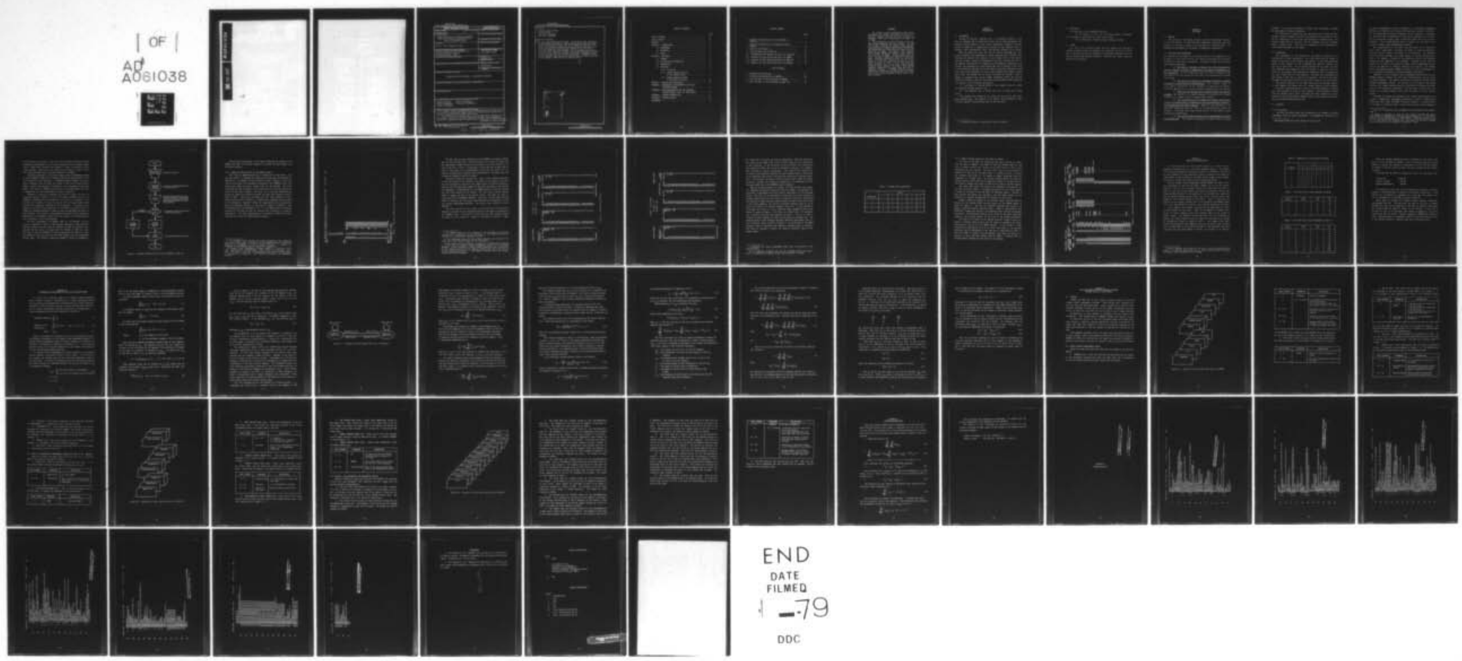
AD-A061 038 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 9/2
ANALYSIS AND USE OF AN INTEGER PROGRAMMING MODEL FOR OPTIMALLY --ETC(U)
NOV 78 P L PRICE, D W SMITH
UNCLASSIFIED DTNSRDC-78/102 NL

| OF |

AD
A061038



END
DATE
FILMED
1-79
DDC

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DTNSRDC-78/102	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ANALYSIS AND USE OF AN INTEGER PROGRAMMING MODEL FOR OPTIMALLY ALLOCATING FILES IN A MULTIPLE COMPUTER SYSTEM		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Perry L. Price, Douglas W. Smith		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS David W. Taylor Naval Ship Research and Development Center, Code 187 Bethesda, Maryland 20084		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (See reverse side)
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE November 1978 13. NUMBER OF PAGES 59
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Networks
Linear Programming
Integer Programming
Binary Programming
Non-Linear Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An integer linear programming model which determines the optimal allocation of files in a multiple computer network has been analyzed to determine under what circumstances the model can be used.

The feasibility of using the model on the CDC 6000 series computers was also studied. The use of the model requires an integer linear programming system and a computerized means of generating and formatting the large amount

(Continued on reverse side)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

78 11 06 095

(Block 10)

Program Element 62760N
Project F53531
Task Area TF53531BOI
Work Unit 1-1800-002

(Block 20 continued)

of input data required by the model. Two integer linear programming systems, ARRIBA and APEX III, were analyzed for use with the model. Either system could be used. The use of ARRIBA was limited to relatively small problems (problems in which the sum of the number of files and computers is ten or less). APEX III could be used to solve both small and large problems, but its use with large problems required costly computer run times. Both systems required essentially the same input data but in different formats. A FORTRAN computer program, DATASUP, was developed to generate the data input for either system (ARRIBA or APEX III) at the user's option.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

An integer linear programming model which determines the optimal allocation of files in a multiple computer network has been analyzed to determine under what circumstances the model can be used.

The feasibility of using the model on the CDC 6000 series computers was also studied. The use of the model requires an integer linear programming system and a computerized means of generating and formatting the large amount of input data required by the model. Two integer linear programming systems, ARRIBA and APEX III, were analyzed for use with the model. Either system could be used. The use of ARRIBA was limited to relatively small problems (problems in which the sum of the number of files and computers is ten or less). APEX III could be used to solve both small and large problems, but its use with large problems required costly computer run times. Both systems required essentially the same input data but in different formats. A FORTRAN computer program, DATASUP, was developed to generate the data input for either system (ARRIBA or APEX III) at the user's option.

SECTION 1
INTRODUCTION

1.1 BACKGROUND

Systems of multiple computers are of increasing interest to the Navy as a possible way of reducing the total cost of data processing. Such computer systems consist of two or more independent computers interconnected by means of a communications system so that resources such as hardware, data (information) files, and software systems can be shared.

When the same data files are used by several geographically separated computers, the combination of these computers into a network would eliminate the need for each computer to store common files. If the operating cost of such a network is determined by the cost to store each file at one computer and the cost to transmit the files among the computers of the network, the operating cost would depend upon the allocation of the files in the network for storage. If the allocation is further restricted so that each file can be available for use at each computer site within a given time bound, the problem can be stated succinctly as follows: Given a number of computers that process common information files, how can the files be allocated so that the allocation yields minimum overall operating cost subject to the following constraints:

(1) the amount of storage needed at each computer does not exceed the available storage capacity, and

(2) the expected time to access each file is less than a given bound.

This problem was studied by Dr. Wesley W. Chu of the Bell Telephone Laboratories. The results of his study[1*] were published in 1969. A revised edition[2] was published later in the same year.

---

*A complete listing of references is given on page 53.

1.2 OBJECTIVE

The objectives of the DTNSRDC study were:

(1) To analyze Dr. Chu's file allocation model to determine the circumstances under which the model can be used.

(2) To determine a feasible means of using the model.

1.3 SCOPE

Dr. Chu's file allocation model did not consider the sharing of computer programs among the computers of a network, nor did it include any of the many technical and managerial difficulties which must be overcome in establishing computer networks. Therefore this report does not address these problems.

SECTION 2
THE MODEL

2.1 GENERAL

This section of the report presents a general nontechnical analysis of Dr. Chu's model. It describes the assumptions, objectives, and constraints of the model as well as two techniques for using the model. The mathematical development of the model is summarized in Appendix A.

2.2 OBJECTIVE AND ASSUMPTIONS

The objective of the model is to allocate files to the computers of the network in a way which produces the lowest operating cost and still satisfies all the stated constraints. The model is based on the following primary assumptions:

(a) <u>The time to transmit a request for a file from one computer to another is short in comparison to the time for transmission of the actual file</u>. If the transmission times for both the request for the file and for the file itself are very short, the correctness of the model still holds.

(b) <u>Each pair of computers is assumed to be able to transmit information in both directions simultaneously</u>. This feature, known as fully duplex operations, is not uncommon in communication systems. The model applies only to multiple computer networks with this feature.

(c) <u>High priority messages are transmitted before low priority messages</u>. The capability to transmit messages in accordance with assigned priorities is a common feature of communication systems.

(d) <u>The time required for a computer to attach a file stored locally is usually small in comparison to the time required to receive a file transmitted from another computer</u>. Implied in this assumption is the additional assumption that all files in the multicomputer network are on-line. If off-line files were allowed as part of the network, access time would be significant.

(e) <u>The file accessing process can be approximated by a Poisson distribution</u>. The Poisson distribution is widely used in queuing problems. It is quite reasonable to assume that file usage is random and approximately Poisson distributed.

The objective of minimizing the operating cost of the multiple computer network considers the costs associated with the storage of files and the tranmissions of both files and information to modify files. The original publication of the model did not consider costs for transmissions to modify files. This cost is considered in the implementation of the model discussed in this report.

2.3 CONSTRAINTS

The achievement of the objective of the model is subject to the satisfaction of three constraints. The first deals with the number of copies of each file to be stored in the network. In his first published paper[1] on this subject, Dr. Chu formulated the model to consider only one copy of each file. In the revised edition[2] of the paper, the one-copy constraint was relaxed so that the number of copies of each file stored in the network could range from a minimum of one to a maximum of one copy of each file for each computer. The model user determined how many copies of each file to store in the network. The use of the model discussed in this report allows for one copy of each file in the network.

The second constraint of the model deals with the storage capacity of each computer and insures that the storage capacity is not exceeded.

The third constraint places a limit on the time allowable for making each file available for use at each computer. This constraint insures that any computer will be able to request and start receiving the transmission of a file from another computer within a time interval not exceeding the maximum allowable access time.

2.4 SOLUTION

2.4.1 Discussion

The file allocation model was formulated as an integer non-linear programming model in binary variables.* (A mathematical description of

---

*Variables which take on the values of 0 and 1 only.

non-linear programming is given in Appendix A). Non-linear programming problems are usually very complex and sometimes cannot be solved at all. By exploiting the binary properties of the variables, a technique was introduced which transformed the non-linear formulation of the model into an equivalent integer linear formulation which was also in binary variables. This transformation made it possible for the model to be solved by integer linear programming techniques.

The transformation of the model from non-linear to an equivalent linear model significantly increased the number of equations making up the model and the number of variables in the equations. For seemingly small allocation problems, the model consists of several hundred equations.* For example, a problem of allocating ten files to six computers consists of 856 equations in 331 unknowns. Problems of this size and larger are too large to be solved by integer linear programming routines that do not make use of auxiliary computer core capacity. Two integer linear programming systems, ARRIBA and APEX III, were investigated for use as solution techniques for solving problems by the model.

ARRIBA** is an all integer linear programming system written in the FORTRAN computer language and, with minor modifications, can be run on any major computer. The system was written to handle relatively small integer programming problems. An analysis of the coding of the system revealed that larger problems could be handled by increasing the array sizes in the dimension statements. The increase in array sizes is limited by available computer core since the system is an all in-core system.

The APEX III System was developed by the Control Data Corporation to provide a mathematical programming capability to users of Control Data Corporation 6000, 7000, and CYBER 170 series computers. A unique feature of the APEX III System is that it can be run as an all in-core system or

---

* The use of equations here encompasses both equalities and inequalities.

** ARRIBA is available to users of CDC computers through VIM Incorporated (User's Organization for Control Data Corporation). It is identified by Catalog Identification H1 UTEX ARRIBA. Others may obtain a tape of the source deck, test problems and a user manual by writing to Control Data Corporation, Box O, Minneapolis, Minnesota 55440.

as an out-of-core system. The out-of-core system provides the same mathematical processing as the in-core system but has the added capability of using disk, extended core storage, or large core memory as additional storage. This out-of-core system can, therefore, be used to solve much larger problems than an in-core system such as ARRIBA.

The data requirements for solving integer linear programming problems by either ARRIBA or APEX III are similar although the data formats for the two systems are quite different. A FORTRAN program, DATASUP, was developed to generate the data input for either of the two solution systems. Figure 1 shows the sequence of activities to be performed in using the model. The output of DATASUP is printed on a file named TAPE7 for input to ARRIBA and on a file named TAPE6 for input to APEX III.

