description/Action
AVAILABLE STORAGE EXCEEDED AFTER ITERATIVE STEP NO. I	SET	Variable KXB (I,J) should be redimensioned by increasing the value of I. The value of KTEST should also be increased by the same amount.
TABLE ERROR***INPUT TO SCA IS CLOBBERED**	TABLE	This message implies that more than one payload in a feasible combination have the same ID. It could be caused by the numbering or naming method in the MPLS.

4. EXECUTION CHARACTERISTICS

4.1 RESTRICTIONS

The SCA program has these limitations:

a. The largest traffic model problem the SCA can accept is 100 payloads with 500 combinations.

b. The level the SCA can reach is limited to a maximum of 6.

c. The program is valid only if there exists a feasible solution to the traffic model.

d. The maximum number of iterations allowed in LP is 243.

4.2 RUNNING TIME

The run time for the SCA program may varv depending on the problem executed. A rough estimate of the time needed for a run can be obtained from the plot of the number of missions as a function of running time in figure 2. The data from which this plot is constructed are from 12 cases with unity cost coefficients.

4.3 ACCURACY/VALIDITY

The SCA program is written in single precision, and has been checked out using small problems, in a range from 11 feasible combinations and 8 payloads to 229 combinations and 97 payloads, with a maximum of 3 payloads per combination. It is felt that the program is operating correctly and is providing reliable solutions to the problems.

Problems used in checkout included three data sets and 36 problems. Each of these problems has been tested on four different sets of cost coefficients with a maximum of three payloads per combination. All solutions have been checked and were found to satisfy the constraints and to be optimal. Some other problems with more than 400 combinations which have a maximum of four payloads per combination were also run and resulted in an indication that no solutions were possible. The reason for this has not been thoroughly analyzed.

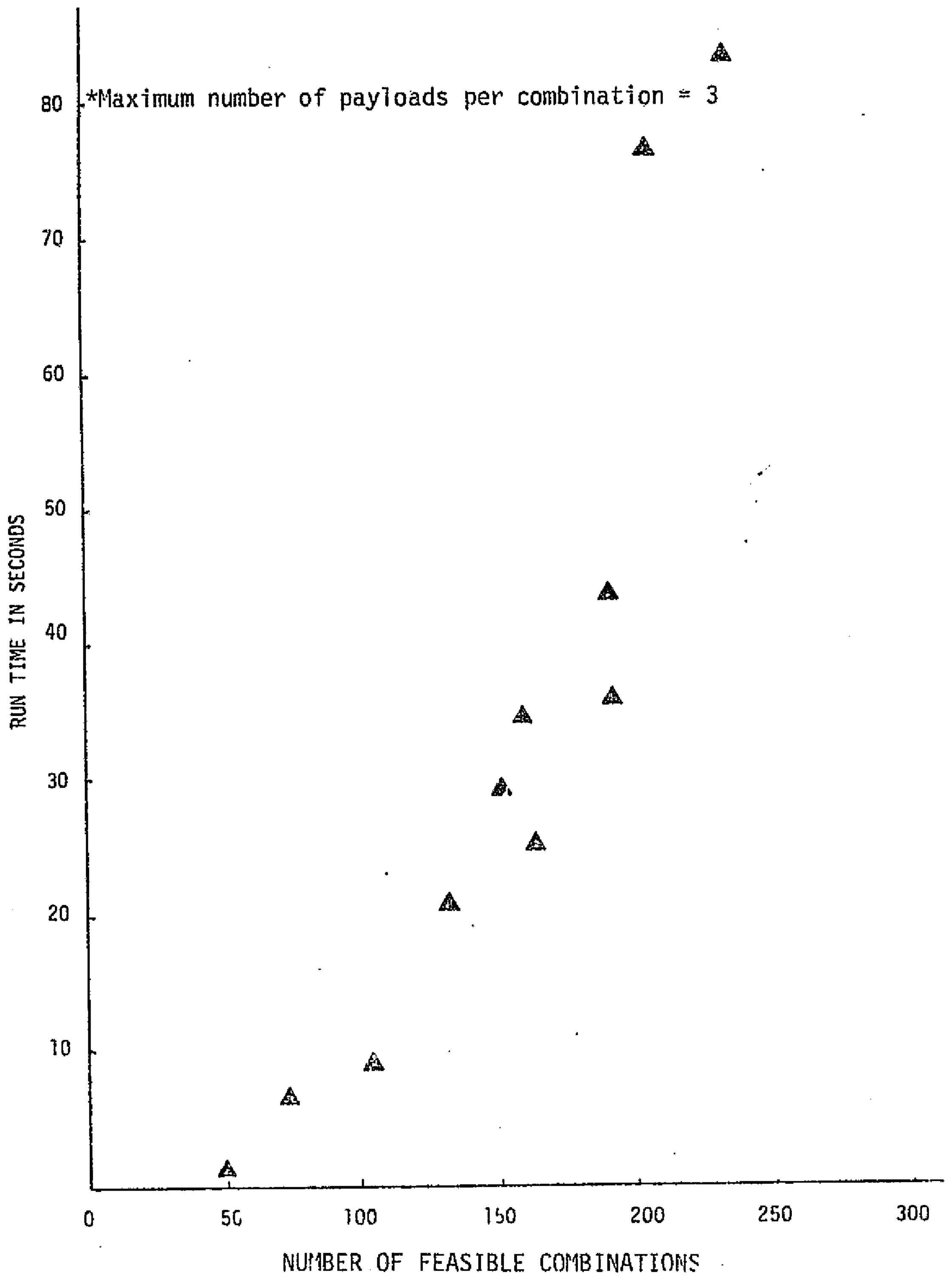


Figure 1.- Sample run times.

5. REFERENCE INFORMATION

5.1 FUNCTIONAL FLOWCHART

Figure 2 illustrates the flow of the controlling subprogram logic. Refer to section 2.1 for the definitions of those symbols used in this functional flowchart .