DATASUP is limited to problems in which the number of files or computers does not exceed 15; i.e., the largest problem which can be handled by DATASUP is one involving 15 files and 15 computers. The solution of a problem of this size by the file allocation model would involve 6330 constraint equations in 1801 unknowns. A problem of this magnitude is within the capability of the APEX III system, which is capable of handling problems as large as those having 8000 constraints in 2500 unknowns. The solution of the maximum size which could be handled by DATASUP was not undertaken during this study because of the high computer cost, estimated to be at least $500.00.

The data requirements for ARRIBA, APEX III, and DATASUP are discussed in Appendix B, which is specifically addressed to user personnel who will prepare the data cards for DATASUP and make the computer runs. Although the user does not prepare data for ARRIBA or APEX III, since this task is done by DATASUP, the data requirements are given for these two systems so that the user will have some insight into what the DATASUP routine does. The computer coding for DATASUP is given in APPENDIX D.

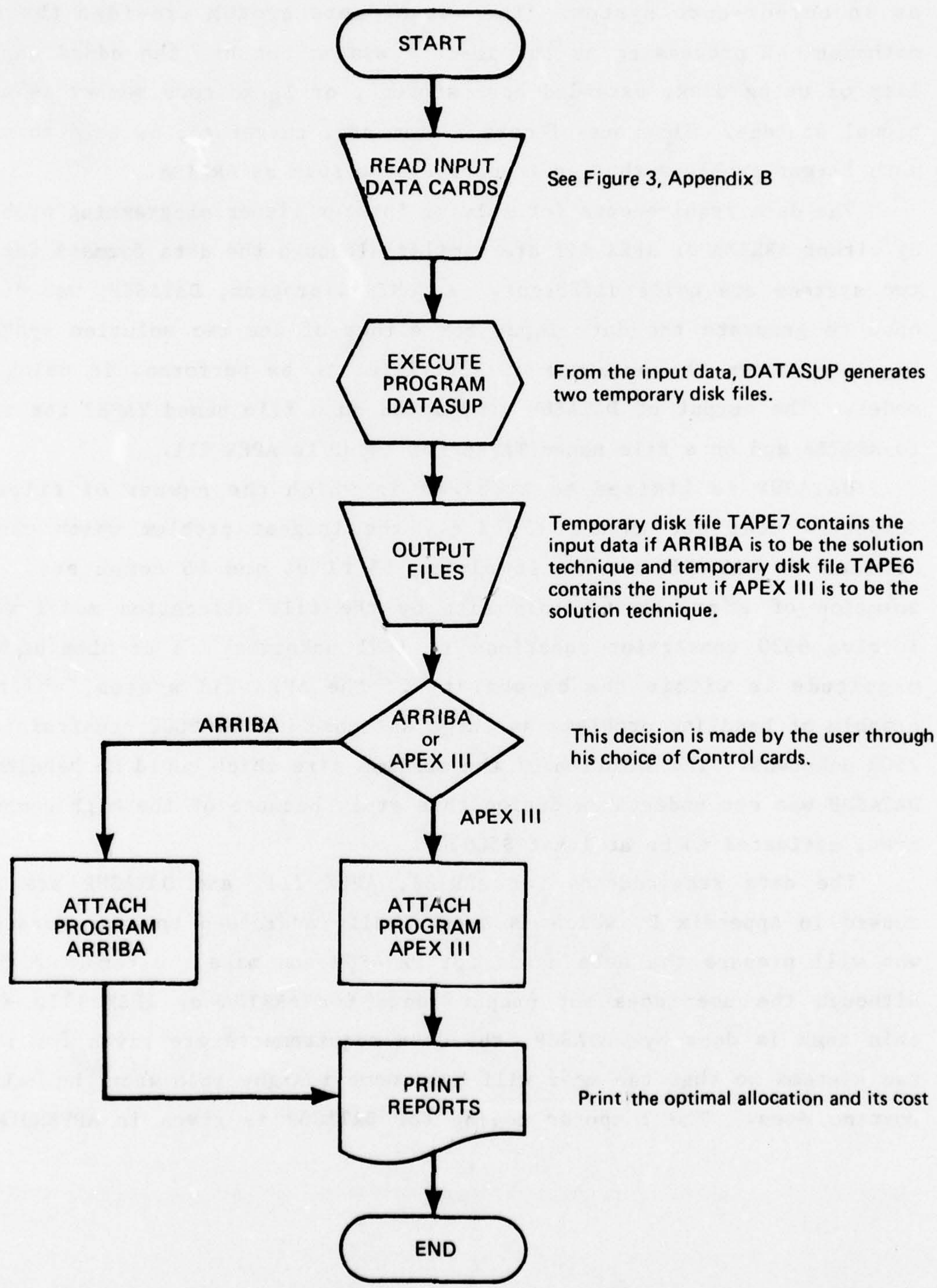


Figure 1 - Problem Solving Steps for Either ARRIBA or APEX III

The next two subsections of this report describe the outputs of the ARRIBA and APEX III solution systems in solving the same sample file allocation problem.

2.4.2 Sample Problem Solution by the ARRIBA System

The output of ARRIBA begins with a listing of the input data. (The user may choose to suppress the printing of these data as described in Appendix B). An example of this listing is shown in Figure 2. The first line of print gives the information contained on the CONTROLS card*. The second line shows the user's chosen title for the problem. The third line shows the data contained on the header card for the rows identification** section which follows. The first row of the rows identification section is always the objective function row which will always be identified as "OBJ". Other rows are named with the first character as "R", which stands for Row, followed by a number which indicates the row number of the constraint. Appendix C shows the organization of the problem from which the numbering sequence of the rows and columns is obtained. Each row identification is preceeded by "+", "-", or "blank space" to indicate relations of less than or equal to, greater than or equal to, or equality respectively. The rows identification is followed by "EOR" which indicates the end of the rows identification section.

---

* The "CONTROLS card" contains the values assigned to the parameters which govern what output is printed and the maximum number of iterations permitted in attempting to reach an optimal solution. A complete description of the CONTROLS card is presented in Appendix B.

** Integer linear programming problems consist of a system of relations (equalities or inequalities). Each relation is referred to as a row. Each term of each row is identified by its row and column number of the system of relations. (For detailed discussion, see Appendixes A and B).

```
 *** CONTROL CARD *** CONTROLS,LIST=1,OBJ VALUE EVERY    1,ACTIVITY LIST EVERY     1,PIVOT LIMIT      100
TITLE.OR CHU-FILE ALLOCATION MODEL TEST CASE--TEST CASE
ROW ID
           +OBJ
           +R1
           +R2
           +R3
           +R4
           +R5
           +R6
           +R7
           +R8
            R9
            R10
           +R11
           +R12
EOR
MATRIX
      RHS    R1     6000.000000
      RHS    R3     2000.000000
      RHS    R7        1.000000
      RHS    R8        1.000000
      RHS    R9        1.000000
      RHS    R10       1.000000
      RHS    R11      11.000000
      RHS    R12      11.000000
      C1     OBJ     790.000000
      C1     R3      603.000000
      C1     R5       -1.000000
      C1     R7        1.000000
      C1     R9        1.000000
      C1     R11      10.000000
      C2     OBJ     580.000000
      C2     R5       -1.000000
      C2     R7        1.000000
      C2     R10       1.000000
      C2     R11      10.000000
      C3     OBJ     930.000000
      C3     R1     3005.000000
      C3     R6       -1.000000
      C3     R8        1.000000
      C3     R9        1.000000
      C3     R12      10.000000
      C4     OBJ     580.000000
      C4     R6       -1.000000
      C4     R8        1.000000
      C4     R10       1.000000
      C4     R12      10.000000
      C5     R3        3.000000
      C5     R5        2.000000
      C5     R7       -1.000000
      C6     R1        5.000000
      C6     R6        2.000000
      C6     R8       -1.000000
PROBLEM HAS  15 ROWS,   7 COLUMNS
EOR
*** CONTROL CARD ***  IPSC
*** CONTROL CARD ***  ARRIBA
```

Figure 2 - Listing of Input Data for Problems Solved by ARRIBA

The next line of print contains the word "MATRIX"; the matrix values* start on the following line. The matrix values are printed by column with the right-hand** column printed first followed by the other columns of the matrix from left to right. Column names (column name and variable name as used in this report are interchangeable) start with the letter "C" followed by a number which indicates the column number in the matrix when the columns are numbered from left to right.

The matrix values are followed by a line of print showing the number of rows and columns of the problem. The number of rows shown on this line does not necessarily agree with the number shown in the Row ID Section, because this line gives the number of rows after the input of the problem has undergone a transformation by the ARRIBA System to put the problem into the form required by the system. The next line of print contains "EOR" to signal the end of the matrix values. The next two lines of print start with "***CONTROL CARD***". The first of these lines contains "IPSC" to indicate that the Gomory Algorithm*** will be employed to solve the integer programming problem. The second line contains the word "ARRIBA".

The next section of the output is the "Activity List Reports" (Figure 3) printed for the iterations chosen by the user as indicated on the CONTROLS Card. The values for each row and each column variable are given for each iteration. Figure 3 (iteration 3) shows R1 = 6000,

---

* The MATRIX values are the elements of the rectangular array which contains the coefficients of the variables in the objective function and the set of constraints composing the model.

** The right-hand column is the column containing the values on the right side of the relation signs of the constraints.

*** The Gomory solution technique is known as a cutting-plane algorithm. It starts with the optimal linear programming solution of the integer problem. At each iteration it adds a linear constraint that is satisfied by any integer solution to the original problem but that rules out the current non-integer solution. This method continues until an integer-valued solution is obtained.

INTEGER PROGRAM *** IPSC *** ARRIBA SYSTEM

ACTIVITY LIST

IDENT OR CHU
ITERATION 3

OBJECTIVE VALUE -1440
ITERATING

COLUMN	ACTIVITY	COLUMN	ACTIVITY	COLUMN	ACTIVITY	COLUMN	ACTIVITY
R1	6000	R2	5400	R3	0	R4	0
R5	2000	R6	1800	R7	0	R8	0
R9	200	R10	179	R11	1598	R12	1800
R13	4800	R14	5400	R15	0	R16	0
R17	16	R18	18	R19	160	R20	179
R21	0	R22	0	R23	200	R24	200
R25	200	R26	200	R27	0	R28	0
R29	200	R30	200	R31	1	R32	1
R33	1	R34	1	R35	0	R36	0
R37	0	R38	0	R39	0	R40	0
R41	0	R42	0	R43	0	R44	0
R45	0	R46	0	R47	0	R48	0
R49	0	R50	0	R51	0	R52	0
R53	0	R54	1	R55	0	R56	0
R57	1	R58	0	R59	1	R60	1
R61	0	R62	0	R63	0	R64	0
R65	1	R66	1	R67	1	R68	1
R69	1	R70	1	R71	1	R72	1
R73	1	R74	1	R75	1	R76	1
R77	1	R78	1	R79	1	R80	1
R81	1	R82	1	R83	1	R84	0
R85	1	R86	1	R87	0	R88	1
R89	0	R90	0	R91	0	R91	0
R92	1	R92	-1	R93	1	R93	-1
R94	1	R94	-1	R95	0	R95	0
R96	1	R97	11	R98	1	C1	1
C2	0	C3	0	C4	0	C5	0
C6	0	C7	0	C8	0	C9	0
C10	0	C11	0	C12	0	C13	0
C14	0	C15	1	C16	0	C17	0
C18	0	C19	0	C20	0	C21	0
C22	0	C23	0	C24	0	C25	0
C26	0	C27	0	C28	0	C29	0
C30	0	C31	0	C32	0	C33	0
C34	0	C35	0	C36	0	C37	0
C38	0	C39	0	C40	0	C41	0
C42	0	C43	0	C44	0	C45	0

ITERATION, 4 OBJECTIVE VALUE, -1708

Figure 3 - Activity List Reports

Figure 3 (Continued)