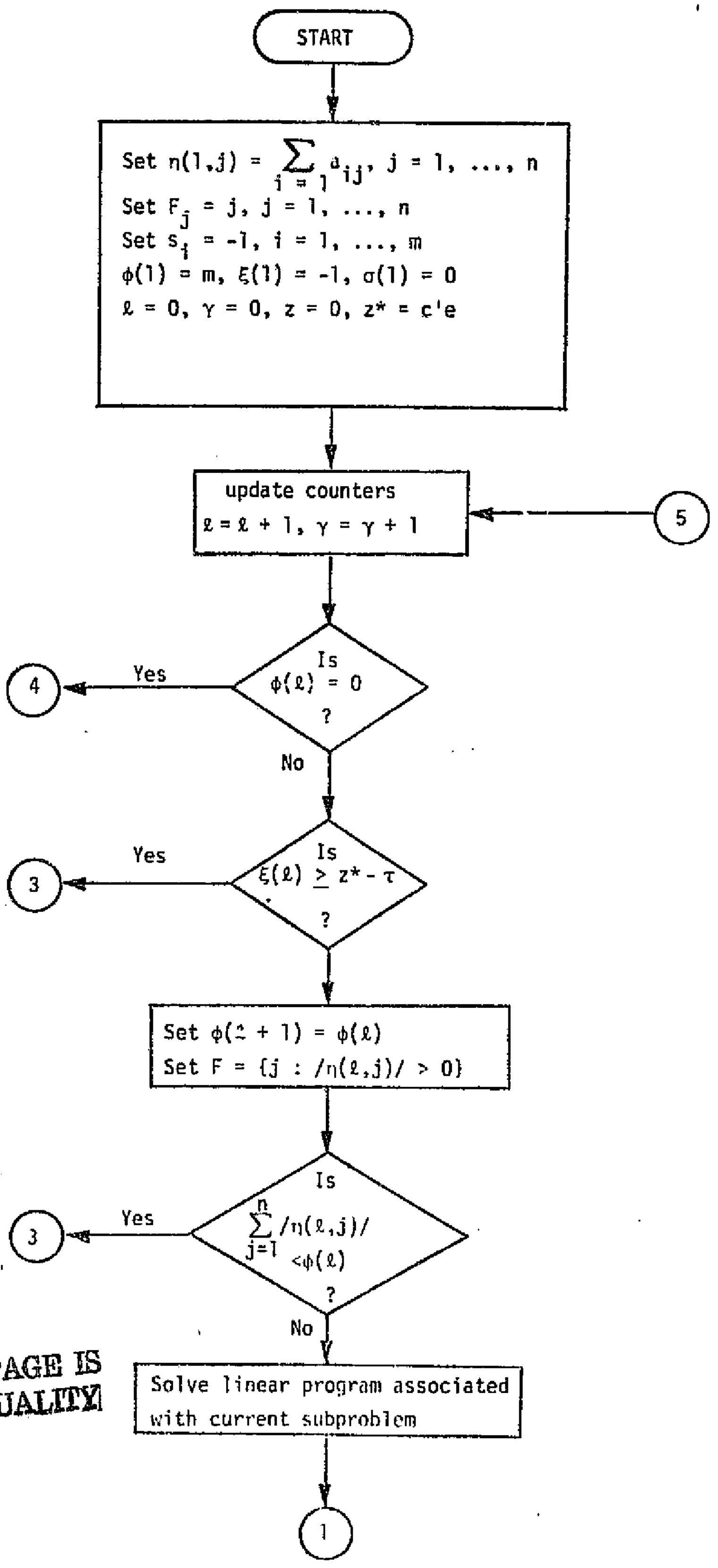

START
Set $\eta(1,j) = \sum_{i=1} a_{ij}$, $j = 1, \ldots, n$
Set $F_j = j$, $j = 1, \ldots, n$
Set $s_i = -1$, $i = 1, \ldots, m$
$\phi(1) = m$, $\xi(1) = -1$, $\sigma(1) = 0$
$\ell = 0$, $\gamma = 0$, $z = 0$, $z^* = c'e$
update counters
$\ell = \ell + 1$, $\gamma = \gamma + 1$
5
Is $\phi(\ell) = 0$?
Yes
4
No
Is $\xi(\ell) \geq z^* - \tau$?
Yes
3
Set $\phi(\ell + 1) = \phi(\ell)$
Set $F = \{j : /\eta(\ell,j)/ > 0\}$
Is $\sum_{j=1}^{n} /\eta(\ell,j)/ < \phi(\ell)$?
Yes
3
No
Solve linear program associated with current subproblem
1

Figure 2.- SCA functional flowchart.