INTEGER PROGRAM *** IPSC *** ARRIBA SYSTEM

ACTIVITY LIST

IDENT OR CHU OBJECTIVE VALUE -2034

ITERATION 8 ITERATING

COLUMN	ACTIVITY	COLUMN	ACTIVITY	COLUMN	ACTIVITY	COLUMN	ACTIVITY
R1	6000	R2	5400	R3	0	R4	0
R5	2000	R6	1800	R7	0	R8	0
R9	200	R10	178	R11	1598	R12	1400
R13	4800	R14	4196	R15	0	R16	0
R17	16	R18	14	R19	160	R20	138
R21	0	R22	0	R23	120	R24	200
R25	116	R26	200	R27	0	R28	0
R29	120	R30	200	R31	1	R32	2
R33	1	R34	1	R35	1	R36	0
R37	0	R38	1	R39	1	R40	0
R41	0	R42	0	R43	1	R44	0
R45	0	R46	1	R47	0	R48	1
R49	0	R50	1	R51	1	R52	0
R53	0	R54	1	R55	1	R56	1
R57	0	R58	0	R59	1	R60	1
R61	0	R62	-1	R63	0	R64	0
R65	0	R66	1	R67	1	R68	0
R69	0	R70	1	R71	1	R72	1
R73	0	R74	1	R75	1	R76	0
R77	1	R78	0	R79	1	R80	0
R81	0	R82	1	R83	1	R84	0
R85	0	R86	0	R87	0	R88	1
R89	0	R90	0	R91	0	R91	0
R92	0	R92	0	R93	0	R93	0
R94	0	R94	0	R95	0	R95	0
R96	0	R97	10	R98	0	C1	1
C2	0	C3	1	C4	0	C5	0
C6	0	C7	0	C8	0	C9	1
C10	0	C11	0	C12	1	C13	0
C14	0	C15	1	C16	0	C17	0
C18	0	C19	0	C20	0	C21	0
C22	0	C23	0	C24	0	C25	0
C26	0	C27	0	C28	0	C29	0
C30	0	C31	0	C32	0	C33	0
C34	0	C35	0	C36	0	C37	0
C38	0	C39	0	C40	0	C41	0
C42	1	C43	0	C44	0	C45	0

OPTIMUM SOLUTION

R2 = 5400, and so on until all rows are identified. These row values are the values on the right side of the relation signs for the constraints at that iteration. At the conclusion of the row values, the column variables values are shown. The C1 and C2 variables in Figure 3, for example, take on values of 1 and 0, respectively. Also shown at each iteration is the value of the objective function for that iteration and, at the bottom of the iteration, the value of the objective function for the next iteration. The Activity List Reports will be printed until either an optimal solution is reached, the problem is determined to be infeasible,* or the iteration limit set on the CONTROLS card is reached.

When an optimum solution is reached, the last Activity List Report will identify the objective function value as optimum. The optimal file allocation is determined from this Activity List Report in the following manner: The first N x M variables** starting with C1 are the variables of interest where N and M are the number of computers and files, respectively. The first M of these variables indicate which of these M files are stored on the first computer, the second M variables indicate which of the M files are stored on the second computer, and so on. In the problem of allocating five files to three computers whose optimum Activity List Report is shown in Figure 3, the first five times three or 15 variables are of interest and have been displayed in Table 1. The 1's in Table 1 indicate on which computer the files are allocated for minimum cost. The absolute value of the objective function indicates the minimum cost for that allocation, which is $2034 in Figure 3. (A peculiarity of the ARRIBA system is that it identifies the objective function value as negative. The value of interest is simply the absolute (positive) value of the number shown.)

---

* Infeasible in linear programming means that no solution to the problem exists.

** The remaining variables are the new variables which were introduced to transform the original model from non-linear to linear.

TABLE 1 - OPTIMAL FILE ALLOCATION

COMPUTERS	FILES				
	1	2	3	4	5
1	1	0	1	0	0
2	0	0	0	1	0
3	0	1	0	0	1

2.4.3 Sample Problem Solution by the APEX III System

The user of the APEX III system has many more options for output than the user of ARRIBA. The APEX III user can process APEX III with only a single control card, known as the Solve Card, or with a user developed control program. The most direct way to use the APEX III system is by the use of the Solve Card. It allows the user to control the execution and output of the system by simply selecting values for parameters on a single card. The Control Program allows the user more options than the Solve Card and is especially useful if the user wants to save files or prepare the output of APEX III for input to another program.

The Solve Card option was found to be completely satisfactory for the file allocation model. A detailed description of the output reports of APEX III is given in the CDC APEX III Reference Manual. Because so many reports and options are available, only the output report for the solution to a specific problem will be discussed in this report.

Figure 4 shows an example of the solution report from the APEX III system for the problem whose solution report for the ARRIBA system is shown in Figure 3. The user is primarily concerned with column 5 of Figure 4 and the objective function value. Column 5, identified as "COL ACTIVITY", contains the values of the variables which indicate the optimum allocation. The objective function value, found in the upper right-hand corner of the solution report, is the cost of the optimum allocation.

The first N x M values in the "Column Activity" Column are the variables of interest where N and M are the number of computers and files, respectively, just as for the ARRIBA system. If the values of these variables are displayed in an N x M table like Table 1, the optimum allocation may be determined in the same way as described for Table 1. Note that APEX III prints only the decimal point when the value of a variable is zero, and that column three of Figure 4 contains "BV" to indicate binary variable for the first N x M variables.

DATE 12/09/76 TIME 14.18.22 C O L U M N S APEX-III 1.000 PAGE 1

PRINT OPTION = COMPLETE OUTPUT VALUE OF OBJECTIVE = 2034.00000
NAME = ILEALLOC OBJ = OBJ RHS = RHS BND = BOUNDS RPSOBJ = 1.0000 RPSRHS = 1.0000
DIR = MINIMIZE COBJ = CRHS = RNG = RPCHOBJ = 0.0000 RPCHRHS = 0.0000

NUMBER	NAME	TYPE	STATUS	COL ACTIVITY	OBJ COEF	BND LOWER	BND UPPER	MARGINAL	
1	COL1	BV	UPPER	1.00000	720.00000	.	1.00000	-210.00000	
2	COL2	BV	LOWER	.	548.00000	.	1.00000	140.00000	
3	COL3	BV	ACTIVE	1.00000	338.00000	.	1.00000	.	
4	COL4	BV	ACTIVE	.	338.00000	.	1.00000	.	
5	COL5	BV	ACTIVE	.	1000.00000	.	1.00000	.	
6	COL6	BV	ACTIVE	.	930.00000	.	1.00000	.	
7	COL7	BV	ACTIVE	.	408.00000	.	1.00000	.	
8	COL8	BV	LOWER	.	548.00000	.	1.00000	210.00000	
9	COL9	BV	UPPER	1.00000	58.00000	.	1.00000	-280.00000	
10	COL10	BV	ACTIVE	.	1000.00000	.	1.00000	.	
11	COL11	BV	LOWER	.	1070.00000	.	1.00000	840.00000	
12	COL12	BV	UPPER	1.00000	198.00000	.	1.00000	-140.00000	
13	COL13	BV	ACTIVE	.	268.00000	.	1.00000	.	
14	COL14	BV	LOWER	.	338.00000	.	1.00000	70.00000	
15	COL15	BV	LOWER	1.00000	720.00000	.	1.00000	420.00000	ARB BV
16	COL16	PL	ACTIVE	.50000	.	.	+INF	.	
17	COL17	PL	ACTIVE	1.00000	.	.	+INF	.	
18	COL18	PL **	LOWER	.	.	.	+INF	.	
19	COL19	PL	ACTIVE	.	.	.	+INF	.	
20	COL20	PL	ACTIVE	.50000	.	.	+INF	.	
21	COL21	PL **	LOWER	.	.	.	+INF	.	
22	COL22	PL	ACTIVE	.	.	.	+INF	.	
23	COL23	PL **	LOWER	.	.	.	+INF	.	
24	COL24	PL **	LOWER	.	.	.	+INF	.	
25	COL25	PL **	LOWER	.	.	.	+INF	.	
26	COL26	PL **	LOWER	.	.	.	+INF	.	
27	COL27	PL **	LOWER	.	.	.	+INF	.	
28	COL28	PL	ACTIVE	.	.	.	+INF	.	
29	COL29	PL **	LOWER	.	.	.	+INF	.	
30	COL30	PL **	LOWER	.	.	.	+INF	.	
31	COL31	PL **	LOWER	.	.	.	+INF	.	
32	COL32	PL **	LOWER	.	.	.	+INF	.	
33	COL33	PL **	LOWER	.	.	.	+INF	.	
34	COL34	PL **	LOWER	.	.	.	+INF	.	
35	COL35	PL **	LOWER	.	.	.	+INF	.	
36	COL36	PL **	LOWER	.	.	.	+INF	.	
37	COL37	PL **	LOWER	.	.	.	+INF	.	
38	COL38	PL **	LOWER	.	.	.	+INF	.	
39	COL39	PL **	LOWER	.	.	.	+INF	.	
40	COL40	PL	ACTIVE	.	.	.	+INF	.	
41	COL41	PL **	LOWER	.	.	.	+INF	.	
42	COL42	PL	ACTIVE	1.00000	.	.	+INF	.	
43	COL43	PL **	LOWER	.	.	.	+INF	.	
44	COL44	PL	ACTIVE	.	.	.	+INF	.	
45	COL45	PL **	LOWER	.	.	.	+INF	.	

Figure 4 - Solution Report from APEX III

SECTION 3
RESULTS AND CONCLUSIONS

The study found that the File Allocation model is a useful tool for optimally allocating files in a multiple computer network. The assumptions on which the model is based were found to be reasonable. The use of the model depends on a data generating routine to generate and format the large amount of data required by the model and an integer linear programming system to solve the integer linear Programming Problem.

The two integer linear programming systems ARRIBA and APEX III are both applicable for use with the model. ARRIBA, the all in-core system, cannot be used to solve problems larger than those shown in Table 2. APEX III, the out-of-core system, can solve the problems of Table 2 and larger problems because of its capability to use extended core capacities. Cost and Central Processing Unit (CPU) time are not critical factors when ARRIBA is the solution system for the model. Table 3 shows that the solution of the largest problem that can be solved by the ARRIBA system costs less than $6.00 and requires less than 40 seconds of CPU time.

Cost, however, does become a critical factor when APEX III is the solution system for the model. APEX III and ARRIBA are roughly equivalent in direct computer cost* for problems which can be solved by either system. For problems larger than those that can be handled by ARRIBA, the direct computer costs for APEX III increase quite rapidly as problems get larger, as shown in Table 4. For example, when eight files are allocated to five computers, the cost is $36.00. When one more file is added, that is, nine files allocated to five computers, the direct computer cost goes from $36.00 to $56.00, an increase of 56 percent.

---

* Direct Computer Cost refers to the cost to solve a problem by the system. It does not include the cost to acquire and maintain the system and other costs associated with the system.

TABLE 2 - PROBLEM SIZE LIMITATIONS FOR ARRIBA

COMPUTERS	FILES 1	2	3	4	5	6	7	8
3	X	X	X	X	X	X	X	
4	X	X	X	X	X	X		
5	X	X	X	X	X			
6	X	X	X	X				
7	X	X	X					
8	X	X	X					
9	X	X	X					
10	X	X						

TABLE 3 - COST AND TIME TO SOLVE PROBLEMS BY ARRIBA

NO. OF COMPUTERS	NO. OF FILES	TIMES (Sec)	COST ($)
3	5	18	3.29
3	7	34	5.11
4	6	32	4.75
5	5	32	4.68
6	4	25	3.25

TABLE 4 - COST AND TIME TO SOLVE PROBLEMS BY APEX III

NO. OF COMPUTERS	NO. OF FILES	TIMES (Sec)	COST ($)
3	5	1	3.00
3	7	2	4.00
3	10	10	23.00
4	6	2	5.00
5	5	2	4.00
5	6	3	6.00
5	8	14	36.00
5	9	22	56.00
5	10	27	76.00

There are several associated costs in addition to the direct computer cost for the APEX III system. A user or potential user must decide whether to acquire the system for installation on his local CDC Computer or whether to use the system by remote job entry through the CDC Cybernet services.

Acquiring the CDC APEX III system will incur the following costs:

Initial Fee	\$2310.00
Monthly License	410.00
Monthly Maintenance	280.00

The monthly license fee can be eliminated by making a one-time payment of \$19,530.00 which is known as the paid-up license fee. If the initial fee of \$2310.00 is considered as a sunk cost, the monthly cost for the availability of the APEX III system is \$690.00.