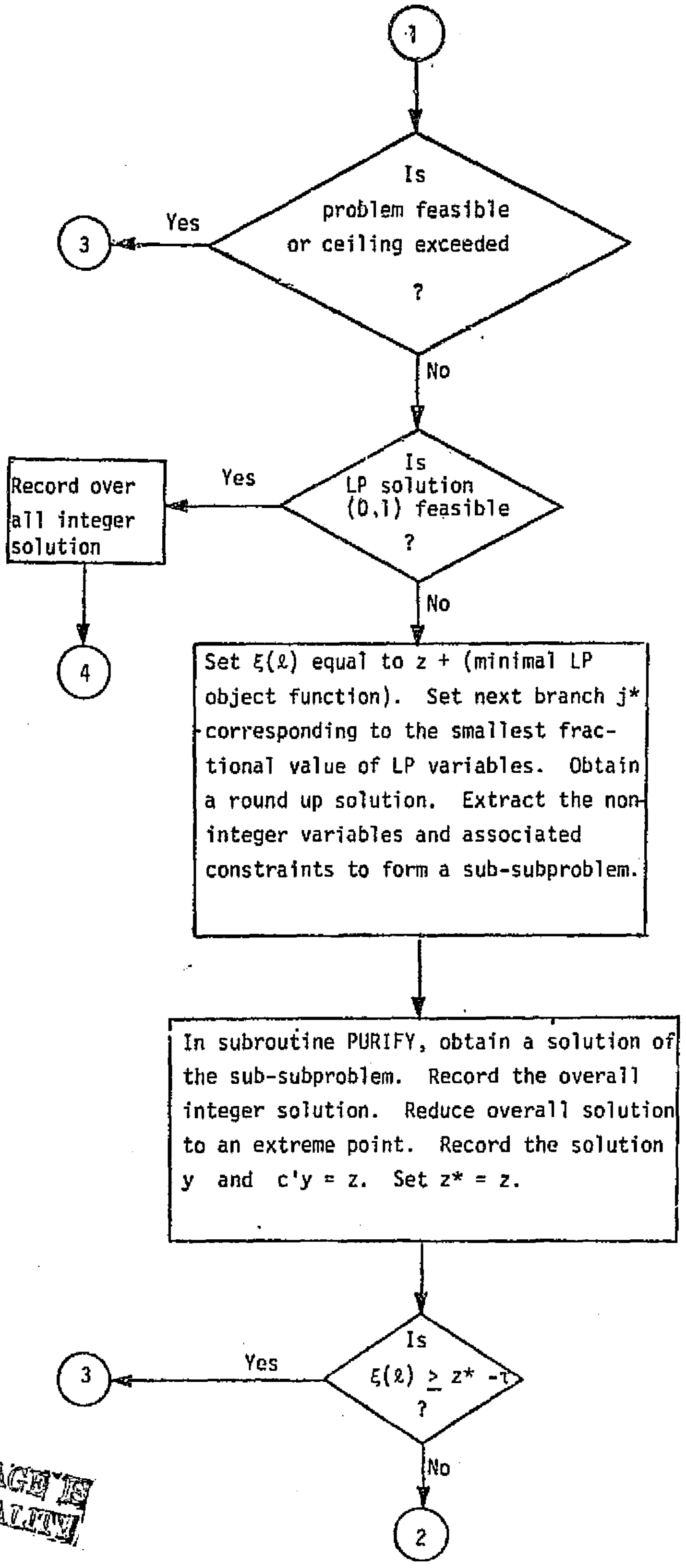
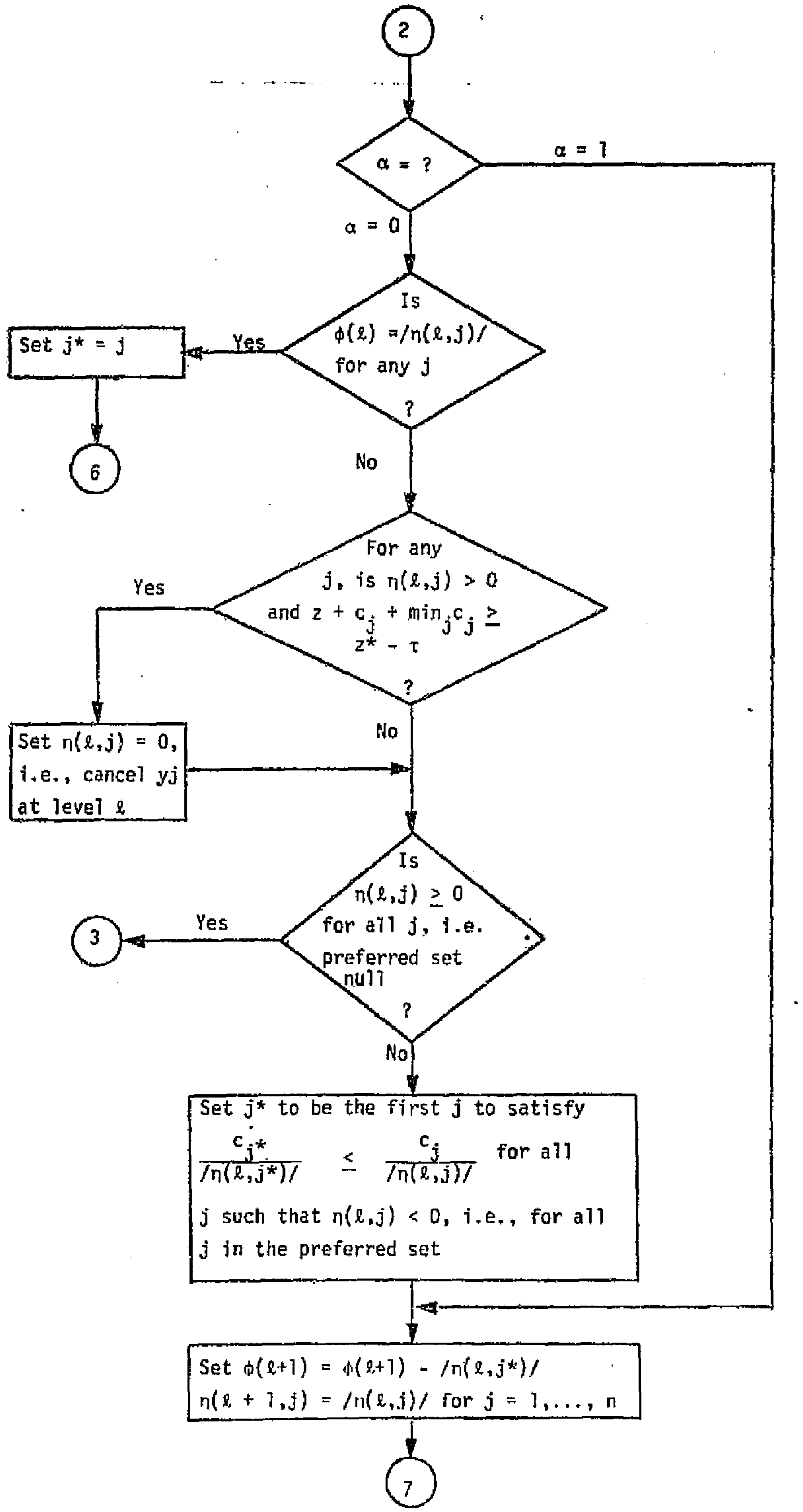

1
Is problem feasible or ceiling exceeded ?
Yes
3
No
Is LP solution (0,1) feasible ?
Yes
Record over all integer solution
4
No
Set $\xi(\ell)$ equal to z + (minimal LP object function). Set next branch j* corresponding to the smallest fractional value of LP variables. Obtain a round up solution. Extract the non-integer variables and associated constraints to form a sub-subproblem.
In subroutine PURIFY, obtain a solution of the sub-subproblem. Record the overall integer solution. Reduce overall solution to an extreme point. Record the solution y and c'y = z. Set z* = z.
Is $\xi(\ell) \geq z^* - \tau$?
Yes
3
No
2

Figure 2.- Continued.



2
α = ?
α = 1
α = 0
Is φ(ℓ) =/η(ℓ,j)/ for any j ?
Yes
Set j* = j
6
No
For any j, is η(ℓ,j) > 0 and z + c_j + min_j c_j ≥ z* − τ ?
Yes
Set η(ℓ,j) = 0, i.e., cancel yj at level ℓ
No
Is η(ℓ,j) ≥ 0 for all j, i.e. preferred set null ?
Yes
3
No
Set j* to be the first j to satisfy c_j* / /η(ℓ,j*)/ ≤ c_j / /η(ℓ,j)/ for all j such that η(ℓ,j) < 0, i.e., for all j in the preferred set
Set φ(ℓ+1) = φ(ℓ+1) − /η(ℓ,j*)/
η(ℓ + 1,j) = /η(ℓ,j)/ for j = 1,..., n
7

Figure 2.- Continued.