Using APEX III through the Cybernet service requires contractual arrangements with CDC. The user must also have at his site a terminal through which jobs can be submitted to the APEX III system's input queue and through which the APEX III output can be received. A cost may be incurred for use of the terminal; that charge is imposed by the user's organization and not by CDC. Terminal charges at DTNSRDC are \$50.00 per hour. Additional cost for telephone connect time is also incurred by use of the Cybernet Services. Presently that cost is \$12.00 per hour.

APPENDIX A
MATHEMATICAL DEVELOPMENT OF THE FILE ALLOCATION MODEL

Dr. Chu's file allocation model is an integer linear programming model in binary variables. Mathematically, a linear programming problem is one requiring the maximization or minimization of a linear expression, referred to as the objective function, which at the same time satisfies a set of linear constraining relations (equalities or inequalities). A linear programming problem can be expressed in the form:

$$\text{Maximize (Minimize)} \quad \sum_{i=1}^{N} c_i X_i \tag{1}$$

$$\text{Subject to the Constraints} \quad \sum_{i=1}^{N} a_{ij} X_i \leq b_j \quad \text{for } j = 1,2,\ldots M \tag{2}$$

$$\text{Each} \quad X_i \geq 0 \tag{3}$$

When all variables are required to be 0 or some positive integer, the problem is referred to as an integer linear programming problem. An integer linear programming problem in binary variables is a special case in which all variables are required to be either 0 or 1. If the objective function or one or more of the constraints is not linear, the problem is called a non-linear programming problem.

In integer linear programming in binary variables, the variables indicate whether a particular action is to be taken. If an action is to be taken, the variable takes on the value one; otherwise it takes on the value zero. In Dr. Chu's formulation of the file allocation problem, the binary variable "X" indicates whether the jth file is stored in the ith computer as follows:

$$X_{ij} = \begin{cases} 1 & \text{jth file stored on ith computer} \\ 0 & \text{jth file not stored on ith computer} \end{cases} \tag{4}$$

$$i = 1, N$$

$$j = 1, M$$

where N is the total number of computers in the multicomputer system, and M is the total number of distinct files in the multicomputer system.

To insure redundant copies of each file in the system we have the constraint

$$\sum_{i=1}^{N} X_{ij} = r_j \qquad \text{for } 1 \leq j \leq M \tag{5}$$

If redundant copies of files are not allowed in the network, Equation (5) becomes

$$\sum_{i=1}^{N} X_{ij} = 1 \qquad \text{for all } j \tag{6}$$

To insure that the storage capacity of each computer is not exceeded we have the constraint

$$\sum_{j=1}^{N} X_{ij} L_j \leq b_i \quad \text{for } 1 \leq i \leq N \tag{7}$$

where L_j is the length of the jth file

b_i is the available storage at the ith computer

Each file has a maximum allowable retrieval time at each computer, which is denoted by T_{ij}. The expected time for the ith computer to retrieve the jth file from the kth computer (from initiation of request to the start of reception) is denoted by a_{ijk}. To insure that a_{ijk} is less than or equal to T_{ij}, we have the constraint

$$(1 - X_{ij}) X_{kj} a_{ijk} \leq T_{ij} \qquad \text{for } i \neq k \text{ and } 1 \leq j \leq M \tag{8}$$

When redundant files are not allowed, X_{ij} or X_{kj} equals zero and therefore the product $X_{ij}X_{kj}$ equals zero. Equation (8) under this condition reduces to

$$X_{kj} a_{ijk} \leq T_{ij} \qquad \text{for } i \neq k \text{ and } 1 \leq j \leq M \tag{9}$$

a_{ijk} is equal to the sum of the expected queuing delay (waiting time) at the ith computer for a channel to the kth computer (w_{ik}), the expected queuing delay at the kth computer for the channel to the ith computer (w_{ki}), and the expected computer access time to the jth file (t_{kj}). The total waiting time is therefore equal to:

$$W_{ik} = w_{ik} + w_{ki} + t_{kj} \tag{10}$$

Dr. Chu found that in most cases, the quantity t_{kj} is much smaller than $W_{ik} = (w_{ik} + w_{ki})$ and can be neglected. An implied assumption is that all files are online. If we disregard t_{kj}, the total waiting time is:

$$W_{ik} = w_{ik} + w_{ki} \tag{11}$$

Therefore, a_{ijk} is approximately equal to W_{ik}.

The transmission of files between two computers is assumed to be full-duplex, that is, each computer is assumed to be able to send and receive transmissions simultaneously. Figure A-1, reproduced from Dr. Chu's paper, illustrates the full-duplex operation of the request for files and the communication of files.

The communication system allows for transmitting request messages (messages requesting files) or reply messages (files) in a priority sequence. In most cases request messages are much shorter than reply messages and are assigned a higher priority. Messages of the same priority are served in the order of arrival. Transmission of a low priority message is interrupted by a high priority message and is resumed after the transmission of the high priority message. This preemptive-resume priority servicing facilitates optimization, since the queuing delay will be minimum if the shortest messages are transmitted first. Since request messages are usually very short in comparison to reply messages, the delay in transmission due to request messages can be neglected. Under these conditions the queuing system at each computer can be viewed as a single server queue with constant service time.

The file accessing process was modeled as a Poisson process. λ_{ik} represents the arrival rate of requests from the ith computer to the

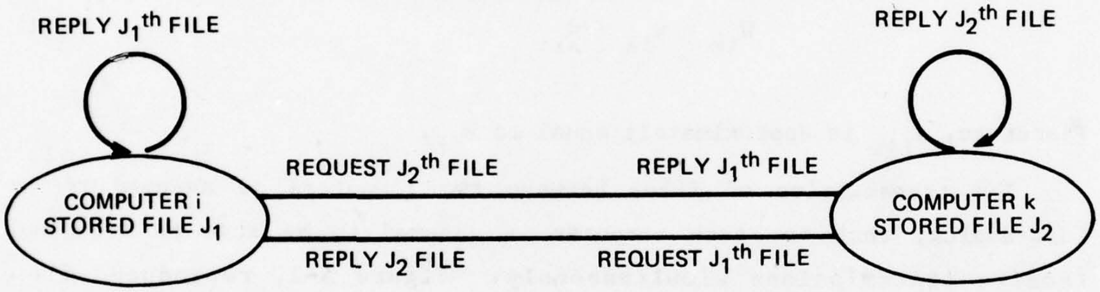


Figure A-1 - Transmission Path between Each Pair of Computers kth computer. The entire length of a file j of length L_j is not always needed each time a request is made for file j. The average length of a transaction of file j is represented by ℓ_j, where ℓ_j is less than or equal to L_j. The average time required to transmit a file j from the kth computer to the ith computer is represented by $1/\mu_{ik}$ (service time). The service time, $1/\mu_{ik}$, is dependent on both ℓ_j and λ_{ik}. Both λ_{ik} and $1/\mu_{ik}$ depend on the unknown allocation of the files to the computers. Both λ_{ik} and $1/\mu_{ik}$ will be expressed in terms of the unknown allocations (X_{ij}).

$$\lambda_{ik} = \sum_{j=1}^{M} u_{ij}(1-X_{ij})X_{kj} \tag{12}$$

where u_{ij} = is the request rate for all or part of the jth file at the ith computer per unit time.

The average time required to transmit a reply message from the kth to the ith computer via a line with transmission rate R is the time required for the kth computer to reply to all the messages requested from the ith computer to the kth computer divided by the total number of requests initiated from the ith computer to the kth computer. This is represented mathematically as

$$\frac{1}{\mu_{ik}} = \frac{1}{\lambda_{ik}} \sum_{j=1}^{M} \frac{1}{\mu_j} u_{ij}(1-X_{ij})X_{kj} \tag{13}$$

where $1/\mu_j = \ell_j/R$ is the time required to transmit each transaction of the jth file. Since ℓ_j and R are constants, μ_j and μ_{ik} are also constants.

The traffic intensity from the kth computer to the ith computer, ρ_{ik}, measures the degree of congestion of the line that provides the transmission path between the kth and ith computers, or the fraction of time that the line is busy. It is defined as the arrival rate divided by the service rate or mathematically as

$$\rho = \frac{\lambda_{ik}}{\mu_{ik}} = \sum_{j=1}^{M} \frac{1}{\mu_j} u_{ij}(1-X_{ij})X_{kj} \tag{14}$$

which is derived from Equation (13) by multiplying both sides by λ_{ik}.

The average waiting time, W_q, from the initiation of a request to the beginning of service must now be computed. This time is required because it is the time a_{ijk} which must be less than T_{ij} as required by Equation (7). Before a formula for W_q is given, additional background information must be developed.

In queuing system terminology, the queuing process of the file allocation model is denoted by M/D/1. M stands for Markovian and indicates Poisson file arrival rates for requests from one computer to another, D stands for Deterministic (constant) service time, and the 1 indicates one server.

The constant service rate can be represented by the Erlang distribution. The Erlang distribution is represented by the formula

$$f(x) = \frac{(\mu k)^k}{(k-1)!} X^{k-1} e^{-k\mu x} \quad 0 < X < \infty \tag{15}$$

where k is any arbitrary positive integer and μ is any arbitrary positive constant.

This formula represents a family of distributions. When the parameter k equals 1, $f(x) = \mu e^{-\mu x}$ which is the familiar exponential distribution. When k becomes infinite, f(x) equals the constant $1/\mu$ and it is for this reason that the Erlang distribution can be used to represent a constant service distribution. The queuing process of the file allocation model can then be represented as $M/E_k/1$, where E_k is the Erlang distribution of type k when k becomes infinite.

W_q for the $M/E_k/1$ queuing system is given by the formula

$$W_q = \frac{k+1}{2k} \frac{\lambda_{ik}}{\mu_{ik}(\mu_{ik} - \lambda_{ik})} \text{ for } i \neq k \tag{16}$$

When E_k represents a constant service time, k becomes infinite and Equation (16) reduces to Equation (17).

$$W_q = \frac{\lambda_{ik}}{2\mu_{ik}(\mu_{ik} - \lambda_{ik})} \text{ for } i \neq k \tag{17}$$

An equivalent expression for Equation (17) is

$$W_q = \frac{1}{\mu_{ik}} \frac{\rho_{ik}}{2(1-\rho_{ik})} \quad \text{for } i \neq k \tag{18}$$

Equations (17) and (18) can be shown to be equivalent by substituting λ/μ for ρ_{ik} in Equation (18) and simplifying the expression.

Substituting W_q for a_{ijk} in Equation (8) gives

$$(1-X_{ij})X_{kj} \frac{1}{\mu_{ik}} \frac{\lambda_{ik}}{2(\mu_{ik}-\lambda_{ik})} \leq T_{ij} \tag{19}$$

which can be simplified to the form

$$(1-X_{ij})X_{kj}\lambda_{ik} - 2\mu_{ik}(\mu_{ik}-\lambda_{ik})T_{ij} \leq 0 \tag{20}$$

When $\mu_{ik} = \mu$ and only one copy of each file is stored in the network, Equation (20) reduces to

$$\sum_{\substack{\ell=1 \\ j \neq \ell}}^{M} u_{i\ell}X_{k\ell}X_{kj} + 2T_{ij}\mu \sum_{\ell=1}^{M} u_{i\ell}X_{k\ell} + u_{ij}X_{kj} - 2\mu^2 T_{ij} \leq 0 \tag{21}$$

The total operating cost, which in linear programming is termed the objective function, must also be expressed in terms of the allocations (X_{ij}). This cost is computed on the basis of the storage and transmission costs which consist of the following components:

- C_{ij} - the storage cost for file j at the ith computer.
- C'_{ik} - the transmission cost from the ith computer to the kth computer.
- u_{ij} - the average hourly request rate for all or part of the jth file at the ith computer.
- ℓ_j - the average length of each transaction of the jth file.
- L_j - the length of the jth file in characters.
- r_j - the number of copies of the jth file stored in the system.
- P_{ij} - the frequency of modification of the jth file at the ith computer after each transaction.