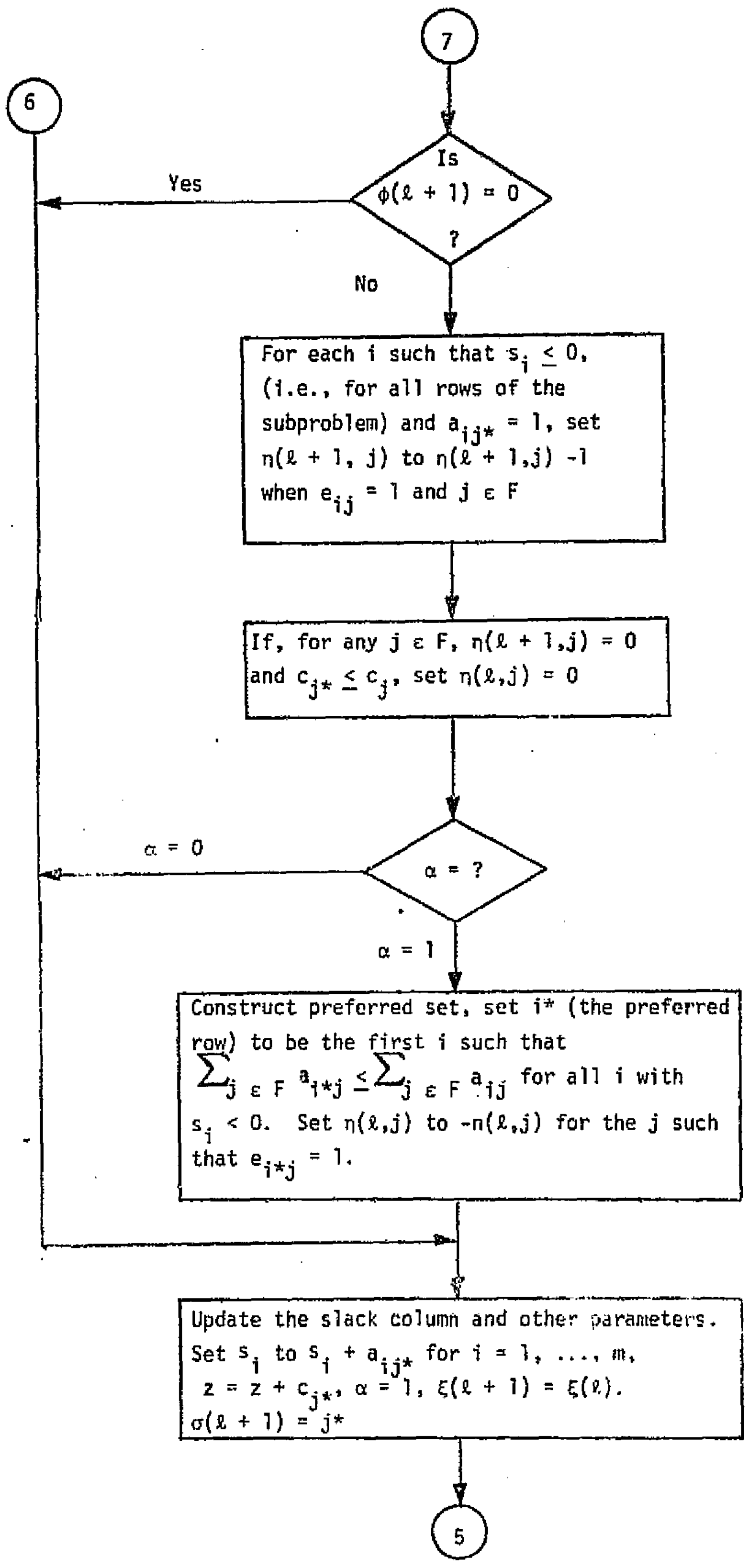

7
6
Is $\phi(\ell + 1) = 0$?
Yes
No
For each i such that $s_i \leq 0$, (i.e., for all rows of the subproblem) and $a_{ij*} = 1$, set $\eta(\ell + 1, j)$ to $\eta(\ell + 1, j) - 1$ when $e_{ij} = 1$ and $j \in F$
If, for any $j \in F$, $\eta(\ell + 1, j) = 0$ and $c_{j*} \leq c_j$, set $\eta(\ell, j) = 0$
$\alpha = ?$
$\alpha = 0$
$\alpha = 1$
Construct preferred set, set i* (the preferred row) to be the first i such that $\sum_{j \in F} a_{i*j} \leq \sum_{j \in F} a_{ij}$ for all i with $s_i < 0$. Set $\eta(\ell, j)$ to $-\eta(\ell, j)$ for the j such that $e_{i*j} = 1$.
Update the slack column and other parameters. Set s_i to $s_i + a_{ij*}$ for $i = 1, \ldots, m$, $z = z + c_{j*}$, $\alpha = 1$, $\xi(\ell + 1) = \xi(\ell)$. $\sigma(\ell + 1) = j*$
5

Figure 2.- Continued.

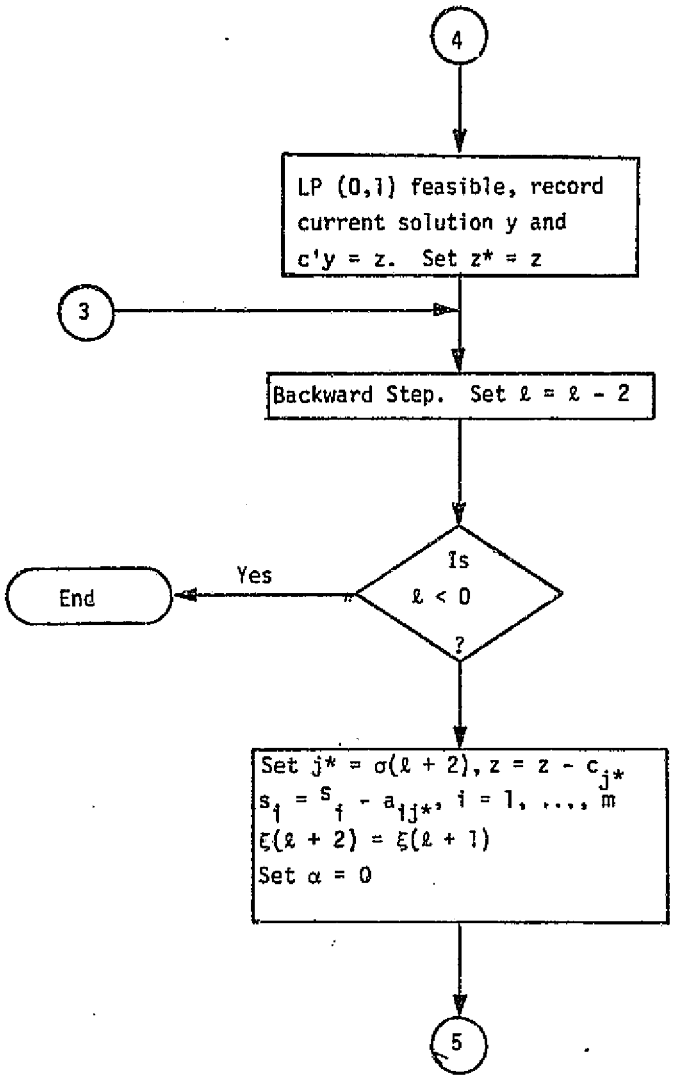

4
LP (0,1) feasible, record current solution y and c'y = z. Set z* = z
3
Backward Step. Set ℓ = ℓ - 2
Is ℓ < 0 ?
Yes
End
Set j* = σ(ℓ + 2), z = z - c_{j*}
s_i = s_i - a_{ij*}, i = 1, ..., m
ξ(ℓ + 2) = ξ(ℓ + 1)
Set α = 0
5

Figure 2 .- Concluded.