The overall operating cost for the multicomputer system of N computers and M files is given by the expression:

$$C = \sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij}L_jX_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N}\frac{1}{r_j} C'_{ik}\ell_j u_{ij}X_{kj}(1-X_{ij})$$

$$+ \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N} C'_{ik}\ell_j u_{ij}X_{kj}P_{ik} \tag{22}$$

where the first term represents the storage cost and the other two terms represent the transmission costs. Equation (22) can be simplified to the form

$$C = \sum_{i=j}^{N}\sum_{j=1}^{M} D_{ij}X_{ij} - \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N} E_{ijk}\, X_{kj}X_{ij} \tag{23}$$

where

$$D_{ij} = C_{ij}L_j + \sum_{k=1}^{N}\left(\frac{1}{r_j} + P_{kj}\right)C'_{kj}\ell_j X_{kj}$$

and

$$E_{ijk} = \frac{1}{r_j}\, C_{ik}\ell_j u_{ij}$$

When only one copy of each file is stored in the network, Equation (20) reduces to

$$C = \sum_{i=1}^{N}\sum_{j=1}^{M} D_{ij}X_{ij} \tag{24}$$

Where

$$D_{ij} = C_{ij}L_j + \sum_{k=1}^{N} C_{ki}\ell_j u_{kj}P_{kj}$$

The optimal file allocation can now be computed when the cost function, Equation (22), is minimized subject to the constraints given in Equations (4), in (5) or (6), in (7), and in (20) or (21).

Equations (20),(21), and(22) are non-linear. The binary nature of the variables permitted tranformations to be made that effectively reduced the zero-one non-linear formulation to an equivalent linear zero-one formulation. The technique employed will be illustrated with the quadratic function, $f(X_1, X_2, X_3,...X_n)$, in which the variables must be 0 or 1. Obviously, the non-linear term X_j^2 can be replaced by X_j without affecting the value of the function. A new variable X_{jk} is needed to replace a product term such as X_jX_k so that its value corresponds to the values of X_j and X_k as follows:

X_j	X_k	X_{jk}
0	0	0
0	1	0
1	0	0
1	1	1

The relation(s) which will insure the desired correspondence must be developed. Note that X_{jk} is the product of binary variables and therefore it must also be a binary variable. X_{jk} can be constrained to be a binary variable by requiring it to be an integer variable and imposing the constraint that its value be less than or equal to one.

Given that X_{jk} is a positive integer or zero, it must be shown that it can be constrained to take on the same value as the product it represents. If X_j and/or X_k is zero, then X_{jk} must be zero. This condiition is satisfied by the following two constraints:

$$X_{jk} \leq X_j \tag{25}$$

$$X_{jk} \leq X_k \tag{26}$$

These two constraints can be combined by addition as follows:

$$2X_{jk} \leq X_j + X_k \tag{27}$$

When X_j and X_k are both equal to one, the new variable, X_{jk}, must also equal one. Since X_{jk} equals one depends on both X_j and X_k, a a linear relation involving both X_j and X_k and requiring X_{jk} to be greater than or equal to one is sought. The number "1" can be expressed as $(X_j+X_k) - 1$ and therefore the desired constraint can be expressed as:

$$X_{jk} \geq X_j + X_k - 1 \tag{28}$$

Equations (27) and (28) and the constraint that X_{jk} is an integer gives the desired correspondence to transform the non-linear formulation of the model to an equivalent linear formulation. Higher order non-linear equations in zero-one variables can be reduced to equivalent linear equations in a similar manner.

In the application of this linerization technique, each non-linear term is replaced by the new linear variable and the corresponding two new constraints are introduced into the problem. For instance, when the non-linear term $X_{ij}X_{kj}$ is replaced by the new linear variable X_{ijkj}, the following two constraints are added to the problem:

$$X_{ij} + X_{kj} - X_{ijkj} \leq 1 \tag{29}$$

$$-X_{ij} - X_{kj} + 2X_{ijkj} \leq 0 \tag{30}$$

The file allocation problem can now be solved by minimizing the objective function, Equations (23) or (24), subject to the constraints given in Equations (4), in (5) or (6), in (7), in (20) or (21), in (29), and in (30) when the non-linear terms are replaced by linear terms as described.

APPENDIX B
DATA REQUIREMENTS FOR THE COMPUTER SYSTEMS
ARRIBA, APEX III, AND DATASUP

B.1 GENERAL

ARRIBA and APEX III are two computer systems capable of solving integer linear programming problems. Both systems were considered for use with the file allocation model. Either system can be used for small problems (problems in which the sum of the number of computers and files is ten or less) but only APEX III can be used to solve larger problems.

Documentation on both systems is available from the Control Data Corporation. The documentation of ARRIBA includes the computer coding.

To facilitate the use of the file allocation model with either solution system, a computer program DATASUP was written to generate and format the input data needed to solve file allocation problems by the model with whichever of the two systems is selected for solution by the user.

This Appendix describes first, the input data and their formats for solving integer linear programming problems by the solution systems, ARRIBA and APEX III, and second, the input data needed by DATASUP to generate and format the data for either of the two solution systems.

B.2 ARRIBA INTEGER PROGRAMMING SYSTEM

Figure B-1 shows the input data setup for any problem to be solved by ARRIBA.

1. CONTROLS Card - This card contains the data which set the limits on the parameters governing how long the model will run in attempting to reach an optimal solution and what output will be printed.

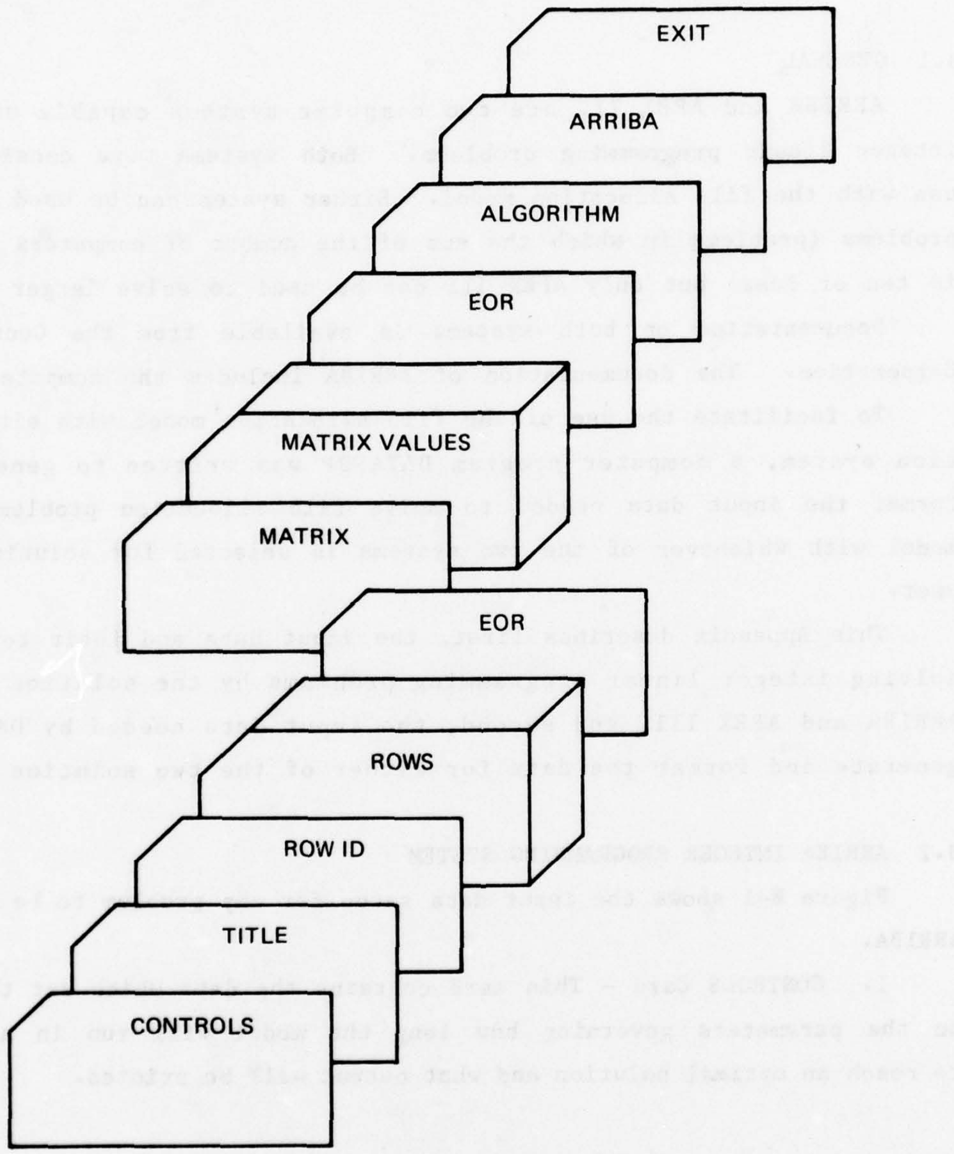


Figure B-1 - Sequence for Data within Data Deck for ARRIBA

CARD COLUMNS	PARAMETER	DESCRIPTION
1 - 8	CONTROLS	The word "CONTROLS"
15	0,1, or blank	1 indicates that the input data are to be printed. 0 or blanks indicate that the input data are not to be printed
34 - 35		Iterations at which the objective function values are to be printed.
56 - 60		Iteration at which the values of all variables will be printed.
73 - 80		Maximum number of iterations allowed by the model in trying to reach an optimal solutions.

If the CONTROLS card contains the word CONTROLS only, the parameters are set to the default values of 0, 10, 1000, and 999, respectively.

2. TITLE Card - This card gives the user-chosen problem title. The first six characters of the problem title appear on all output.

CARD COLUMNS	PARAMETER	DESCRIPTION
1 - 6	TITLE,	The word "TITLE" followed by a comma.
7 - 80		Any title information selected by the user.

3. ROW ID Card - The third card, a header card which signals the beginning of the rows section, contains ROW ID in columns 1 - 6. It is followed by rows identification cards in the following format:

CARD COLUMNS	PARAMETER	DESCRIPTION
12	+, -, or blank	+ indicates that the constraint is of the form $\leq$ - indicates that the constraint is of the form $\geq$. A blank indicates that the constraint is of the form =.
13 - 18	Row Identification	Row name in six alphanumeric characters.

The first row name is always the name of the objective function. The objective function row must contain a "+" in column 12 to indicate minimization or a "-" to indicate maximization.

4. EOR Card - This card terminates the definition of rows. The EOR card is not to be confused with the End-of-Record Control card used in the CDC Scope Operating system. The EOR card contains "EOR" in columns 1 - 3.

5. MATRIX Card - This card signals that the body of the matrix follows. It contains the word MATRIX in columns 1 - 6. The MATRIX cards are followed by matrix values cards in the following format:

CARD COLUMNS	PARAMETER	DESCRIPTION
7 - 12	Variable Name	1- to 6-character column variable name.
13 - 18	Row Identification	Row identification must correspond to the row identification given in the ROW ID section.
19 - 30	Matrix Values	Non-zero matrix values read in with F12.6 specification.

6. EOR Card - This card is identical to the EOR Card described in subparagraph 4. It signals the end of the matrix values.

7. ALGORITHM Card - Three algorithms are available as solution techniques for problems to be solved by ARRIBA. The Gomory cutting plane technique has been determined to be the best choice for the file allocation problem. This algorithm is identified by placing "IPSC" in card columns 1 - 4.

8. ARRIBA Card - This card contains the word "ARRIBA" in card columns 1 - 6 and signals the system to begin computation.

9. EXIT Card - This card contains the word "EXIT" in card columns 1 - 4 and signals the end of the job.

B.3 APEX III MATHEMATICAL PROGRAMMING SYSTEM FOR USERS OF CDC COMPUTERS

Figure B-2 shows the input data setup for the file allocation model to be solved by the APEX III system.

1. NAME Card - This is the first card of the data deck. It is the problem header card which identifies the problem by a title.

CARD COLUMNS	PARAMETER	DESCRIPTION
1 - 4	NAME (Title)	The word "NAME"
15 - 24	Problem Name	This is the user-chosen problem name using 1 to 10 alphanumeric characters.

2. ROWS SECTION HEADER Card - This is the second header card of the data deck. It signals the beginning of the rows section.

CARD COLUMNS	PARAMETER	DESCRIPTION
1 - 4	ROWS	The word "ROWS"

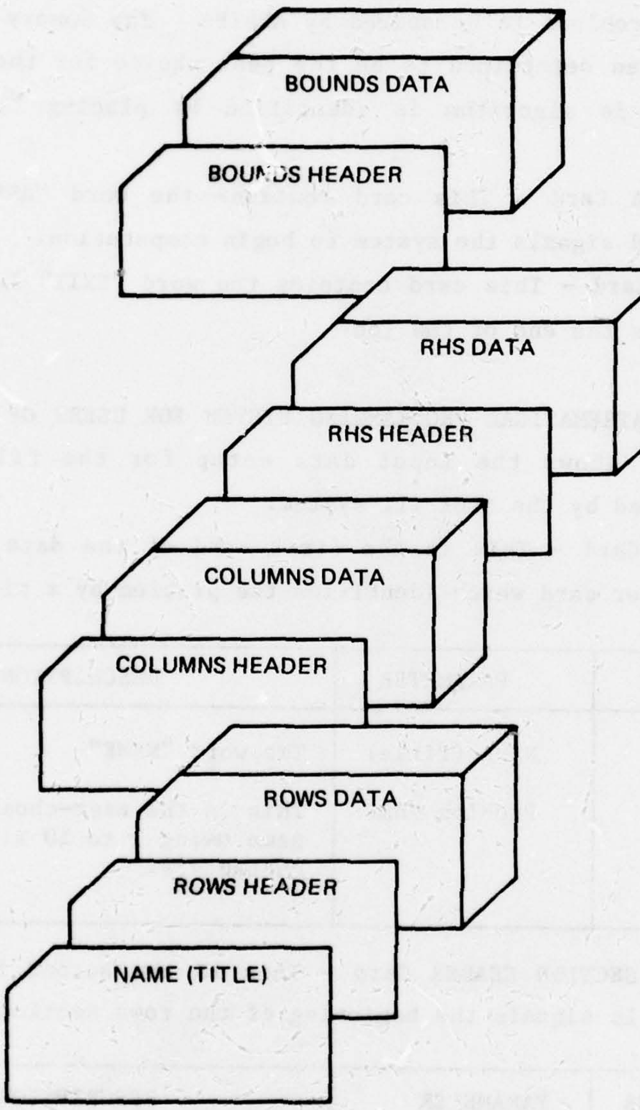


Figure B-2 - Sequence for Data within Data Deck for APEX III

3. ROWS SECTION DATA Cards - These cards immediately follow the ROWS header card. They define the constraint relations of the linear programming problem and assign names to the rows.

CARD COLUMNS	PARAMETER	DESCRIPTION
2 - 3	E, G, or L	E = equality G = Greater than or equal to L = less than or equal to
5 - 14	Row Name	This is the user-chosen row name using 1 to 10 alphanumeric characters.

4. COLUMNS SECTION HEADER Card - This header card signals the beginning of the Columns section. It contains the word "COLUMN" in card columns 1 - 7.

5. COLUMNS SECTION DATA Cards - These cards immediately follow the column header card and define the coefficients of the column variables. (These data correspond to the MATRIX values of the ARRIBA system).

CARD COLUMNS	PARAMETER	DESCRIPTION
5 - 14	Variable Name	1- to 10-character column variable name.
15 - 24	Row Name	1- to 10-character row name
25 - 36	Matrix Coefficient	Non-zero matrix coefficients.

6. RIGHT-HAND-SIDE (RHS) HEADER Card - This card is the fourth required section header and must follow the columns section data. This card contains RHS in columns 1 - 3.

7. RHS SECTION DATA Cards - These cards immediately follow the RHS header card. They contain the values on the right side of the relation signs. The format for these cards is the same as the format for the columns section data. Columns 5 - 14 contain the 1- 10-character RHS column name.

8. BOUNDS SECTION HEADER Card - This card is the fifth required section header and must follow the RHS section data. It contains "BOUNDS" in card columns 1 - 6.

9. BOUNDS SECTION DATA Cards - These cards immediately follow the Bounds header card.

CARD COLUMNS	PARAMETER	DESCRIPTION
2 - 3	BV	BV indicates that the bounded variables take on values of 0 or 1 only.
5 - 14	BOUNDS	User chosen name for the bounds set in one to ten characters.
15 - 24	Variable Name	This is the same variable name used in the columns data section

B.4 DATASUP - DATA GENERATING AND FORMATTING ROUTINE

Figure B-3 shows the input data structure for DATASUP. The variables N and M are used throughout this subsection and will always have the meaning defined in subparagraph 1.

1. The first data card contains the values of the three variables N, M, and μ, where N is the number of computers, M is the number of files, and μ is the rate of service (service rate is the number of files that can be transmitted per unit of time; $1/\mu$ is the average service time). The formats for these data are I4, I4, and F4.0, respectively.

2. The second card or group of cards contains values for the one-demensional array SLC. The elements of this array are the average lengths of transactions of each of the M files. The format for these M elements is 10F8.0.

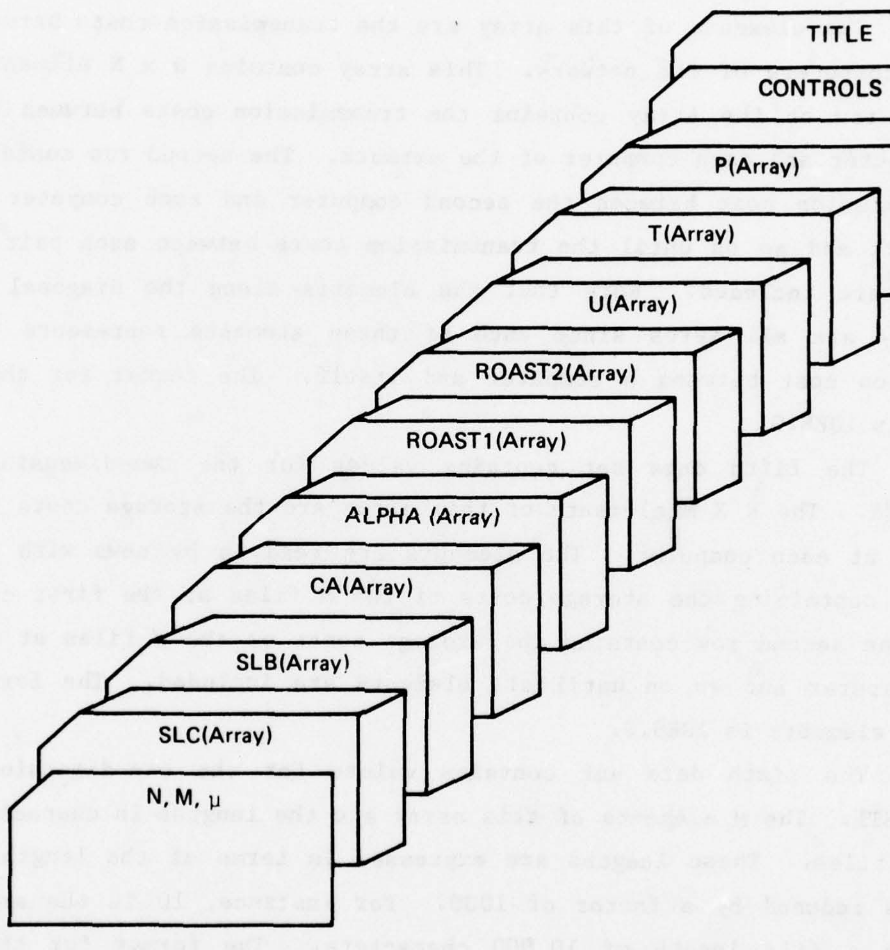


Figure B-3 - Sequence for Data within Data Deck for DATASUP

3. The third data set contains values for the one-dimensional array SLB. The elements of this array are the lengths in characters of the M files. The format for these M elements is 10E8.0.

4. The fourth data set contains values for the two-dimensional array CA. The elements of this array are the transmission costs between each two computers of the network. This array contains N x N elements. The first row of the Array contains the transmission costs between the first computer and each computer of the network. The second row contains the transmission cost between the second computer and each computer of the network and so on until the transmission costs between each pair of computers are included. Note that the elements along the diagonal of this array are all zeros since each of these elements represents the transmission cost between a computer and itself. The format for these elements is 10E8.0.

5. The fifth data set contains values for the two-dimensional array ALPHA. The N X M elements of this array are the storage costs for each file at each computer. The elements are read in by rows with the first row containing the storage costs of the M files at the first computer. The second row contains the storage costs of the M files at the second computer and so on until all elements are included. The format for these elements is 10E8.0.

6. The sixth data set contains values for the one-dimensional array ROAST1. The M elements of this array are the lengths in characters of the M files. These lengths are expressed in terms of the length in characters reduced by a factor of 1000. For instance, 10 in the array represents a file length of 10,000 characters. The format for these elements is 10F8.0.

7. The seventh data set contains values for the one-dimensional array ROAST2. The N elements of the array are the lengths in characters of the storage availabilities of the N computers of the network. These lengths are also expressed as the actual lengths reduced by a factor of 1000. The format for these elements is 10F8.0.

8. The eighth data set contains values for the two-dimensional integer array U which contains N X M elements. The elements of the array are the hourly request rates for all or part of the M files at each of the

N computers. The elements of this array are read in by rows with the first row containing the request rates of the M files at the first computer. The second row contains the request rates of the M files at the second computer, and so on until the request rates of all files at all computers are included. The format for these elements is 10F8.0.

9. The ninth data set contains values for the two-dimensional array T. The N X M elements of this array are the maximum average retrieval time in seconds for each file at each computer. This array is read in by rows with the first row containing the maximum retrieval times for the M files at the first computer. The second row contains the maximum retrieval times for the M files at the second computer and so on until all elements are included. The format for these elements is 10F8.0.

10. The tenth data set contains values for the two-dimensional array P. The N X M elements of this array are the frequency of modification of the jth file at the ith computer after each transaction. This array is read in by rows with the first row containing the frequencies of modifying files at the first computer. The second row contains the frequencies of modifying files at the second computer and so on until the frequencies of modifying all files at each computer are included. The format for these elements is 20F4.2.

11. The eleventh data set contains only one card. This one card supplies input to the CONTROLS card of the ARRIBA system. If the default values (See Section B.2) are to be used for these parameters, this card must be left blank.

CARD COLUMNS	PARAMETER	DESCRIPTION
1 - 8	CONTROLS	The word "CONTROLS"
15	0,1, or blank	1 indicates that the input data are to be printed. 0 or blank indicates that the input data will not be printed.
34 - 35		Iterations at which the objective function values will be printed.
56 - 60		Iterations at which the values of all variables will be printed.
73 - 80		Maximum number of iterations allowed by the model in trying to reach an optimal solution.

12. The 12th data set also contains only one card. This card contains any title information the user chooses for his problem. All 80 columns of the card may be used.

APPENDIX C
PROBLEM ORGANIZATION

Using the equation numbers given in Appendix A, the file allocation model minimizes Equation (24) subject to Equations (4), (6), (7), (21), (29), and (30). The organization of the problem shown below is the organization from which the row and column numbers shown in Figures 3 and 4 are obtained.

1. Objective function row

$$\sum_{i=1}^{N} \sum_{j=1}^{M} D_{ij}X_{ij} \tag{24}$$

2.

$$\sum_{\ell=1}^{M} u_{i\ell}X_{k\ell}X_{kj} + 2T_{ij}\mu \sum_{\ell=1}^{M} u_{i\ell}X_{k\ell} + u_{ij}X_{kj} - 2\mu^2 T_{ij} \leq 0 \tag{21}$$

where $j \neq \ell$ and $i \neq k$ for $i = 1, N$; $j = 1, M$; and $k = 1, N$

This constraint set consists of (N-1)(M)(N) equations.

3.

$$- X_{ij} - X_{kj} + 2X_{ijkj} \leq 0 \tag{30}$$

This constraint set consists of N times the combination of M rows taking two at a time, (N[C(M,2)]), required by the non-linear terms in Equation (21).

4.

$$X_{ij} + X_{kj} - X_{ijkj} \leq 1 \tag{29}$$

This equation set also consists of N[(C(M,2)] rows, required by the non-linear terms in Equation (21).

5.

$$\sum_{i=1}^{M} X_{ij} = 1 \quad \text{for all } j \tag{6}$$

This constraint set consist of M equations. It insures that each file is allocated to one of the computers. This constraint also satisfies the requirements of Equation (4) that all X_{ij}'s be 0 or 1.