5.2 SYMBOL DEFINITIONS

Table IV defines a list of parameters specified in DATA statements in the SCA subprogram. Table V gives the description of all variables used in labeled COMMON.

TABLE IV - PARAMETERS IN DATA STATEMENTS

Parameter name	Dimension	Type	Value	Description
KSTR1	1	I	1 (0)	When KSTR1 = 1, the algorithm always extracts and gets a (0,1) solution from the sub-subproblem. Otherwise, the sub-subproblem is explored only when the LP roundup solution is currently feasible.
KSTR2	1	I	1 (0)	When KSTR2 = 1, the algorithm will attempt to reduce (0,1) feasible solutions to (0,1) feasible extreme points of the associated continuous problem. Otherwise, this attempt will not be made.
KSTR3	1	I	1(0)	When KSTR3 = 1, the search will select branches by the minimum-cost-per-constraint-satisfied rule; otherwise it uses the maximum-number-of-constraints-satisfied rule.
KSTR4	1	I	1 (0)	When KSTR4 = 1, the search will ignore other criteria and branch on the minimum fractional LP value when available. Otherwise, the branch selection rule is determined by KSTR3.

TABLE IV. - PARAMETERS IN DATA STATEMENTS - Concluded

Parameter name	Dimension	Type	Value	Description
KSTR5	1	I	1	When KSTR5 = 1, the algorithm supposes that the user will supply a slack cost, either by setting it directly to MM or by supplying MAXCST, in which case MM is set to 5* MAXCST. If KSTR5 = 0, the algorithm automatically takes the slack cost as 0.
KSTR6	1	I	1	When KSTR6 = 1, LP is performed after forward and backward steps. Otherwise LP is performed only after forward steps.

TABLE V. - VARIABLES IN LABELED COMMON

• COMMON Block Name: C9

Description: C9 retains the information about the interactive selection of the cost coefficient of the objective function and the output of the occurrence table.

Location*	Name	Dimension	Type	Description
2	MM	1	I	Total number of feasible combinations generated by MPLS
51	COSTOP	1	I	Indicator of the choice of cost coefficients on the objective function
54	NOTAB	1	I	Indicator of listing or suppressing the occurrence table output

• COMMON Block Name: C13

Description: C13 retains information about the structure of the constraint matrix which defines the traffic model problem and a flag to trigger the OSARS.

Location	Name	Dimension	Type	Description
1 - 501	NOPERC	501	I	The position vector indicating the number of ones in each column of the constraint matrix
502 - 2501	KARR	2000	I	The position vector indicating the rows which correspond to one entries in the constraint matrix
2502	KM	1	I	The number of rows of the constraint matrix

*The unspecified cells are used for the OSARS.

TABLE V. - VARIABLES IN LABELED COMMON - Continued

• COMMON Block Name: C13 - Continued

Location	Name	Dimension	Type	Description
2503	KN	1	I	The number of columns of the constraint matrix
2504	NOSARS	1	I	A flag indicating the user's choice of the use of the OSARS

• COMMON Block Name: C16

Description: C16 contains information about the location of testing range of each combination.

Location	Name	Dimension	Type	Description
1 - 500	LOCEOW	500	A	A vector to store alphanumerical identification of testing site where a particular mission is to be launched

• COMMON Block Name: C18

Description: C18 contains information about the coefficients and the priorities of those variables in the objective function.

Location	Name	Dimension	Type	Description
1 - 500	KCJ	500	I	The cost coefficients of the objective function is to be minimized in the SCA. They may be defined as feasible combination's OMS weight, one-load factor, or one-payload length
501 - 1000	KPRIOR	500	I	The weight factor to determine each feasible combination's priority.

TABLE V.- VARIABLE IN LABELED COMMON - Continued

● COMMON Block Name: C27

Description: C27 retains various information about the branch-and-bound algorithm used in the SCA.

Location	Name	Dimension	Type	Description
1 - 50	ZLB	50	F	The lower bound of the branch-and-bound method
51 - 3550	KXB	7 × 500	I	The number of unsatisfied constraints of each variable at different levels
3551 - 3650	KYS	100	I	The value of the slack variable
3651 - 4150	KCOM	500	I	A vector to store the column number of those variables in the preferred set
4151 - 4651	KPREF	501	I	A vector to store the row number of the infeasible constraints
4652 - 5151	LBSC	500	I	An indicator of a decision variable's value in the LP solution. LBSC(K) negative means variable K is one.
5152 - 5651	LPBTA	500	I	A vector to store the pivot row of each LP iteration
5652 - 6252	LIDRW	601	I	Scaled cost coefficients of the objective function

TABLE V.- VARIABLE IN LABELED COMMON - Continued

● COMMON Block Name: C28

Description: C28 transmits information between the SET routine and the SIMPLX routine.

Location	Name	Dimension	Type	Description
1	KPV	1	I	Indicator of the number of elements in the preferred set
2	KHP	1	I	Level indicator of the SCA
3	KTEST	1	I	The highest level allowed in the SCA
4	MM	1	I	Cost coefficients of the slack variable of the objective function
5	NLP	1	I	An indicator of the number of times subroutine SIMPLX has been called
6	KSOL	1	I	Indicator of the number of improved solutions reached in the SCA
7	KUNF	1	I	A variable to indicate the feasi-bility of the LP solution
8	KFEAS	1	I	An indicator of the binary feasi-bility of LP solution
9	KZS	1	I	Current value of the objective function
10	KZSTR	1	I	Current upper bound of the objective function
11	J2	1	I	Branching variable in the SCA

TABLE V.- VARIABLE IN LABELED COMMON - Continued

● COMMON Block Name: C28 - Continued

Location	Name	Dimension	Type	
12	K1	1	I	The number of variables in the preferred set
13	NEP	1	I	The number of times an extreme point reduction option is used
14	NEPSC	1	I	The number of times an extreme point reduction is successful
15	NOGO	1	I	A flag indicating infeasibility in subroutine PURIFY
16	LD7	1	I	A flag to print out final information for subroutine PURIFY
17	KSUM	1	I	Current value of the objective function in a sub-subproblem

TABLE V.- VARIABLE IN LABELED COMMON - Concluded

● COMMON Block Name: C29

Description: C29 retains various information about the partial solution of the SCA.

Location	Name	Dimension	Type	Description
1 - 500	KSEQU	500	I	A vector to store the partial solution
501	KSC	1	I	Level indicator
502	KANDO	1	I	A flag indicating that the OSARS can schedule the traffic model when it equals one
503	KNSKD1	1	I	A flag to choose the solution output format for the SCA or the OSARS
504 - 553	KPSKED	50	I	A vector to store the missions scheduled by the OSARS
554	KPRMAX	1	I	Best attainable mission priority
555	KNOSKD	1	I	The number of missions scheduled by the OSARS

5.3 SUBROUTINE DOCUMENTATION

Individual subroutine documentation appears in alphabetical order on the following pages.

SUBROUTINE BISRCH

IDENTIFICATION

Name/Title	- BISRCH (Binary Column Search)
Author/Date	- Han Chang, July 1975
Machine Identification	- UNIVAC 1110
Source Language	- FORTRAN V

PURPOSE

Subroutine BISRCH searches and positions particular row entries for the given column of a constraint matrix. It was written to update the η matrix, generate a preferred set, test for an extreme point, etc.

USAGE

- CALLING SEQUENCE

  CALL BISRCH ($,K,J)

  Arguments:

Parameter Name	In/Out	Dimension	Type	Description
$	Out	1	I	Nonstandard return signal when a particular entry has been found.
K	In	1	I	The column index on which a particular row entry is to be searched
J	In	1	I	Row index on which an entry is to be positioned at a given column K

● Data In/Out

Labeled COMMON (refer to the labeled COMMON description section):

Block Name	Input	Output
C13	1 - 501	
	502 - 2501	

METHOD

● Model

Subroutine BISRCH searches over a column interval of the constraint matrix to locate a particular entry. The output consists of a nonstandard return whenever this entry has been found.

SUBROUTINE FNDFLT

IDENTIFICATION

Name/Title	- FNDFLT (Find Flight)
Author/Date	- Han Chang, July 1975
Machine Identification	- UNIVAC 1110
Source Language	- FORTRAN V

PURPOSE

Subroutine FNDFLT provides the necessary information about a combination by the interactive request from the user. It also saves that information from the traffic model in a scratch file of the user's choice.

USAGE

- CALLING SEQUENCE
  CALL FNDFLT (KZS,LIDRW,IPONT)

  Arguments:

Parameter Name	In/Out	Dimension	Type	Description
KZS	In	1	I	Total number of missions to be output
LIDRW	In	1	I	A vector to store the mission numbers
IPONT	In	1	I	A flag to trigger the output of the statistics of the current flight schedule

- DATA In/Out

  Labeled COMMON (refer to the labeled COMMON description section):

Block Name	Input	Output
C13	2503	

METHOD

Subroutine FNDFLT searches over a data file (logic unit 2) to locate particular missions specified by the user. Detailed information on those combinations are output in an understandable format by calling subroutine DISPLY. Information about the combinations in the traffic model is saved on the scratch file (logic unit 1) at the user's request.

SUBROUTINE PURIFY

IDENTIFICATION

Name/Title	- PURIFY (Purification of LP Solution)
Author/Date	- Han Chang, July 1975
Machine Identification	- UNIVAC 1110
Source Language	- FORTRAN V

PURPOSE

Subroutine PURIFY purifies noninteger linear programming solution into an improved rounded-up integer solution.

USAGE

- CALLING SEQUENCE
  CALL PURIFY
- Data In/Out
  Labeled COMMON (refer to the labeled COMMON description section):

Block Name	Input	Output
C13	1 - 501	
	502 - 2501	
	2502	
C18	1 - 500	
C26	4	
C27	3551 - 3650	
	4151 - 4652	
	4653 - 5154	
	5155 - 5654	
	5655 - 6255	

Block Name	Input	Output
C28	1	
	2	
	6	
	9	
	10	
	13	
	15	15
	17	17
C29	1 - 500 501	1 - 500 501

METHOD

- Model

Whenever the linear programming solution is not integrally feasible, a subproblem is created by eliminating the columns associated with variables having an integer value and those constraints satisfied by these variables. In the subproblem, PURIFY successively sets to one the variable corresponding to the minimum-cost-per-constraints-satisfied ratio until all constraints have been satisfied.

REFERENCE

C. E. Lemke, H. M. Salkin, and K. Spielberg, "Set Covering by Single Branch Enumeration with Linear Programming Subproblems," Operations Research 19, pp. 998-1022 (1971).

SUBROUTINE RIDMOD

IDENTIFICATION

Name/Title	- RIDMOD (Read Input and Mod)
Author/Date	- Han Chang, July 1975
Machine Identification	- UNIVAC 1110
Source Language	- FORTRAN V

PURPOSE

Subroutine RIDMOD provides the selection of flight/combination information for user's reference when he goes through the SCA interactively. Based on this information, the user can make his judgment as to which combination he would add or delete from the traffic model.

USAGE

- CALLING SEQUENCE
  CALL RIDMOD

- DATA In/Out
  Labeled COMMON (refer to the labeled COMMON description section):

Block Name	Input	Output
C31		1 - 4

METHOD

Subroutine RIDMOD collects the user's numerical choices on flight/combination option and sorts them out by using a MOD function. Then subroutine SET uses this information as input to generate the proper flight/combination information.

SUBROUTINE SET

IDENTIFICATION

Name/Title	- SET (Set Covering Algorithm)
Author/Date	- Han Chang, July 1975
Machine Identification	- UNIVAC 1110
Source Language	- FORTRAN V

PURPOSE

Subroutine SET is the main driver of the SCA. SET determines the feasibility of the intermediate solution from subroutine SIMPLX on each level and decides whether it should go forward or backward from the existing node.

USAGE

- CALLING SEQUENCE

  CALL SET (IMODE)

  Arguments:

Parameter Name	In/Out	Dimension	Type	Description
IMODE	In	1	I	Indicator of whether the interactive feature is needed in SET

- DATA In/Out

  Labeled COMMON (refer to the labeled COMMON description section):

Block Name	Input	Output
C13	1 - 501	1 - 501
	502 - 2501	502 - 2501
	2502	2502
	2503	2503

Block Name	Input	Output
C13	2504	2504
	2505	
C18	1 - 500	1 - 500
	501 - 1000	
C26		1
		2
		3
		4
		5
		6
		7
		8
		9
		10
C-27	1 - 50	1 - 50
		51 - 3550
		3551 - 3650
		3651 - 4150
		4151 - 4651
		4652 - 5151
		5152 - 5651
		5652 - 6252
C-28		1
		2
		3
		4
		5
		6
	7	7

Block Name	Input	Output
C28	8	8 9 10 11 16
C29	1 - 500	
	501	501
	502	

METHOD

In each level, subroutine SET determines the subproblem of free variables and sends it to subroutine SIMPLX. SET tests the result of the subproblem which has been solved by SIMPLX, then directs the program to branch forward if there is a possibility of getting a better solution; otherwise, a backward step will be taken.