6.

$$\sum_{j=1}^{M} X_{ij}L_j \leq b_i \quad \text{for } 1 \leq i \leq N \tag{7}$$

This constraint set consists of N equations. It insures that the storage capactities of the computers are not exceeded.

The numbers of rows (including the objective function row) and columns (including the RHS column) of any problem can be computed from the following formulas:

$$\text{Number of Columns} = (N \times M) + N[C(M,2)] + 1$$

$$\text{Number of Rows} = (N-1)(N \times M) + 2N[C(M,2)] + (M+N) + 1$$

PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

APPENDIX D

PROGRAM DATASUP

```
      PROGRAM DATASUP(INPUT,TAPE5=INPUT,TAPE6,TAPE7,TAPE8,OUTPUT,TAPE9=O
     1UTPUT)
      REAL MU
C     DIMENSION AND INTEGER STATEMENTS
      INTEGER XS,XB,XD,RB
      INTEGER ML1,ML
      INTEGER G(15,15)
      DIMENSION TITLE(8),RUST(4)
      DIMENSION UV(15),TV(15),ROAST1(15),ROAST2(15),P(15,15)
      DIMENSION VAR(225,16),VARN(105,15),ROAST(16,15),VART(225,105),
     1ALPHA(15,15),SLB(15),CA(15,15),SLC(15),OBJ(225),U(15,15),T(15,15)
C     FUNCTION TO ASSIGN THE SUBSCRIPTS (I,J) A NUMBER
      M(I,J,K)=(I-1)*K+J
C     DATA STATEMENTS
      DATA OB,AB,BA,AX,CX,ARB/1H+,2H+R,2H R,1HC,3HOBJ,3HRHS/
      DATA STRESS,YZ/7H      C,2H E/
      DATA ABC,EFG/1HL,1HE/
      READ(5,53)N,M,MU
   53 FORMAT(2I4,F4.0)
C     GENERATING NUMBERS (CONSTANTS) THAT WILL BE USED FREQUENTLY FOR SUBSCRIPTING
C     AND IN DO LOOPS
      NM=N*M $ NCOMB=M*(M-1)/2 $ NNCOMB=N*NCOMB
      NK=M*N*(N-1) $ LA=M+1 $ NTOT=NK+2*NNCOMB+    M+N
      READ(5,54)(SLC(I),I=1,M)
   54 FORMAT(10F8.0)
      READ(5,55)(SLB(I),I=1,M)
   55 FORMAT(10E8.0)
      READ(5,55)((CA(I,J),J=1,M),I=1,N)
      READ(5,55)((ALPHA(I,J),J=1,M),I=1,N)
      READ(5,54) (ROAST1(I),I=1,M)
      READ(5,54)(ROAST2(I),I=1,N)
      DO 71 J=1,N
      DO 57 I=1,M
   57 ROAST(I,J)=ROAST1(I)
   71 ROAST(LA,J)=ROAST2(J)
      READ(5,54)((U(I,J),J=1,M),I=1,N)
      READ(5,54)((T(I,J),J=1,M),I=1,N)
      READ(5,56) ((P(I,J),J=1,M),I=1,N)
   56 FORMAT(20F4.2)
C     GENERATING THE OBJECTIVE VALUES
      DO 8 I=1,N
      DO 8 J=1,M
      S=0.
      DO 9 K=1,N
      IF(K.EQ.I)GO TO 9
      S=CA(K,I)*SLC(J)*U(K,J)*(1+P(K,J))+S
    9 CONTINUE
      KY=(I-1)*M+J
      OBJ(KY)=(S+ALPHA(I,J)*SLB(J))*1000000
    8 CONTINUE
C     MAKING AN ARRAY THAT WILL GIVE THE SUBSCRIPTS (I,J) A NUMBER(NOTE J MUST BE
C     .LT.I.
      KNOT=0
      DO 10 I=1,M
      DO 10 J=1,M
      IF(I.GE.J)GO TO 10
      KNOT=KNOT+1
```

```
      G(I,J)=KNOT
   10 CONTINUE
C  VAR
C   MAKING AN ARRAY VAR WHICH WILL BE USED REPEATEDLY AND WHICH GIVES THE LINEAR
C   PART OF EQUATION SET 15.
      DO 15 I=1,N
      DO 15 J=1,N
      TTU=2.*T(I,J)*MU $ VAR(H(I,J,M),LA)=TTU*MU $ IF(TTU.EQ.0.)GO TO 15
      DO 16 L=1,M
   16 VAR(H(I,J,M),L)=U(I,L)*TTU
      VAR(H(I,J,M),J)=U(I,J)+VAR(H(I,J,M),J)
   15 CONTINUE
C  VART
C   MAKING AN ARRAY VART WHICH WILL BE USED REPEATEDLY AND WHICH IS THE
C   X(K,L)X(K,J) PART IN EQUATION SET 15 (NOTE WHEN J.EQ.L THEN X(K,L)X(K,J)
C   BECOMES X(K,J).
      DO 11 I=1,N
      DO 11 J=1,M
      IF(T(I,J).EQ.0.)GO TO 11
      DO  7 L=1,M
      IF(J-L)13,7,14
   13 VART(H(I,J,M),G(J,L))=U(I,J) $ GO TO 7
   14 VART(H(I,J,M),G(L,J))=U(I,J)
    7 CONTINUE
   11 CONTINUE
C  VARW
C   MAKING AN ARRAY VARW WHICH WILL BE USED REPEATEDLY AND WHICH IS THE
C   X(K,L)+X(K,J) IN THE TWO EQUATIONS X(K,L)+X(K,J).GE.2X(K,L)X(K,J) AND
C   X(K,L)+X(K,J)-1.LE.X(K,L)X(K,J) FOR J.NE.L.
      LN=0
      K=1
      DO 12 J=1,M
      DO 12 I=1,M
      IF(J.GE.I)GO TO 12 $ LN=LN+1
      HL1=H(K,I,M) $ HL=H(K,J,M)
      VARW(LN,HL1)=VARW(LN,HL)=1.
   12 CONTINUE
C   WRITING HEADING NECESSARY FOR THE INTEGER PROGRAMMING PROGRAM APEX.
      WRITE(6,60)
   60 FORMAT(* NAME*,11X,*FILE ALLOC*/* ROWS*)
      WRITE(6,61)
   61 FORMAT(* N  OBJ*)
C   WRITING OUT THE ROWS AND THE TYPE OF RELATIONSHIP
C   A  .LE. RELATIONS
      NOTE=NR+2*NNCOMB     $ NOOT=NR+2*NNCOMB+   M
      DO 42 K=1,NOTE
      WRITE(6,62)ABC,K
   62 FORMAT(1X,A1,2X,*ROW*,I4)
   42 CONTINUE
C   B  .EQ.  ELATIONS
      NOTE=NOTE+1
      DO 43 K=NOTE,NOOT
   43 WRITE(6,62)EFG,K
C   C  .LE. RELATIONS
      NOOT=NOOT+1
      DO 44 K=NOOT,NTOT
   44 WRITE(6,62)ABC,K
```

```
      C   HEADING OF COLUMNS NEEDED FOR INPUT INTO APEX (THE INTEGER PROGRAMMING
      C   PROGRAM)
            WRITE(6,63)
       63 FORMAT(*COLUMNS*)
      C   START OF DO LOOP WHICH WILL WRITE ALL OF THE LINEAR PART OF EQUATION SET 15.
            DO 18 IC=1,NM
      C   FIRST WRITE THE OBJECTIVE VALUE
            X=OBJ(IC) $ IF(X.EQ.0.)GO TO 19
            WRITE(6,64)IC,X
       64 FORMAT(4X,*COL*,I4,3X,*OBJ*,7X,F12.8)
      C   HERE TWO INDEXES ARE USED. KOS IS FOR COLUMNS AND KROS IS FOR ROWS. WHENEVER
      C   KOS=KROS ALL VALUES IN THE LINEAR PART OF EQUATION SET 15 ARE ZERO. ALSO KOS
      C   HELPS DETERMINE ROW PLACEMENT (NOTE THAT EVERY VAR VALUE IS USED N-1 TIMES
      C   AND THEREFORE MUST BE USED IN DIFFERENT ROWS. KOS TELLS YOU HOW FAR ALONG
      C   YOU ARE IN THE COLUMNS AND HENCE HELPS TELL YOU HOW FAR ALONG YOU SHOULD BE
      C   IN THE ROWS.
       19 KOS=(IC-1)/M $ IR=IM=1
       20 KROS=(IR-1)/(M*(N-1)) $ IF(IR.GE.NR+1)GO TO 21
      C   THIS IF STATEMENT LETS ALL VALUES=0 IF KOS=KROS. IT LEADS TO DIFFERENT
      C   PLACES IF KOS .LT. OR .GT. KROS. THIS IS BECAUSE YOU NEED TO KNOW IF YOU
      C   PASSED THE KOS=KROS SECTION TO KNOW WHICH ROW TO BE IN.
            IF(KOS-KROS)22,23,24
      C   A  KOS.LT.KROS
       22 X=VAR(IM,IC-KOS*M) $ IF(X.EQ.0.)GO TO 25 $ IR=IR+KOS
            IM=IM+1
            WRITE(6,65)IC,IR,X
       65 FORMAT(4X,*COL*,I4,3X,*ROW*,I4,3X,F12.5)
            IR=IR-KOS+N-1 $ GO TO 20
      C   B  KOS.EQ.KROS
       23 IR=IR+(N-1)*M $ IM=IM+M $ GO TO 20
      C   C  KOS.GT.KROS
       24 X=VAR(IM,IC-KOS*M) $ IF(X.EQ.0.)GO TO 25 $ IR=IR+KOS-1
             IM=IM+1
            WRITE(6,65)IC,IR,X
            IR=IR-KOS+N $ GO TO 20
      C   D  INCREMENTING IF X=0
       25 IR=IR+N-1 $ IM=IM+1 $ GO TO 20
      C   NOTE THAT IN A,B,C AND D WE GO BACK UNTIL THE ROW NUMBER REACHES A CERTAIN
      C   LIMIT. THEN WE GO INTO THIS NEXT SECTION WHICH WRITES THE -X(K,L)-X(K,J)
      C   PART OF 2X(K,L)X(K,J)-X(K,L)-X(K,J) .LE.0.
      C   AGAIN WE USE KOS, AND ALSO WE USE KOP WHICH IS ASSOCIATED WITH ROWS. THIS
      C   TIME ALL VALUES ARE ZERO UNLESS KOS=KOP.
       21 IF(IR.EQ.NR+(KOS+1)*NCOMB+1)GO TO 27
            KOP=(IR-NR-1)/NCOMB
       28 IF(KOS.EQ.KOP)GO TO 29 $ IR=IR+NCOMB
            IM=IM+NCOMB $ GO TO 21
       29 ICC=IC-KOS*M
            KLOCK=MOD(ICC-1,M)+1
            X=-VARW(IR-NR-KOS*NCOMB,KLOCK) $ IF(X.EQ.0.)GO TO 30
            WRITE(6,65)IC,IR,X
       30 IR=IR+1 $ IM=IM+1 $ GO TO 21
      C   THIS SECTION WRITES THE X(K,L)+X(K,J) PART OF -X(K,L)X(K,J)+X(K,L)+X(K,J)
      C   .LE.1. IF KOS = KOT THEN WE WRITE OUR VALUES.
       27 IF(IR.EQ.NR+(KOS+11)*NCOMB+1)GO TO 31
            KOT=(IR-NR-NNCOMB-1)/NCOMB
            IF(KOS.EQ.KOT)GO TO 32 $ IR=IR+NCOMB
            IM=IM+NCOMB $ GO TO 27
```

```
      32 X=VARW(IR-NR-NNCOMB-KOS*NCOMB,KLOCK) $ IF(X.EQ.0.)GO TO 33
         WRITE(6,65) IC,IR,X
      33 IR=IR+1 $ IM=IM+1 $ GO TO 27
C     CONSTRAINT REQUIRING A GIVEN FILE TO BE ON AT LEAST ONE COMPUTER