REFERENCE

C. E. Lemke, H. M. Salkin, and K. Spielberg, "Set Covering by Single Branch Enumeration with Linear Subproblems," Operations Research 19, 998-1022 (1971)

SUBROUTINE SIMPLX

IDENTIFICATION

Name/Title	- SIMPLX (Simplex)
Author/Date	- Han Chang, July 1975
Machine Identification	- UNIVAC 1110
Source Language	- FORTRAN V

PURPOSE

Subroutine SIMPLX solves the linear program associated with a subproblem defined by subroutine SET. This subroutine provides the LP solution over free variables and the information about whether subroutine SET will take a forward or backward step.

USAGE

- CALLING SEQUENCE

  CALL SIMPLX

- Data In/Out

  Labeled COMMON (refer to the labeled COMMON description section):

Block Name	Input	Output
C13	1 - 501	
	502 - 2501	
	2502	
	2503	
	2504	
C18	1 - 500	
C26	4	
	5	
	6	
	7	

Block Name	Input	Output
	8	
	9	
	10	
C27	1 - 50	
	51 - 3550	
	3551 - 3650	
	3651 - 4150	
	4151 - 5151	
	4652 - 5151	
	5152 - 5651	
	5652 - 6252	
C28	1	
	2	
	5	
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	15	
	16	
	17	
C29	501	1 - 500
	502	501

METHOD

A revised dual simplex method has been employed in this routine for optimizing a linear program. This method was designed to accomplish exactly the same function as the original simplex method, but in a way which is more efficient for execution on a digital computer. It computes and stores only the information that is currently needed, and it carries along the essential data in a compact form. In fact, all relevant information at each iteration can be obtained immediately after the inverse of basis has been found. The bookkeeping of a huge conventional simplex tableau becomes less attractive.

Another advantage of this method is that the basic inverse is stored in the product form rather than a matrix form at each iteration to keep the minimal usage of core storage.

REFERENCE

G. Hadley, Linear Programming, Addison-Wesley Co., Inc., Reading, Massachusetts, 1963.

SUBROUTINE TABLE

IDENTIFICATION

Name/Title	- TABLE (Form Occurrence Table)
Author/Date	- Han Chang, July 1975
Machine Identification	- UNIVAC 1110
Source Language	- FORTRAN V

PURPOSE

Subroutine TABLE prints the feasible combination payload occurrence table in a particular year by user's request and translates this table into input format for subroutine SET.

USAGE

- CALLING SEQUENCE
  CALL TABLE (IYEAR)

  Arguments:

Parameter Name	In/Out	Dimension	Type	Description
IYEAR	In	1	I	The year indicator on which data case is based

- DATA In/Out
  Labeled COMMON (refer to the labeled COMMON description section):

Block Name	Input	Output
C9	2	
	51	
	54	

Block Name	Input	Output
C13		1 - 501
		502 - 2501
		2502
		2503
		2505
C16	1 - 500	
C18		1 - 500
		501 - 1000

METHOD

Subroutine TABLE determines and prints a list for each payload of all feasible combinations which include that payload by reading the relevant information from a data file (logical unit 2) which has been generated by the MPLS.

5.4 SAMPLE INPUT/OUTPUT

This sample input/output is to provide the reader with an example of executing the SCA interactively. The procedure to sign on the demand terminal and execute the MPLS is detailed in the SAMPLE User's Guide (ref. 2) and will not be repeated here. All the underlined tutorials are the options which the user may encounter in the SCA execution. Each of the underlined tutorials is accompanied by a section code referring to the location of the option's explanation. The alphabet prior to the section code on each underlined tutorial will correspond to the short description of the user's response. The descriptions are as follows:

(a) The user wants to execute SCA interactively.

(b) The user wants unity cost coefficients so he can get a traffic model with minimum number of missions.

(c) The user wants to see the missions with maximum number of payloads.

(d) The user needs feasible combination number 5 to be included in the traffic model.

(e) The user wants to see the display of Manual Flight/Combination Options.

(f) The user wants the program to continue to find traffic model.

(g) After the traffic model is found, the user needs to see the information on flight number 7.

(h) He does not want to see any more, so he enters zero.

(i) The user wants to have another traffic model.

(j) He wants flights 10 and 11 to be excluded from the traffic model; the missions with maximum number of payloads have been changed.

(k) The user wants the program to continue to find a traffic model as he specified (Notice: flights 10 and 11 did not appear in the traffic model.)

(l) The user is not interested in seeing the information on traffic model.

(m) The user does not want another traffic model.

(n) The user terminates the execution of the SCA.

```
@RUN HCLXIE,1239C-W010-C,FM3-L79197
DATE: 092275       TIME: 170746
>@USE TEMPOR.,FD3-L73486*TEMPOR
READY
>@MAP TEMPOR.MAP,SAMPLE
MAP 0026-09/22-17:09 -(19,)

START=013533, PROG SIZE(I/D)=14225/50394
SYS$*RLIB$. LEVEL 70-1
END OF COLLECTION - TIME 3.571 SECONDS
>@XQT SAMPLE
 INPUT TUG CHARACTERISTICS AND MISSION MODEL DATA:
 (FOR EXAMPLE @ADD SAMPLE.DATA99)
>@ADD TEMPOR.DATA15
 SELECT AN OPTION: ( 3 TO TERMINATE )
>1
 SELECT DISPLAY OPTIONS: ( 7 FOR ALL & 0 FOR NONE )
>4
 SELECT AN OPTION: ( 3 TO TERMINATE )
>2
 SELECT AN ANALYSIS TYPE: (4 FOR NONE)
>2
 INPUT YEAR FOR ANALYSIS: ( 79 TO 91 )
>80
 SELECT INTERACTIVE OPTIONS: (3 FOR NONE)                    (a)  3.2.1
>2
 SELECT PERSONAL DATA BASE TO GENERATE FEASIBLE MISSION: (9 FOR NONE)
>567
 INPUT MAXIMUM NUMBER OF PAYLOADS ALLOWED IN ONE COMBINATION:
>3
 SELECT MISSION TYPE:
              1: FOR INPUT CHANGES TO LIST
              0: NO CONSTRAINTS APPLIED
             -1: APPLY CONSTRAINTS USING LIST
>-1
 SELECT DISCIPLINE MIX:
              1: FOR INPUT CHANGES TO LIST
              0: NO CONSTRAINTS APPLIED
             -1: APPLY CONSTRAINTS USING LIST
>-1
 INPUT 1 TO PRINT MISSION CLASS CODE LIST; OTHERWISE SKIP A LINE
>0
 INPUT 1 TO PRINT PAYLOAD DISCIPLINE MIX LIST; OTHERWISE SKIP A LINE
>0
 **** MPLS STARTED  ************
 ********************  STATISTICAL ANALYSIS FOR 1980  *******************
 TOTAL NUMBER OF COMBINATIONS GENERATED:        24
       NUMBER OF FEASIBLE COMBINATIONS:         11
       NUMBER OF INFEASIBLE COMBINATIONS:       13

 TOTAL ELAPSED TIME:          47
       (ALL TIMES ARE IN MILLISECONDS)
       AVERAGE TIME PER FEASIBLE COMBINATION:      4
       AVERAGE TIME PER GENERATED COMBINATION:      1
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
CHOOSE COST COEFFICIENT FOR EACH FLIGHT:                                        (b) 3.2.2
>1

1980      OCCURRENCE TABLE
     PAYLOAD     MISSIONS
  1)   1000     1     9    10
  2)   1050     2    11
  3)   2000     3     9    11
  4)   2050     4    10
  5)   3001     5
  6)   3002     6
  7)   3051     7
  8)   3052     8
  ** MAX NO. SINGLES =     9

OCCURRENCE TABLE AND SCA INTERFACE REQUIRED     36 MILLISECONDS
CHOOSE CRITERIA FOR FLIGHT/COMBINATION SELECTION: (5 FOR NONE)                  (c) 3.2.3
>1
YOU WILL GET TUTORIALS IN MODE 1.
AVAILABLE MISSIONS WITH MAXIMUM NUMBER OF PAYLOADS:
MISSION             PAYLOADS
    9                  2
   10                  2
   11                  2
    1                  1
    2                  1
CHOOSE MANUAL FLIGHT/COMBINATION OPTION:                                        (d) 3.2.4
>5
AVAILABLE MISSIONS WITH MAXIMUM NUMBER OF PAYLOADS:
MISSION             PAYLOADS
    9                  2
   10                  2
   11                  2
    1                  1
    2                  1
CHOOSE MANUAL FLIGHT/COMBINATION OPTION:                                        (e) 3.2.4
>0
TABLE FOR MANUAL FLIGHT/COMBINATION OPTIONS:
      0: LIST OF MANUAL FLIGHT/COMBINATION OPTIONS
      N: ENTER COMBINATION "N"
     -1: NEW SELECTION CRITERIA ARE DESIRED.
     -2: CONTINUE ON TO TERMINATION.
     -3: VIEW ALL COMBINATIONS SPECIFIED SO FAR.
     -4: VIEW INFORMATION ON COMBINATIONS SPECIFIED SO FAR
     -5: REMOVE LAST SPECIFIED COMBINATION
CHOOSE MANUAL FLIGHT/COMBINATION OPTION:                                        (f) 3.2.4
>-2
ALL PAYLOADS CAN BE SCHEDULED IN THE FOLLOWING  6 MISSIONS.
         5          6          7          8          10          11
TRAFFIC MODEL COST IS          6
  TOTAL ELAPSED TIME IN SET=          95
  TIME PER MISSION IN MILLISEC.=          15
  HIGHEST LEVEL REACHED =  2
DO YOU WISH TO SEE INFORMATION ON THESE MISSIONS?                               (g) 3.2.5
          0: NONE
         -1: PRINT ALL
         -2: PRINT ALL AND SAVE ON SCRATCH FILE
         -3: SAVE ON SCRATCH FILE ONLY
         N : ENTER MISSION "N"
```

```
>7

 FLT. NO.    7     LAUNCH SITE: ETR

  PAYLOADS:    LS-01     LCR  A
                3051
  SHUTTLE SEQUENCE        3-R
  ALTITUDE                300.
  INCLINATION             28.5
  TOTAL LENGTH DOWN:  13.     TOTAL WEIGHT DOWN:    632.0
  PAYLOAD MARGIN: 64318.    LOAD FACTOR:  .01049
  SHUTTLE DELTAV:  1275.
>0                                                              (h)  3.2.5
 INPUT 1 IF YOU WANT A DIFFERENT SCHEDULE: OTHERWISE SKIP A LINE   (i)  3.2.6
>1
 WHICH MISSIONS DO YOU WANT OMITTED ?                              (j)  3.2.6
>11
>10
>0
 AVAILABLE MISSIONS WITH MAXIMUM NUMBER OF PAYLOADS:
 MISSION              PAYLOADS
     9                   2
     1                   1
     2                   1
     3                   1
     4                   1
 CHOOSE MANUAL FLIGHT/COMBINATION OPTION:                          (k)  3.2.4
>-2
 ALL PAYLOADS CAN BE SCHEDULED IN THE FOLLOWING  7 MISSIONS.
          2          4          5          6          7          8          9
 TRAFFIC MODEL COST IS            7
   TOTAL ELAPSED TIME IN SET=        172
   TIME PER MISSION IN MILLISEC.=          24
   HIGHEST LEVEL REACHED =   1
 DO YOU WISH TO SEE INFORMATION ON THESE MISSIONS?                 (ℓ)  3.2.5
             0: NONE
            -1: PRINT ALL
            -2: PRINT ALL AND SAVE ON SCRATCH FILE
            -3: SAVE ON SCRATCH FILE ONLY
             N : ENTER MISSION "N"
>0
  STATISTICS FOR CURRENT FLIGHT SCHEDULE

       AVERAGE NUMBER OF PAYLOADS PER FLIGHT = 1.14
       TOTAL NUMBER OF TUGS REQUIRED =    0
       TOTAL NUMBER OF INITIAL OMS KITS REQUIRED =      0
       TOTAL NUMBER OF SECOND AND THIRD OMS KITS REQUIRED =      0
 INPUT 1 IF YOU WANT A DIFFERENT SCHEDULE: OTHERWISE SKIP A LINE   (m)  3.2.6
>0
 SELECT AN OPTION: < 3 TO TERMINATE >                              (n)  3.2.7
>3
>
```

6. REFERENCES

1. Lemke, C. E., Salkin, H. M. and Spielberg, K.: Set Covering by Single-Branch Enumeration with Linear-Programming Subproblems, Operations Research, Vol. 19, 1971, pp. 998-1022.

2. Williams, J.: SAMPLE User's Guide, Rev. 2, LEC-6642, Aug. 8, 1975.

3. G. Hadley, Linear Programming, Addison-Wesley Co., Inc. Reading, Massachusetts, 1963.