      31 KAZ=MOD(IC,M) $ IF(KAZ.EQ.0)KAZ=M
         IR=NR+2*NNCOMB+    KAZ $ X=1.
         WRITE(6,65) IC,IR,X
C     CONSTRAINT REQUIRING FILE ALLOCATION NOT TO VIOLATE STORAGE OF COMPUTERS.
         KOT=(IC-1)/M+1 $ IR=NR+2*NNCOMB+    M+KOT
         X=KOAST(KAZ,KOT)
         WRITE(6,65) IC,IR,X
C     END OF LINEAR SECTION
      18 CONTINUE
C     BEGINNING OF NONLINEAR SECTION IN A DO LOOP.
C     KOS AND KROS ARE AGAIN USED AS WITH VAR. IF KOS=KROS ALL VALUES ARE ZERO AND
C     KOS IS AGAIN USED TO DETERMINE THE APPROXIMATE ROW.
         NMP1=NM+1 $ MMCOMB=NNCOMB+NM
         DO 34 IC=NMP1,MMCOMB $ KOS=(IC-1-NM)/NCOMB $ IR=IM=1
      35 IF(IR.EQ.NR+1)GO TO 36 $ KROS=(IR-1)/((N-1)*M)
         IZ=MOD(IC-NM-1,NCOMB)+1
         IF(KOS-KROS)37,38,39
C     A  KOS.LT.KROS
      37 X=VART(IM,IZ) $ IF(X.EQ.0.)GO TO 40 $ IR=IR+KOS
         WRITE(6,65) IC,IR,X
         IR=IR-KOS+N-1 $ IM=IM+1 $ GO TO 35
C     B  KOS.EQ.KROS
      38 IR=IR+(N-1)*M $ IM=IM+M $ GO TO 35
C     C  KOS.GT.KROS
      39 X=VART(IM,IZ) $ IF(X.EQ.0.)GO TO 40 $ IR=IR+KOS-1
         WRITE(6,65) IC,IR,X
         IR=IR-KOS+N $ IM=IM+1 $ GO TO 35
C     D  INCREMENTING IF X=0.
      40 IR=IR+N-1 $ IM=IM+1 $ GO TO 35
C     WRITING THE 2X(K,L)X(K,J) PART OF 2X(K,L)X(K,J)-X(K,L)-X(K,J).LE.0.
      36 X=2. $ IR=IC+NR-NM
         WRITE(6,65) IC,IR,X
C     WRITING THE -X(K,L)X(K,J) PART OF -X(K,L)X(K,J)+X(K,L)+X(K,J).LE.1.
         X=-1. $ IR=IC+NR+NNCOMB-NM
         WRITE(6,65) IC,IR,X
C     END OF NONLINEAR SECTION
      34 CONTINUE
C     RIGHT HAND SIDE VALUES HEADING
         WRITE(6,66)
      66 FORMAT(* RHS*)
C     RIGHT HAND SIDE FOR THE CONSTRAINTS FROM EQUATION SET 15
         NRMO=NR-1 $ NMO=N-1
         DO 41 I=1,NRMO,NMO
         KI=(I-1)/(N-1)+1
         X=VAR(KI,LA) $ IF(X.EQ.0.)GO TO 41
         IT=I+N-2
         DO 1 IR=I,IT
       1 WRITE(6,68) IR,X
      41 CONTINUE
      68 FORMAT(4X,*RHS*,7X,*ROW*,I4,3X,F12.5)
C     WRITING 1. FOR THE RHS FOR -X(K,L)X(K,J)+X(K,L)+X(K,J).LE.1
         X=1. $ NOTT=NR+NNCOMB+1 $ NOTTT=NR+2*NNCOMB+    M
         DO 2 IR=NOTT,NOTTT
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
      2 WRITE(6,68)IR,X
C     WRITING THE TOTAL STORAGE AVAILABLE
        NOST=NOTTT+1 $ NOSS=NOTTT+N
        DO 4 IR=NOST,NOSS
        X=ROAST(LA,IR-NOTTT)
      4 WRITE(6,68)IR,X
C     WRITING OUT WHICH COLS ARE TO BE BINARY (0 OR 1).
        WRITE(6,50)
     50 FORMAT(*BOUNDS*)
        DO 51 I=1,NN
        WRITE(6,52)I
     52 FORMAT(* BV BOUNDS*,4X,*COL*,I4,3X,*1.*)
     51 CONTINUE
C     WRITING ENDATA TO SIGNAL APEX (THE INTEGER PROGRAMMING PROGRAM) THAT ALL
C     DATA HAS ENDED.
        WRITE(6,67)
     67 FORMAT(* ENDATA*)
        READ(5,137)LIST,IOVE,IALE,IPL
    137 FORMAT(I1,I4,I5,I3)
      3 FORMAT(*CONTRO*,8X,I1,I20,I25,I20)
        READ(5,5)TITLE
      5 FORMAT(8A10)
        WRITE(7,3)LIST,IOVE,IALE,IPL
        WRITE(7,5)TITLE
        WRITE(7,6)
      6 FORMAT(*ROW ID*)
        WRITE(7,17)
     17 FORMAT(11X,*+OBJ*)
        REWIND 6
        READ(6,136)
    136 FORMAT(//)
        DO 26 I=1,NTOT
        READ(6,45)AZ,XS
     45 FORMAT(A2,5X,I4)
        ZS=XS+.5
        J=ALOG10(ZS)+1
        IF(AZ.EQ.YZ)GO TO (141,142,143,144)J
        GO TO (46,47,48,49)J
    141 WRITE(7,77)BA,XS $ GO TO 26
    142 WRITE(7,78)BA,XS $ GO TO 26
    143 WRITE(7,79)BA,XS $ GO TO 26
    144 WRITE(7,80)BA,XS $ GO TO 26
     46 WRITE(7,77)AB,XS $ GO TO 26
     47 WRITE(7,78)AB,XS $ GO TO 26
     48 WRITE(7,79)AB,XS $ GO TO 26
     49 WRITE(7,80)AB,XS $ GO TO 26
     26 CONTINUE
     77 FORMAT(11X,A2,I1)
     78 FORMAT(11X,A2,I2)
     79 FORMAT(11X,A2,I3)
     80 FORMAT(11X,A2,I4)
        READ(6,139)
    139 FORMAT(5X)
     70 READ(6,58)XA,XB,XC,XC,XE
     58 FORMAT( 4X,A1,2X,I4,3X,A3,I4,3X,F12.5)
        ZB=XB+.5
        ZD=XD+.5
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
      IF(XA.NE.AX)GO TO 59
      IF(XC.EQ.CX)GO TO 69
      J=ALOG10(ZB)+1 $ K=ALOG10(ZD)+1 $ JK=J+(K-1)*4
      GO TO (81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96)JK
   81 WRITE(8, 97) XB,BA,XD,XE $ GO TO 70
   82 WRITE(8, 98) XB,BA,XD,XE $ GC TO 70
   83 WRITE(8, 99) XB,BA,XD,XE $ GO TO 70
   84 WRITE(8,100) XB,BA,XD,XE $ GO TO 70
   85 WRITE(8,101) XB,BA,XD,XE $ GO TO 70
   86 WRITE(8,102) XB,BA,XD,XE $ GO TO 70
   87 WRITE(8,103) XB,BA,XD,XE $ GO TO 70
   88 WRITE(8,104) XB,BA,XD,XE $ GO TO 70
   89 WRITE(8,105) XB,BA,XD,XE $ GC TO 70
   90 WRITE(8,106) XB,BA,XD,XE $ GO TO 70
   91 WRITE(8,107) XB,BA,XD,XE $ GO TO 70
   92 WRITE(8,108) XB,BA,XD,XE $ GO TO 70
   93 WRITE(8,109) XB,BA,XD,XE $ GO TO 70
   94 WRITE(8,110) XB,BA,XD,XE $ GO TO 70
   95 WRITE(8,111) XB,BA,XD,XE $ GO TO 70
   96 WRITE(8,112) XB,BA,XD,XE $ GO TO 70
   97 FORMAT(6X,*C*,I1,3X,A2,I1,4X,F12.2)
   98 FORMAT(6X,*C*,I2,2X,A2,I1,4X,F12.2)
   99 FORMAT(6X,*C*,I3,1X,A2,I1,4X,F12.2)
  100 FORMAT(6X,*C*,I4,   A2,I1,4X,F12.2)
  101 FORMAT(6X,*C*,I1,3X,A2,I2,3X,F12.2)
  102 FORMAT(6X,*C*,I2,2X,A2,I2,3X,F12.2)
  103 FORMAT(6X,*C*,I3,1X,A2,I2,3X,F12.2)
  104 FORMAT(6X,*C*,I4,   A2,I2,3X,F12.2)
  105 FORMAT(6X,*C*,I1,3X,A2,I3,2X,F12.2)
  106 FORMAT(6X,*C*,I2,2X,A2,I3,2X,F12.2)
  107 FORMAT(6X,*C*,I3,1X,A2,I3,2X,F12.2)
  108 FORMAT(6X,*C*,I4,   A2,I3,2X,F12.2)
  109 FORMAT(6X,*C*,I1,3X,A2,I4,1X,F12.2)
  110 FORMAT(6X,*C*,I2,2X,A2,I4,1X,F12.2)
  111 FORMAT(6X,*C*,I3,1X,A2,I4,1X,F12.2)
  112 FORMAT(6X,*C*,I4,   A2,I4,1X,F12.2)
   69 J=ALOG10(ZB)+1 $ GO TO (113,114,115,116)J
  113 WRITE(8,117)XB,XC,XE  $ GO TO 70
  114 WRITE(8,118)XB,XC,XE $ GO TO 70
  115 WRITE(8,119)XB,XC,XE $ GO TO 70
  116 WRITE(8,120)XB,XC,XE $ GO TO 70
  117 FORMAT(6X,*C*,I1,4X,A3,3X,F12.6)
  118 FORMAT(6X,*C*,I2,3X,A3,3X,F12.6)
  119 FORMAT(6X,*C*,I3,2X,A3,3X,F12.6)
  120 FORMAT(6X,*C*,I4,1X,A3,3X,F12.6)
   59 WRITE(8,139)
      WRITE(7,121)
  121 FORMAT( *EOR*/*MATRIX*)
  140 READ(6,122)RA,RB,RC
  122 FORMAT(4X,A3,10X,I4,3X,F12.6)
      ZB=RB+.5
      IF(RA.NE.ARB)GO TO 127
      J=ALOG10(ZB)+1 $ GO TO (123,124,125,126)J
  123 WRITE(7,128)RA,BA,RB,RC $ GO TO 140
  124 WRITE(7,129)RA,BA,RB,RC $ GO TO 140
  125 WRITE(7,130)RA,BA,RB,RC $ GO TO 140
  126 WRITE(7,131)AB,BA,RB,RC $ GO TO 140
```

```
128 FORMAT(6X,A3,2X,A2,I1,4X,F12.2)
129 FORMAT(6X,A3,2X,A2,I2,3X,F12.2)
130 FORMAT(6X,A3,2X,A2,I3,2X,F12.2)
131 FORMAT(6X,A3,2X,A2,I4,1X,F12.2)
127 REWIND 8
134 READ(8,132)RUST
132 FORMAT(A7,3A10)
    IF(RUST(1).NE.STRESS)GO TO 133
    WRITE(7,132)RUST $ GO TO 134
133 WRITE(7,135)
135 FORMAT(*EOR*/*IPSC*/*ARRIBA*/*EXIT*)
    REWIND 7
    STOP
    END
```

REFERENCES

1. Chu, Wesley W., Dr., "Optimal File Allocation in a Multicomputer Information System," Information Processing 68 - North-Holland Publishing Company - Amsterdam, pp. 1219-25 (1969).

2. Chu, Wesley W., Dr., "Optimal File Allocation in a Multiple Computer System, IEEE Transaction on Computers, Vol. C-18, No. 10, pp. 885-89 (Oct 1969).

INITIAL DISTRIBUTION

Copies	
2	DLSIE
1	Dr. Wesley W. Chu University of California Computer Science Department School of Engineer and Applied Science Los Angeles, California 90024
12	DDC

CENTER DISTRIBUTION

Copies		
2	18/1809/1809.3	
1	1804	
1	1805	
15	187	
10	182	
10	5214.1	Reports Distribution
1	522.1	Unclassified Lib (C)
1	522.2	Unclassified Lib (A)